SESSION

DOMAIN-SPECIFIC LANGUAGES + REQUIREMENTS ENGINEERING + OBJECT ORIENTED SYSTEMS, REAL-TIME SYSTEMS, AND FORMAL METHODS

Chair(s)

TBA

ISBN: 1-60132-446-4, CSREA Press ©

ISBN: 1-60132-446-4, CSREA Press ©

Automatic Discovery of Platform Independent Models of Legacy Object-Oriented Systems

Ismaïl Khriss[1], Gino Chénard[2]
[1] Département de mathématiques, d'informatique et de génie, UQAR, Rimouski, Québec, Canada
[2] Département d'informatique, Cégep de Rimouski, Rimouski, Québec, Canada

Abstract - *Software modernization is needed to perform the evolution of a system when conventional practices can no longer achieve the desired evolution goal. In their initiative called architecture-driven modernization (ADM), the Object Management Group proposes to use MDA to perform this modernization. However, ADM needs new tools and techniques to migrate systems developed on a non-model-driven environment to a model-driven environment. To enable this migration, we need to discover two kinds of models from the implementation of a legacy system: a platform independent model (PIM) and a platform description model (PDM). In this paper, we propose an automatic approach to discover a PIM from an object-oriented system source code. The PIM is expressed in a UML class diagram. The approach uses two analysis techniques and was validated on several systems written in Java and gives good results for a number of them.*

Keywords: Software evolution, software modernization, model-driven engineering, reverse engineering, software comprehensions.

1 Introduction

Corporations depend heavily on their software systems for their daily operations. Repeated changes to them result in a complex source code, which complicates the evolution process. This process can be divided into three categories: maintenance, replacement and modernization. The modernization is evolving a system when conventional practices can no longer achieve the desired evolution goal [1]. In face of complex legacy systems and poor documentation, modernization is often the only practicable solution.

Modernization can take many forms like platform change, source code translation, structural improvement, modularization, and data reengineering [1]. Even if modernization appears as an ideal solution, it is a tricky task. The prediction of its risks and costs is difficult. To face this problem, the Object Management Group (OMG) has created Architecture-Driven Modernization (ADM). Its main objective is to build a set of standards to simplify the interoperability of modernization tools [2]. ADM proposes modernization by using Model-Driven Architecture (MDA).

MDA is an approach of development where the principal elements are the models representing the various aspects of the software system at different levels of abstraction. The development process is based on successive model transformations resulting in the source code [3]. MDA distinguishes among those models, two important: the platform independent model (PIM) and the platform-specific model (PSM). The PIM of a system is its view from the platform independent viewpoint, and its PSM represents its view from the point of view of a particular implementation platform. An interesting aspect of the MDA initiative is the model-to-model (M2M) transformation applicable to a PIM to obtain a PSM. This kind of transformation requires a model called the platform description model (PDM)[1] which captures the knowledge of the implementation platform.

A PDM provides, for use in a platform-specific model, concepts representing the different kinds of elements to specify the use of the platform by an application [3]. For instance, the model of the EJB platform provides concepts like `Bean` and `Home` components. OMG recommends to use UML profiles for capturing PDMs. UML profiles provide a generic extension mechanism (stereotypes, tag definitions, and constraints) for customizing UML models to particular domains and platforms. They are expressed as UML class diagrams.

Our modernization process of a legacy system within ADM, is expressed in Figure 1[2]. It is the migration of a system from a non-model-driven environment to a model-driven environment. This process includes two stages. The first stage uses reverse engineering to discover abstract models of the legacy systems. More specifically, we have:

- a Text-to-Model (T2M) discovery to get a PSM from the source code;
- a M2M discovery to get a PDM from a PSM;
- a M2M discovery to get a PIM from a PSM and a PDM.

The second stage of the modernization process uses traditional forward engineering to obtain the source code of the new system using M2M and M2T transformations. We propose to describe a PDM in two kinds of views. The first view is a UML profile of the system's implementation platform. The second view is a set of transformation templates expressed in

[1] OMG uses the term platform model.

[2] The idea of illustrating the modernization process with a horseshoe is inspired from [1].

QVT (Query/View/Transformation) [4]. Those templates are parameterized models capturing the source code of the system's implementation platform.

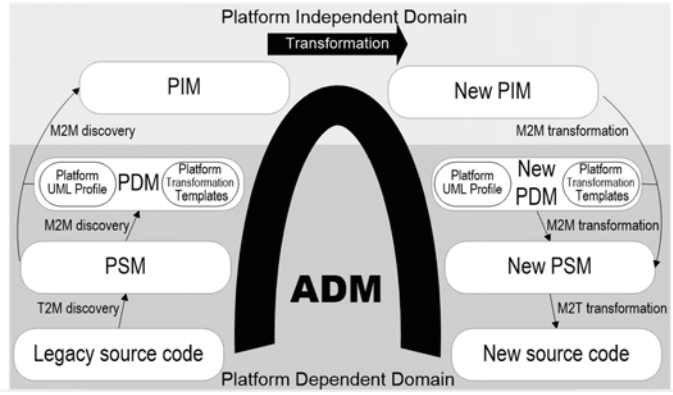


Figure 1. The modernization process within ADM

To simplify the discovery of the models representing the legacy system, we presented in [5] an approach to discover the first view of a PDM from the source code of a legacy object-oriented system and in [6] its second view. In this paper, we propose a new approach to discover the PIM of the system. Our approach takes as input the source code of the legacy system and the UML profile of its implementation platform; and gives as output the PIM of the system. The PIM is expressed in a UML class diagram. Our approach uses two analysis techniques for discovering the PIM of a legacy system: data flow analysis and analysis of identifiers.

This paper is organized as follows. Section 2 gives an overview of the approach through a running example. Section 3 presents the validation of our work. Section 4 discusses some related works. Finally, section 5 provides some concluding remarks.

2 Description of the approach

2.1 Definitions

We start by clarifying some terms that we will use in the description of our approach.

The *vocabulary of a legacy system* is the set of the words contained in its source code (comments excluded). This vocabulary (denoted by V) consists of: V_{PI}(the vocabulary introduced by the problem domain and therefore it is a platform independent), V_{PS} (the vocabulary introduced by the implementation platform and therefore it is a platform specific) and V_{PL}(the vocabulary introduced by the programming language.) We have $V = V_{PI} \cup V_{PS} \cup V_{PL}$.

A *pattern of an identifier* is a word (or a combination) contained as a sub string in its name. We obtain patterns by splitting identifiers on delimiters like uppercase letters or underline symbols. For example, the identifier `AccountEntity` contains three patterns: `Account`, `Entity` and `AccountEntity`.

Let `V1` and `V2` be two variables of the source code of a system. `V2` is related by data-flow to `V1` if the DFA of the source code discovers that `V2` takes the value of `V1`.

A *concept* is a formulation of a system's requirement from the problem domain or from the solution domain and is identified by its *name*. We distinguish two kinds of concepts: *platform independent concept (PIC)* and *platform-specific concept (PSC).* A PIC is a concept from the problem domain, and a PSC is a concept from the solution domain.

A PIC belongs to V_{PI} and it can be one of the following types: *classifier* (*class* or *interface*), *attribute*, *operation* or *parameter.* It indicates a kind of element needed to support a requirement from the problem domain.

A PSC belongs to V_{PS} and it can be one of the following types: *classifier, attribute, operation* or *parameter.* It indicates a kind of element needed to support a requirement from the solution domain.

A concept may have one or several property concepts. A *property concept* (PRP) helps its parent concept in the formulation of a system's requirement, whether from the problem or from the solution domain. A PRP has two attributes: *name* and *value.*

To guarantee the integrity of an implementation platform, a *constraint* can exist between two PSCs `C1` and `C2` and can be one of the following types:

- a *dependency constraint* occurs when the implementation of `C1` requires the implementation of `C2`;
- a *compatibility constraint* occurs when the implementation of `C1` is compatible with the implementation of `C2`;
- a *incompatibility constraint* occurs when `C1` and `C2` cannot be implemented together;
- a *refinement constraint* occurs when the implementation of `C2` presents different variations, and the implementation of `C1` is one of these variations.

A group of classifiers (operations respectively) is a set of classifiers (respectively operations) grouped on a criterion. To ease the description of our approach, we assume that a group always contains at least two elements, otherwise it is not a group. A criterion, to create one of these groups may be one of the following:

- *pattern-based*, it allows the grouping of classifiers or operations with a common pattern in their name;
- *relationship-based*, it allows the grouping of classifiers implements a common interface or inheriting a common classifier.

A group can be divided into subgroups according to one of the criteria defined above. Of course, an operation or a classifier can belong to different groups or subgroups. For example, from the classifiers listed in Table 1, the group of classifiers based on the pattern `Bean` contains these items: `AccountBean`, `BankManagerBean`, `BankTransactionBean`, `CheckingAccountBean`, `CustomerBean`, `DepositBean`, `SavingAccountBean` and `WithdrawalBean`.

2.2 Overview

To illustrate our approach, we use a "toy" banking system containing 47 classifiers as described in Table 1. We can see easily in this table by the classifiers' names some PICs like `Account` and `Customer` and some PSCs like `Bean` and `Home`. Figure 2 describes a part of the UML profile model of the system's implementation platform and Figure 3 describes its PIM which is the result of our approach.

The UML profile (see figure 2) shows the different concepts present in the implementation platform. Note the different prefixes, capturing the kinds of concepts have been added to ensure unique identifiers classes in the UML profile. We also see examples of existing constraints between these concepts. For instance, one can see in the figure that we find, among others, that all classifiers that implement the concept of classifier `Entity` also implement the `get` operation concept.

Table 1. Classifier names

Account	AccountBean	AccountCMP
AccountData	AccountHome	AccountUtil
BankManager	BankManagerBean	BankManagerHome
BankManagerSession	BankManagerUtil	BankTransaction
BankTransactionBean	BankTransactionCMP	BankTransactionData
BankTransactionHome	BankTransactionUtil	CheckingAccount
CheckingAccountBean	CheckingAccountCMP	CheckingAccountData
CheckingAccountHome	CheckingAccountUtil	Customer
CustomerBean	CustomerCMP	CustomerData
CustomerHome	CustomerUtil	Deposit
DepositBean	DepositCMP	DepositData
DepositHome	DepositUtil	SavingAccount
SavingAccountBean	SavingAccountCMP	SavingAccountData
SavingAccountHome	SavingAccountUtil	Withdrawal
WithdrawalBean	WithdrawalCMP	WithdrawalData

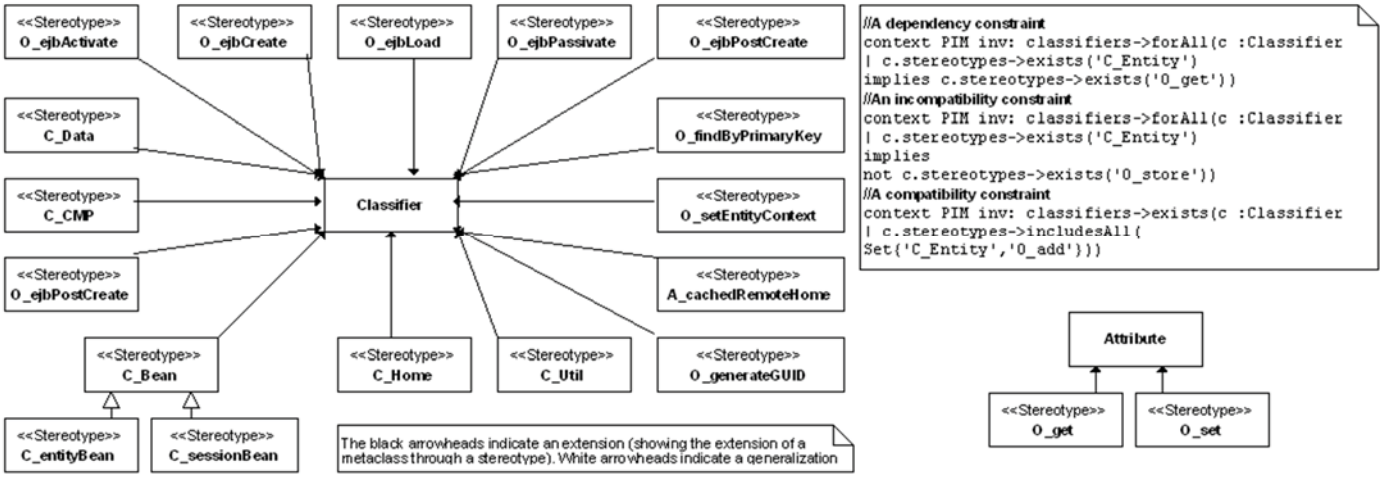


Figure 2. A part of the UML profile model of the system's implementation platform

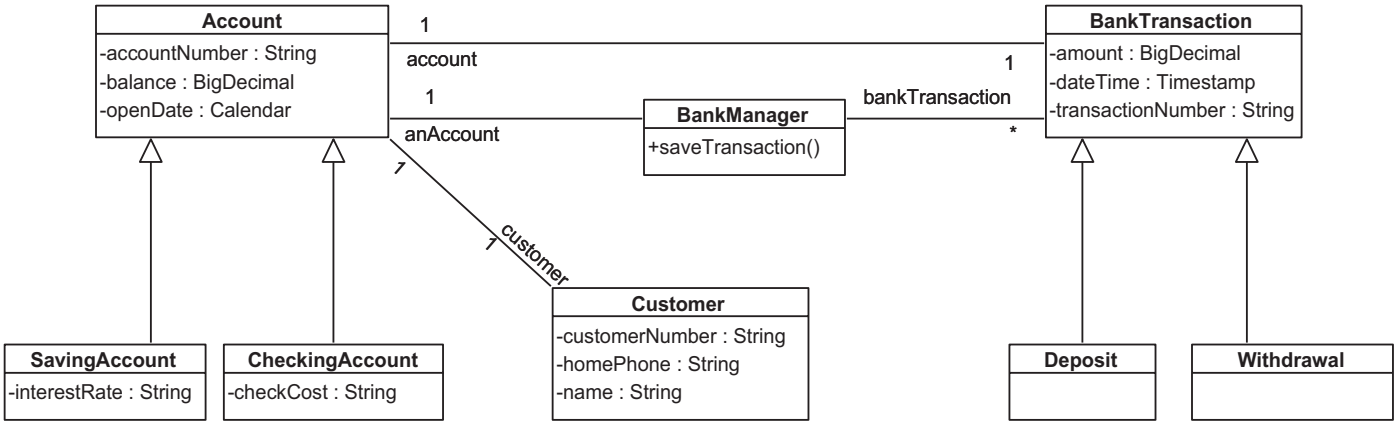


Figure 3. The PIM of the system

The PIM (see Figure 3) presents the domain classes of the system and the relationships between them. A banking transaction is performed on an account and is run by a manager (`BankManager`). Two types of accounts are available: `CheckingAccount` and `SavingAccount`. Two types of banking transactions are identified: `Deposit` and `Withdrawal`.

Our approach is based on the hypothesis that the code related to an implementation platform has a particular characteristic that distinguishes it from others: it is repetitive or semi-repetitive. This assumption was the basis of our previous work that proposes the discovery of the UML profile model of the system's implementation platform [5]. So, it becomes possible to identify the business vocabulary by filtering out the implementation platform related code. The business vocabulary extracted will enable the discovery of the PIM of a legacy system.

Our PIM discovery approach is divided mainly into 5 steps.

2.2.1 Discovering class PICs

The first step discovers class PICs by looking in the system's PSM for words representing them. This step uses the rule R1. This rule assumes that a class PIC implements at least two concepts of the implementation platform.

Rule 1 (R1). A classifier `C` in the PSM is related to a class PIC `P` whose name is that of `C` in which any occurrence of the name of a classifier PSC has been removed. This link is provided if `P` is bound to more than one classifier PSM.

In our toy system, we discover the following Class PICs related the classifiers: `Account`, `BankManager`, `BankTransaction`, `CheckingAccount`, `Customer`, `Deposit`, `SavingAccount` and `Withdrawal`. For example, we discover that the `Account` class PIC is bound to at least two classifiers (`AccountBean` and `AccountHome`) after removing in their names occurrences of the name of `Bean` and `Home` classifier PSCs.

2.2.2 Discovering attribute PICs

The second step of the PIM discovery aims to discover, for each PIC class, attribute PICs bound to it by searching for words in the vocabulary of the PSM that could represent them. This step is based on the rule R2. The rationale of this rule is that an attribute of an implementation platform will be repetitively present in a group of PSM classifiers implementing an issue of the platform.

Rule 2 (R2). An attribute of a classifier PSM bound to a PIC class is related to a homonymous attribute PIC if an attribute of the same name does not belong to each classifier of one of the subgroups of a classifier PSC in connection with a criterion of the implementation of an interface or classifier inheritance.

In our toy system, the attribute `accountNumber` of the PSM classifier `AccountData` bound to the class PIC `Account` is in turn related to the attribute PIC `accountNumber`, because no subgroup of a classifier PSC sees its classifiers share a homonymous attribute. This rule discovers that the attribute `cachedRemoteHome` of the PSM classifier `AccountUtil` also bound the class PIC `Account` is not an attribute PIC because all classifiers of a subgroup of the classifier PSC `Util` contain an attribute of the same name.

2.2.3 Discovering operation PICs

The third step of the PIM discovery aims to identify, for each PIC class, operation PICs bound to it by searching for words in the vocabulary of the PSM that could represent them. This step is based on the rule R3. The rationale of this rule is the same as the previous one.

Rule 3 (R3). An operation of a classifier PSM bound to a PIC class is related to a homonymous operation PIC if an operation of the same name does not belong to each classifier of one of the subgroups of a classifier PSC in connection with a criterion of the implementation of an interface or classifier inheritance and the name of this operation is not bound to a PSC operation. Each occurrence of a name of a PSC in the name and type of each parameter of this operation is removed.

In our toy system, since all classifiers of the group bound to the classifier PSC `Bean` have the operation `ejbAcivate`, this operation is not an operation PIC bound to the class `Account`. But, the operation `saveTransaction` of the classifier PSM `BankManager` represents an operation PIC related to the class PIC `BankMananger` because no subgroup of a classifier PSC sees its classifiers share a homonymous operation.

2.2.4 Discovering parameter PICs

The fourth step of the PIM discovery aims to identify, for each of operation PIC, parameter bound to it by searching for words in the vocabulary of the PSM that could represent them. This step is represented by the rule R4.

Rule 4 (R4). A parameter `P1` of an operation `O1` of a classifier PSM `C1`, where `O1` is connected to a homonymous operation `O2` of a PIM class `C2`, represents a parameter PIC if `P1` is bound to an element of the PIM, by first of the following conditions apply:

- if there is an attribute `A1` of `C1` where `A1` is bound to an attribute `A2` (respectively an association end `AE`) of `C2` and `A1` is linked to `P1` by data flow analysis, then `P1` is linked to `A2` (`AE` respectively);
- if there is an attribute `A1` of `C1` where `A1` is bound to an attribute `A2` (respectively an association end `AE`) of `C2` where the name is the longest pattern in the name of `P1`, then `P1` is linked to `A2` (`AE` respectively);
- if the name of `C2` is the longest pattern in the name of `P1`, then `P1` is linked to `C2`.

In our toy system, the parameter `amt` of the operation `credit` of the classifier PSM `Account` is bound to the operation PIC `credit` of the class PIC `Account`. This parameter becomes a parameter PIC because it is bound by data flow analysis to the attribute `balance` of the class PIC `Account`.

2.2.5 Discovering relationships between PICs

The last step of the PIM Discovery phase is to discover the relationships between the discovered PICs. This step is based on the rules R5, R6 and R7.

Rule 5 (R5). If the type of an attribute related to a class PIC `C1` PIC is another PIC class `C2` then this attribute is replaced by an association between `C1` and `C2`.

In our toy system, the attribute PIC `customer` related to the class PIC `Account` is of type the PIC class `Customer`, then an association between these two PICs is created and `customer` is no longer an attribute PIC of the class PIC `Account`.

Rule 6 (R6). If it exists an inheritance relationship between a PSM classifier `C1` bound to a class PIC `P1` and another PSM classifier `C2` bound to a class PIC `P2`, then an inheritance relationship is added between `P1`and `P2` in the PIM.

In our toy system, an inheritance relationship is created between the class PICs `CheckingAccount` and `Account` because an inheritance relationship exists between their related classifier PSMs.

Rule 7 (R7). If it exists an exception `E` thrown by an operation of a PSM classifier `C1` linked to a class PIC `P1` and the type of `E` is a PSM classifier `C2` related to a PIC class `P2`, then a dependency relationship is added from `P1` to `P2`.

No example is reported in our toy system.

3 Validation of approach

In this section, we introduce our validation framework. Then we present the results obtained by the validation process. To validate our approach, we have developed a tool to support it. We implemented our data flow analysis by using the library offered by Soot[3] for such analysis.

Our current prototype tool supports Java source code. It does not currently support other kinds of systems' artifacts such as deployment descriptor files. Therefore, some concepts of the PIM could not be discovered if they are only present in those kinds of unsupported artifacts. Our approach is scalable to support large systems as the underlying algorithm has a polynomial complexity.

3.1 Validation framework

To prove the relevance of our approach, we have applied our tool to five systems:

1. our toy system presented in this paper;
2. a Sales Management System (SGV) generated from a UML model using the MDA tool SourceCafe[4];
3. a banking system from a book on JDBC (BJDBC) [7];
4. a Virtual Shopping Mail (VSM) system from Oracle [5];
5. an OTN Financial Brokerage Service (OFBS) from Oracle[6];
6. the well-known Sun's EJB Pet Store[7].

The implementation platform of the first two systems is exclusively based on EJB. The implementation platform of the systems 4, 5 and 6 is partially based on EJB. The third system uses an ad-hoc implementation platform. Note also that even if the first two systems are based exclusively on EJB, they do not contain exactly the same set of PSCs since each uses a different implementation of EJB.

The search for items by reverse engineering can lead to three possible outcomes [8]. An item identified by the research results is a "true positive" if it is part of the expected results. It is a "false positive" if it is not part of the expected results. Finally, a "false negative" is something that should be find, but it is not. Based on these results, it is possible to develop metrics. In our case, we use the following two metrics: *precision* and *recall*.

We collect the following data: number of items discovered properly (true positive), number of wrongly discovered elements (false positive) and number of undiscovered elements (false negative). We use them calculate to calculate metrics.

The first metric is the precision. It assesses if the identified items are valid. The second metric evaluated is the recall. It assesses what percentage of the expected items are identified.

We have automated the process of validation. The elements expected of all tested systems were extracted manually after analyzing their source code. The availability of the EJB documentation and familiarity of the business area of these systems have helped this analysis. However, as this task is manual and often an implementation platform is not always rigorously implemented, we cannot affirm that the information is 100% correct or complete. These elements are placed in a file. When our prototype finishes his work, it compares its results with the contents of this file. Different traces are also generated for debugging purposes.

3.2 Validation results

Table 2 summarize respectively the results obtained from the validation of our approach. Note that the quality of the UML profile in the input has an impact on the results. In fact, an error caused by the erroneous discovery of the UML profile can bring other errors in the PIM discovery process. A detailed discussion of this issue and others can be found in a technical report [9].

[3] http://sable.github.io/soot/, 05-20-2015

[4] www.sourcecafe.com, 09-27-2010

[5] www.oracle.com/technology/sample_code/tutorials/vsm1.3, 09-27-2010

[6] www.oracle.com/technology/sample_code/tech/java/j2ee/fbs10g, 09-27-2010

[7] java.sun.com/developer/releases/petstore, 04-13-2012

Table 2. Results of the validation of the PIM discovery process

	Toy	SGV	BJDBC	VSM	OFBS	Pet
VP class PIC	8	4	2	14	10	6
FP class PIC	0	0	6	16	4	16
FN class PIC	0	0	3	5	5	31
Recall class PIC	100,00	100,00	40,00	73,68	66,67	16,22
Precision class PIC	100,00	100,00	25,00	46,67	71,43	27,27
VP attribute PIC	11	21	5	42	24	4
FP attribute PIC	0	2	20	44	11	82
FN attribute PIC	0	1	3	22	25	79
Recall attribute PIC	100,00	95,45	62,50	65,63	48,98	4,82
Precision attribute PIC	100,00	91,30	20,00	48,84	68,57	4,65
VP operation PIC	1	0	4	16	10	1
FP operation PIC	2	2	16	36	19	73
FN operation PIC	0	0	0	77	44	17
Recall operaiton PIC	100,00	100,00	100,00	17,20	18,52	5,56
Precision operation PIC	33,33	0,00	20,00	30,77	34,48	1,35
VP parameter PIC	0	0	3	0	1	0
FP parameter PIC	0	0	1	7	0	4
FN parameter PIC	0	0	5	30	46	19
Recall parameter PIC	100,00	100,00	37,50	0,00	2,13	0,00
Precision parameter PIC	100,00	100,00	75,00	0,00	100.00	0,00
VP association PIC	4	0	1	1	2	3
FP association PIC	0	0	2	5	9	7
FN association PIC	0	3	2	13	14	106
Recall association PIC	100,00	0,00	33,33	7,14	12,50	2,75
Precision association PIC	100,00	100,00	33,33	16,67	18,18	30,00
VP inhericance PIC	4	0	0	0	0	0
FP inhericance PIC	0	0	2	10	0	3
FN inhericance PIC	0	0	0	0	0	0
Recall inhericance PIC	100,00	100,00	100,00	100,00	100.00	100,00
Precision inhericance PIC	100,00	100,00	0,00	0,00	100.00	0,00
VP dependency PIC	0	0	0	1	2	0
FP dependency PIC	0	0	2	12	0	6
FN dependency PIC	0	0	1	0	3	1
Recall dependency PIC	100,00	100,00	0,00	100,00	40,00	0,00
Precision dependency PIC	100,00	100,00	0,00	7,69	100.00	0,00

4 Related work

Different approaches have been proposed for model-driven migration, to the best of our knowledge, no work has tried to find a PIM automatically by distinguishing between the concepts specific to an implementation platform and those dependent on the business domain of a legacy system. Below, we present some of those approaches.

MoDisco [10] is an extensible framework to develop model-driven tools to support modernization of legacy systems. To enable the reuse of components between modernization solutions, its architecture is organized in three layers: use-cases, technologies and infrastructure. The first layer aims at providing tools for a specific modernization use-case. The second layer offers specific components for one legacy technology, and the last layer provides generic components independent from any legacy technology.

In [11] and [12], the authors describe a framework to reverse engineering MDA models from object oriented code. The framework has three abstraction levels: models, metamodels and formal specifications. As in our work, they also aim to extract a PIM from legacy object-oriented systems. However, the PIM extracted is captured in use case and sequence models. For instance, in [11], they present an algorithm for discovering use case models. They start from public methods to perform their extraction. The model generated is then at very low-level functional view even if their approach tries to cluster the use cases. In [12], the authors describe an approach to extract sequence diagrams from Java code by using the MoDisco platform [10]. In our opinion, the discovered models do not really describe a PIM since their works do not perform any distinction between platform independent and platform dependent concepts.

Li et al. [13] describes an approach for discovering use case model from the source code by using reflection instrumentation techniques during the execution of a system under analysis. The relations among use cases are mined using a set of rules. Composite use cases are also built by using the extracted relations.

Pérez-Castillo et al. reports a case study on business process recovery from legacy information systems [14]. Their recovery procedure is based on their framework called MARBLE (Modernization Approach for Recovering Business processes from LEgacy systems). The framework uses ADM and its four abstraction levels to recover business knowledge from legacy information systems. The highest abstraction level represents the business process model. The transition from one level to another is done through model transformations. In our opinion, the quality of the resulting business process is questionable. This is why, they propose that the last transformation should be supported by the manual intervention of business experts to refine the business processes obtained.

Sadovykh et al. [15] report a real case of modernization using ADM from a legacy C++ system to Java. The creation of their PSM is based on human experts who have to decide which part of PSM is related to the problem domain and which is related to the solution domain. They claim that it is really difficult to automate this distinction because it needs the input from experts in the problem and solution domains.

Fleurey et al [16] present a semi-automatic approach for model-driven migration. They automatically extract a PSM from a legacy system and use transformations to generate a PIM and a PSM in the new target architecture. These transformations are defined manually.

Chen et al. [17] present an approach which analyzes assembler source code to extract a set of models representing the software components, and its architecture described in a formal language. First, a PSM is created from the source code of the legacy system. Then, decomposition techniques are used to restructure the PSM. Next, abstraction rules are applied to

create a PIM from the restructured PSM. The approach presents two limitations. First, the abstraction rules have to be defined manually. Second, the extracted PIM cannot be reused since the names of its model elements are not meaningful as the source of extraction is an assembler source code.

5 Conclusion

In this paper, we proposed a new approach to enable the modernization of legacy object-oriented systems through a model-driven migration. We presented the key elements of the approach. We described an automatic process for discovering the PIM of the legacy system. This PIM is expressed in a UML class diagram. We finally presented the results of our validation process.

The main contribution of our approach lie in the automation brought upon by the deployed algorithm in the discovery process of the necessary model from the source code of a legacy object-oriented system. This discovered model will achieve better the migration of this legacy system to a model-driven environment.

6 References

[1] R. C. Seacord, D. Plakosh, and G. A. Lewis, *Modernizing Legacy Systems: Software Technologies, Engineering Processes, and Business Practices*: Addison Wesley Professional, 2003.

[2] OMG, "Architecture-Driven Modernization Task Force. Architecture-Driven Modernization scenarios," 2006.

[3] J. Miller and J. Mukerji, "MDA Guide Version 1.0.1," Object Management Group omg/2003-06-01, 2003.

[4] OMG, "MOF QVT Final Adopted Specification," 2005.

[5] G. Chénard, I. Khriss, and A. Salah, "Towards the Discovery of Implementation Platform Description Models of Legacy Object-Oriented Systems," presented at the Workshop on Processes for Software Evolution and Maintenance (WoPSEM 2010) Massachusetts, United States, 2010.

[6] G. Chénard, I. Khriss, and A. Salah, "Towards the Automatic Discovery of Platform Transformation Templates of Legacy Object-Oriented " presented at the Models and Evolution (ME) 2012 workshop a satellite event at MoDELS 2012, Insbrusck, Autriche, 2012.

[7] G. Reese, *Database Programming with JDBC and Java, Second Edition*: O'Reilly & Associates, Inc., 2000.

[8] I. Philippow, D. Streitferdt, M. Riebisch, and S. Naumann, "An approach for reverse engineering of design patterns," *Software and Systems Modeling,* vol. 4, pp. 55-70, 2005.

[9] I. Khriss and G. Chénard, "A new approach for model-driven modernization of legacy object oriented systems," Université du Québec à Rimouski,2016.

[10] H. Bruneliere, J. Cabot, J. Frédéric, and M. Frédéric, "MoDisco: a generic and extensible framework for model driven reverse engineering," presented at the Proceedings of the IEEE/ACM international conference on Automated software engineering, Antwerp, Belgium, 2010.

[11] P. Claudia, M. Liliana, and F. Liliana, "Recovering Use Case Diagrams from Object Oriented Code: An MDA-based Approach," in *Information Technology: New Generations (ITNG), 2011 Eighth International Conference on*, 2011, pp. 737-742.

[12] L. Martinez, C. Pereira, and L. Favre, "Recovering sequence diagrams from object-oriented code: An ADM approach," in *International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE), 2014*, 2014, pp. 1-8.

[13] Q. Li, S. Hu, P. Chen, L. Wu, and W. Chen, "Discovering and Mining Use Case Model in Reverse Engineering," in *Fuzzy Systems and Knowledge Discovery, 2007. FSKD 2007. Fourth International Conference on*, 2007, pp. 431-436.

[14] R. Pérez-Castillo, I. García-Rodríguez de Guzmán, M. Piattini, and Á. S. Places, "A case study on business process recovery using an e-government system," *Software: Practice and Experience,* vol. 42, pp. 159-189, 2012.

[15] A. Sadovykh, L. Vigier, A. Hoffmann, J. Grossmann, T. Ritter, E. Gomez*, et al.*, "Architecture Driven Modernization in Practice - Study Results," presented at the Proceedings of the IEEE International Conference on Engineering of Complex Computer Systems, Potsdam, Germany, 2009.

[16] F. Fleurey, E. Breton, B. Baudry, A. Nicolas, and J.-M. Jézéquel, "Model-Driven Engineering for Software Migration in a Large Industrial Context," presented at the Proceedings of the international conference on Model Driven Engineering Languages and Systems, Nashville, United States, 2007.

[17] F. Chen, H. Yang, B. Qiao, and W. C.-C. Chu, "A Formal Model Driven Approach to Dependable Software Evolution," presented at the Proceedings of the International Computer Software and Applications Conference, Chicago, United States, 2006.

Domain Modelling and Language Issues for Family History and Near-Tree Graph Data Applications

C.A. Maddra and K.A. Hawick
Computer Science, University of Hull, Cottingham Road, Hull HU6 7RX, UK.
Email: {c.maddra, k.a.hawick }@hull.ac.uk
Tel: +44 01482 465181 Fax: +44 01482 466666

April 2016

ABSTRACT

Domain-Specific Modelling is a powerful software engineering approach to building complex software systems. Domain-Specific Languages provide a powerful way of capturing and encapsulating the applications vocabulary and central ideas for whole families of software applications. We describe some domain-specific modelling approaches and techniques based around the application domain of family history or genealogy systems where the central data model is a near tree-like structure. We discuss how: querying; modification; and aggregation patterns of operation can be implemented in a number of ways for this domain application. We explore the scalability of the DSL approach and discuss wider issues in model and language development.

KEY WORDS
software design; tree data; genealogy applications; domain-specific languages.

1 Introduction

Domain specific modelling [23, 39] provides an important software engineering approach to formulating software designs based upon important features such as data models and key data structures for specific application areas. In this article we focus on the application area of family tree storage and analysis software.

Domain Specic Languages (DSLs) [13, 34] are little programming languages, that are often customizable [4] and are typically designed for use around exclusively around a chosen domain. Current popular general purpose languages such as C, C# and Java all focus on the implementation level [6, 18] of software at machine or virtual machine level. DSLs aim to provide what can be seen as a suitable compromise between hard to express concrete machine executions at implementation level and hard to execute vague human conversation at the solution level. The holy grail of a DSL project is to capture the essence of a family of applications within a DSL to allow separation of concerns from the implementation boilerplate and the solution creativity. This then allows multiple applications to be created without having to waste intellectual capital on the code intelligently automated by the DSLs back end. The separation can also allow for portability across different architectures through using different back ends.

Popular examples of DSLs include CSS, Regex, Flex/Bison, Blitz++, Make and JMock expectations. DSLs are not a mere set of libraries allowing you to plug other peoples algorithms into your solution at implementation level but a language which allows writing new algorithms and reusing others in the domains level of abstraction. This is code automation rather than just manual reuse. DSLs allow the user to work at the correct level of abstraction for writability, readability and maintainability. Writing at this level of abstraction captures the intention of the programmer rather than the byproduct of their intention in the form of an implementation [30].

The DSL approach has found uses in applications areas including: automatic structural optimisation for agent-based models [19]; operating systems development [7]; tile assembly in biological applications [8]; image processing [14]; wireless sensor nets [29]; and network systems configuration [36]. DSLs also find use more directly in development of software tools and software itself such as software version conversions [12]; code generation [24]; and program generation [27].

Generally DSLs are implemented as a user interface over a semantic model to maintain separation of the user interface and back end logic. A semantic model is an executable implementation of a domain generally in the form of one or more libraries implemented in a high level programming language. The DSL acts as the user interface to this semantic model, shielding the user from the technical implementation and allowing them to focus on the solution space. There may be many DSLs over a single semantic model allowing programmers to only concern themselves with the sub-domains they want to while retaining all code in a single base. Although there is no explicit extra functionality, a convenient way of populating the semantic model allows for more complicated systems to be created without concern of the underlying implementation. In other words, a DSL can allow the programmer to spend their intellectual capital on the problem domain rather than the solution space. The enhanced leverage on the underlying semantic model allows greater efficiency of thought and thus more functionality for the same work. This can be likened to the increase in productivity the manufacturing industry has seen from their progressive increase in automation and componentisation.

The definition of what makes a language a DSL is a blurry one, with languages being more or less of a DSL rather than a direct cut-off. This is especially true for internal DSLs which can be classed as a separate language to their host or merely a feature of it. The underlying point is not if a language is a DSL but if it can reduce the abstraction gap between the problem and solution domain compared to its general purpose counterparts. The successful use of DSLs relies on the mindset and practices of the programmers as much as the languages themselves. The way a language is used in a project can dene whether it is a DSL in that project or not as much as the features of the language.

There are a number of DSL tools and frameworks available [10, 11, 33]. Generally, a DSL generally conforms to this checklist of desirable features: 1) Clearly Identifiable Domain; 2) Limited Expressiveness; 3) Fluent Language Structure. It is helpful to look at DSLs though a concrete domain so patterns can be mapped directly to a set of related example implementations, which also allow us to show the compound nature of improvements in language. It is hard to display the advantages of domain specificity in a vacuum. Small changes in the level of abstraction can simply be mapped onto a library call or macro apparently removing the benefit of DSLs over GPLs, the advantage of these small changes in abstraction adds to a large change in the overall abstraction because it changes the level that the programmer needs to think. Working with macro's and library calls locks the programmer into the implementation level of a solution, and can quickly become clumsy in a comment littered solution. Domain specific solutions lock a user into thinking at the intention level rather than the implementation.

Our chosen domain of genealogy is suited to domain specific languages because it is well established with domain jargon which can be used in DSLs and is a firm domain which has guaranteed future work making a DSL worth the initial effort. Genealogical analysis is a new area which is becoming more important as we get more data and DSLs have a strong role to play in this as scientists realise it's not just about how much data you have but also the ease in creating novel ways to analyze this data. DSLs can be used as a method of input for genealogical analysis which can improve the quantity and quality of work possible given similar circumstances [26, 31]. Genealogy is a useful domain because it's a concrete application of directed graphs which link multiple genetic families together into a collection of blood lines forming tree-like graphs. Graphs are an exciting area for computer science because of their applicability as network oriented databases.

Genealogy file types can also be used to create single records of people which are not interlinked but stored merely for the recording of a persons information. For this report this is ignored as it is not important for genealogy research unless used to merge into trees and is a subset of the use of DSLs for full tree creation and modification.

We want to investigate the issues for using domain specific languages in graph like environments, with the concrete example of genealogy. Currently the genealogy area is filled with graphical applications to edit, view and share data because it is suitable for the layman market. We want to consider the useful techniques for and create a case study of the use of textual internal DSLs in this area to see what can be performed without the use of a graphical user interface.

Our article is structured as follows: In Section 2 we review background ideas on the genealogical application domain. We focus on technical aspects we have explored in our present project in Section 3 and in Section 4 we provide some case study details on our approach. In Section 5 we give a discussion of the implications of using Python and other tools for this sort of system and offer some conclusions and areas for further r work in Section 6.

2 Genealogy Domain Review

Genealogy is one of the oldest hobbies in the world. Whether people are drawn to it because of intrigue from a blank slate or tradition in the family technology is making finding, storing and sharing your ancestors far easier [3, 35]. Historically Genealogy has been done through paper based systems such as census data and birth records which are reasonably difficult to gain access to. With the advent of cloud oriented genealogy services such as ancestry.com and familysearch.org anybody can access these records to build their own family trees. These cloud oriented systems also remove the problem of scale as huge amounts of records can be searched through at a speed far faster than a human could search through even a few pages of paper records. Digital searching allows for complex searches including wild cards, conditionals and inferred missing data which are time consuming and difficult to perform by a human because of manual calculation and problems with remembering combinations of conditions.

Currently the genealogy market is dominated by on-line services because of the convenience of the cloud for storage and social networking. Big providers such as ancestry.org are active in improving the amount of data available to genealogists and the level of analytics which can be performed on this data [2].

One of the biggest offline suites is the gramps project, which is an open source python genealogy IDE which focuses not only on GEDCOM but also it's own Gramps XML format. The main concept behind Gramps' features is to allow the user to focus on their genealogical research in as much or as little detail as they want with peace of mind that it's all safely stored, searchable and sharable whenever they want. This is done using a GUI with limited options for extension, fitting its goal audience of hobbyist family tree researchers. Gramps is officially released for Linux with community supported windows and mac releases, this means different operating systems will give a different experience.

A long running genealogy suite is LifeLines which has been maintained as an open source project after the creator Tom Wetmore stopped working on it in 1994. Lifelines is based completely around GEDCOM but not any particular standard. It allows you to extend the default GEDCOM standard to suit your needs although this will cause comparability issues and data loss when sharing your genealogy data. Importantly for us lifelines pioneered the idea of using a report generation language to produce all the reports of the program rather than relying on preset report types. This DSL based approach allowed lifelines to perform any reasonable report generating task which could be imagined leaving it the choice of GenWeb and GenServ for their

backend This report generation language is covered further in similar genealogy DSLs.

GEDCOM (Genealogical Data Communication) is a specification designed to pass genealogical data between parties. It was developed by the Church of Jesus Christ of Latter-day Saints to aid genealogical research. GEDCOM is widely supported by genealogy software and has a simple reference based lineage-linked structure to link families together. [1].

For cross platform sharing to work the Genealogical Data Communication (GEDCOM) standard must be consistently followed. Sadly when big players such as Family Tree Maker do not follow the GEDCOM standard by adding in new tags it means software which uses GEDCOM has started to accept non-officially valid GEDCOM as good input to appease users. This makes it impossible to create a gedcom parser which will accept all 'GEDCOM' and clarifications on the subset accepted must be made. GRAMPS solves this problem with a third party add on called GEDCOM-extensions which supports pervasive changes to the gedcom standard. We focus on GEDCOM in this project because it is the current de-facto standard for sharing genealogical data.

The GRAMPS project setout with the goal of making a portable, machine and human readable format which can be read and written without data loss. The GRAMPS documentation alludes to XMLs compatibility with source control software when uncompressed and small file size when compressed. For performance reasons the XML representation is not used as the internal database for GRAMPS but as an export format. This format has not been used by this project because of it's limited compatibility with software suites aside from GRAMPS. Including GRAMPS XML as an extension to this project would be possible as the domain specific language layer would not need changing as the semantic models interface could remain the same.

The file format of Family tree maker, Ancestry.com's flagship software and claimed to be the number 1 selling genealogy software. The format is called FTW because Family Tree Maker was called Family Tree Maker for Windows in previous version. This format is proprietary and requires the Family Tree Maker software to convert to other formats. There have been complaints about the poor conversion to other formats as well [22]. This format is not suitable for this projects research because it's not open and it's associated tool is paid software aimed towards editing in a graphical setting.

Zandhuis presents the "Semantic web" as a way of storing genealogical data in an open and extensible way rather than the current file formats used. It presents a first attempt at a genealogical ontology to start the discussions for a standardized ontology with direct goals towards improving the current problems in exchanging genealogical data and automating it's processing. Integrity checks and intelligent processing can be performed on the genealogical data to check for constancy and potential errors. The ontology formalizes things such as events timing before, after or during a time period making automation of searching and analysis possible. This work could be complimented by a DSL which allows the input and manipulation of the semantic web in the same way this project deals with gedcom. Given that satisfactory APIs are provided by the ontology the DSL could give rich feedback based only on interfaces with the extensible ontology.

Displaying Genealogical data is an issue related to what the user needs to ascertain from the data. The traditional family tree is a generational graph starting from a single person as the root, but this method of notation does not show people in relation to time and poorly scales as descendants have families of their own causing excessive growth [25].

Mass scale genealogy is made possible by crowd sourcing many peoples research into combined files to aid everybody's research. The quick and successful merging of analogue genealogy data such as paper records relies on new copies being made and hand checking through existing records. This is tedious and error prone but has the advantage of interactions between researchers and knowledge known by the researcher but not stored in records may be combined with the existing information to aid merging or add new information. This process can be aided through digitized records because much of the tedious comparing work can be automated, leaving the researcher to merely confirm assumptions by the merging program. Unfortunately this can lead to errors in false-positive merges and false-false misses of merges which could be avoided by a human's intuition, errors due to incorrect records on either side are hard to avoid aside from common sense checks such as timeliness and location. The full automatic merging of digital files is also an issue, especially without strong standardization on what information is to be included in records (even if this requirement is just a method for saying if a field is not available) and how to format these records to make digital searching and sorting possible in all cases. Domain specific languages and enforced underlying models allows these to be enforced and inform the user where input does not meet the standard which is not possible in analogue or pure file-format input.

The semantic model idea allows the data to be stored in any format which matches the DSLs interfaces, this can be combined with other research areas such as work in aiding the automated process of using graph algorithms to merge family trees have been developed [38].

There are a number of established Genealogy DSLs available. We describe some of these and their features.

Life lines [37] has a reports language which you can specify any type of report you would like, with common reports programs given with the distribution as examples. This gives flexibility in comparison to the common solution of pre-made templates which you merely provide input to at the cost of the removal of simple GUI modifications of pre-made templates. The LifeLines language is implemented as many function calls in C. The decision to use C reduces the potential for internal DSL tricks as C has low language extension and tinkering support, this leaves the lifelines language close to the implementation level rather than bringing the user up to their solution level of abstraction.

This language has been used during undergraduate dissertation at the University of Hull but without success because of not being able to get to grips with the language referring to the simple examples and lack of an active online community. The documentation for the language is example based with the most common reports already having programs written up. Sadly the complexity of the reports language still makes it difficult for a novice to

understand how the reports are generated from the given code, this would be solved with an intention level approach. This example from the Lifelines language documentation [37] prints the ancestry of an individual. The individual is specified at runtime using the terminal.

```
proc main ()
  {
    getindi(indi)
    list(ilist)
    list(alist)
    enqueue(ilist, indi)
    enqueue(alist, 1)
    while (indi, dequeue(ilist)) {
      set(ahnen, dequeue(alist))
      d(ahnen) ". " name(indi) nl()
      if (e, birth(indi)) { " b. " long(e) nl() }
      if (e, death(indi)) { " d. " long(e) nl() }
      if (par, father(indi)) {
        enqueue(ilist, par)
        enqueue(alist, mul(2,ahnen))
      }
      if (par,mother(indi)) {
        enqueue(ilist, par)
        enqueue(alist, add(1,mul(2,ahnen)))
      }
    }
  }
```

Another DSL similar to our case study was written in 2013 by Paul Johnson [21]. This DSL is implemented in perl which has good support for extension by internal DSLs and Johnson has taken advantage of this to provide a good object oriented domain specific experience for the user. This DSL's distribution includes an unfinished 'lines2perl' program which converts lifelines programs into the DSL. This DSL fulfils the same role as the family tree manipulation DSL from this project and is extremely similar once the differences between a pearl internal DSL and python internal DSL are mitigated.

open a GEDCOM file and print out the names and birth dates of all individuals.

```
my $ged = Gedcom->new(
    grammar_version => "5.5",
    gedcom_file     => $gedcom_file,
    read_only       => 1);

for my $i ($ged->individuals)
{
 for my $bd ($i->get_value("birth date"))
 {
   print $i->name, " was born on $bd\n";
 }
}
```

3 Building Domain Specific Languages

The central idea of the DSL community is to create a language which works at the correct level of abstraction for a chosen finite domain, generally moving towards a more declarative environment to express solutions. Having a defined grammar for the domain means problems with overlap between domains, for example 'agents' having separate meanings for artificial intelligence and agent based modelling are explicitly dealt with as the domain of the language sets the context and semantics for the jargon within the language.

Writing languages in their interpreter allows us to perform interactive programming as is talked about in early 4th generational languages literature as monologue vs. dialogue [28]. With a concrete domain, 2-way conversations between the programming language and the programmer are useful because repercussions of a statement on an unknown dataset cannot be known before execution. An example of this is when asking for a persons mother, if they have more than one mother a simple question from the language run time environment will notify the user while allowing the syntax for regular events clean. In our case study this problem is solved by using mother() which selects the only available mother or if a conflict exists requests the user which they would like to use. If the user knows which mother they would like to use before performing the statement they can specify their chronological index or name.

How a DSL is related to it's host language denotes whether it's internal or external. Internal DSLs are hosted within an existing GPL and implemented using there language features. Internal DSLs are popular within the functional community because of the extensibility of languages like LISP and Ruby. Early languages such as C don't offer much support for internal DSLs because of their lack of extendable features. Recent imperative/-functional mixed languages such as SCALA and Python are a mix between these two extremes. The internal DSL code is generally mixed in with standard GPL code seamlessly and compiled or executed during the same process.

External DSLs are an explicitly different language to their host language which has it's own parser. As external DSLs have a separate parser to their host language they have complete control of their syntax and semantics rather than just what's afforded by the extensibility of the host. This extra control lends itself to domains where the model of execution or order of operations cannot be elegantly expressed using standard GPL syntax such as database querying. external DSLs can be intended for use as stand alone files e.g. configuration files or as explicit sections within GPL code eg regex.

An additional definition of "active libaries" [5] can be made for DSLs which although they are treated as internal DSLs they play a role in the compilation or execution of their code allowing for compile and/or run time domain specific optimization to be performed. Active libraries are popular with the extensible languages communities such as LUA and have been used in projects such as Husselmann's [20]. For this project we have created a set of internal python DSLs. Internal DSLs have been chosen because they allow seamless inter-operation between not only the DSLs themselves but also any other python code which is appropriate for the graph data created by the DSLs from the gedcom files. Generally speaking internal DSLs also have a lower implementation time to their external counterparts. Creating multiple languages each with a niche allows for a unique and appropriate view for each different sub-domain, layering these over the same semantic model helps code re-use.

In the traversal DSL, we use fluent interfaces for traversing the family tree. These allow a user to start at any person in the tree and move through their relatives in a single command. This is compatible with the use of run-time dialogue within the language as mentioned earlier so the fluent interface can be guided where they're are conflicts or multiple options.

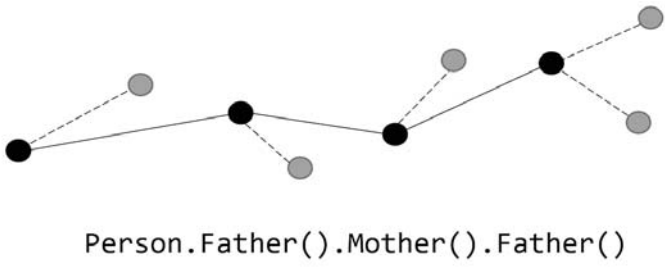


Figure 1: Fluent Interface to system.

It is considered good practice to separate the implementation of a DSL from it's front end [9]. Our DSLs are layered over a single semantic model because they are all based around the same software family and using the same boilerplate. Using the same backbone for each DSL allows code reuse and DSL inter operation without extra implementation time from re-writing code and creating interfaces. A semantic model is the semantics of a domain area in code, generally as a library or a set of libraries.

The semantic model is perfectly capable of being used by itself without the intervention of the DSLs, the DSLs are merely a level of abstraction above the semantic model allowing for better communication in that domain. We have chosen to have several DSLs over the same model so they can be different 'looking glasses' into the sub domains of our choice. Different users want to see different things from the same domain and this can be solved by using different languages for different purposes.

The downside of this is for people who wish to use multiple sub domains will have to learn multiple DSLs, although this is a direct trade with having to learn more language features from a single bloated language. There's an argument that by learning the principles large general purpose languages in general you learn the principles of how to program in any similar language.

Figure 2: System architectural model.

Figure 2 shows how Python can be integrated into the model architecture.As mentioned previously one of the benefits of using an internal DSL is access to the host language's existing ecosystem. Python is a particularly strong candidate for this because of the large amounts of scientific libraries and frameworks available such as Seaborn, Bokeh and Pygal for visualisation. Pre-existing tools to communicate with external languages and toolkits make this advantage even stronger because external interoperation with status-quo tools is dealt with out of the box. An example of this would be a user of our case study using Pymatlab to send the results of a query from our DSL into MATLAB for a pre-made script to produce graphs on the members connectivity.

4 Python Internal DSL

This project's back-end implementation expands on work by Nikola Skoric [32] (which itself was expanding on work by Daniel Zappala of Brigham Young University) who has created a GEDCOM parser which implements a subset of the GEDCOM 5.5 specification. Expanding on an existing GEDCOM parser saved time in picking the tags to implement and implementing them. This parser allows us to the the GEDCOM information and ingest it into an object oriented language representing it as a graph stored within a dictionary. This back end forms our semantic model of the family tree area, this semantic model can perform all our supported tasks in our domain even without the use of our front end DSLS.

The front-end DSLs have been created to allow sub-domains of genealogy to have a distinct DSL each which use the dominant data type of that sub domain as the basis for actions with sub-domain specific jargon. Doing this allows us to work around the essence of each sub-area for example the traversal DSL is based around a tree and the family tree DSL is based around individuals and families. You can use the jargon of a family tree or data type tree in these DSLs interchangeably because the difference between in the DSLs is merely the users framing because the underlying representation is the same.

The DSLs are intended to be used within the python interpreter so the user can receive feedback as they program. The DSLs can also be used to create python scripts for re-use (for example doing common operations on multiple files at different times) or writing a new program or language as a level of abstraction above these DSLs. An example of a simple further level of abstraction over these DSLs such as a check box GUI are useful for people who do not wish to program at the expense of depth of expression. An interesting further level of abstraction would be a visual family tree manipulation language designed towards touch screens because it could provide intuitive manipulation of family trees which requires no computing knowledge at all.

As this project has been built on top of an existing GEDCOM parser project the handling of additional known and emerging data formats isn't ideal to implement. This implementation decision was an error at the start of the project which cannot be changed without re-writing the projects software. Using parsing technology such as Pyparse or funcparlib to filter GEDCOM data into a custom made, extensible model for the area of genealogy would have been a better solution because further work could be done with other back-end models and different input/output data types.

GEDCOM files store genealogy data using a lineage linked data format, this forms one or more graphs containing all the nodes. Order within the file itself is insignificant as the people and families are identified using reference tags.

The storage of the data in python is inherited from Nikola Sko-

ric's parser which this project builds upon. GEDCOM files are stored as individual lines which contain all the data and as containing objects for people and families which serve as encapsulating references.

5 Discussion

There are a number of advantages of using a DSL over that of using a model directly. Given that software reuse through a semantic model (in this case a framework) is beneficial because of the inherent benefits of software reuse we can ask: as a semantic model can be used directly, why is using a DSL beneficial. The theory behind using a DSL is to improve the programmers leverage on a certain domain, whereas the semantic model is merely designed to encapsulate business logic and facilitate software reuse. The separation of a back end model and a front end DSL aids the maintainability and scale-ability of the solution. This projects DSLs can provide a case study to this question.

Determining a qualitative measure such as leverage on a domain is a difficult task and has been tackled by many authors. A quantitative method of grading code is lines of code, they're issues with this method although for these purposes it is a viable indication [15–17]. The Lines of code metric can only be considered viable if both examples are written in good faith to be the most appropriate code for the task. For example not splitting statements into several lines to effect results or vice versa.

Traditionally genealogy data has been dealt with through word of mouth and paper records. The growing adoption of digital storage has opened up the question for how can we digitally store and manipulate genealogy data with at least the same amount of expression, compatibility and longevity as the previous paper records. To challenge this we must ask: where does paper excel and fail and how can a digital system improve upon the current situation.

Paper records are extremely flexible because the owner has complete control of what to write without needing any skills aside from reading and writing. This also leads to a problem because with flexibility comes increased chance for non-standardised, illegible or inconsistent records. Paper is also a long term storage solution depending on the type of paper used, environment it's stored in and the organisation of the storage system. (MORE)

The main candidates for digital user interfaces are:

- Direct file format editing e.g. GEDCOM
- Text based programmatic editing e.g. through a programming language
- Graphical editing e.g. manipulating a visualisation of a tree
- Form based applications e.g. GRAMPS
- Web based applications e.g. familysearch.org

These different strengths and weaknesses mean different audiences can benefit from different user interfaces. For example for quick viewing and editing of genealogy data applications are suitable for all audiences including those without programming knowledge, for more complex or repetitive tasks then a programming language can save time, effort and reduce mistakes. The benefit of the domain specific language over a semantic model approach we have taken in this project is different forms of user interface can all work on the same model. This means once you have spent the time developing the DSL developing graphical or form based interfaces above this DSL could be faster and cross-compatible with code written in the DSL.

6 Conclusion

In summary the intent-level communication lets the domain expert communicate in the language of the domain and their intention to be automatically moved to the implementation level by the DSL environment. We have found this reduces the time to implement manipulations, analysis and traversals of a GEDCOM file in relation to directly using the GEDCOM files or writing your own algorithms in python directly.

Using an internal DSL allows the project to nativity inter-operate with existing python libraries. This would be useful if for example the manipulation DSL was used with modifications performed by some existing graph analysis library.

The DSLs created for this project can be used as the backbone of future projects to reduce their workload. This goes with the extensible software philosophy of kernel and configuration being separate to allow malleable, reliable software. A future project in the genealogy DSL area could be a visual DSLs for modifying family trees. This project would be angled towards the applicability of visual DSLs where the domain especially lends itself to tactile manipulation and a case study for the use of a domain specific language to aid in the creation of another domain specific language in the same software family.

A possible future project could investigate the related area of graph-oriented DSLs, this could be considered a level of abstraction up from this project which focus a certain sub domain of graphs (family trees).The graph domain is exciting because of current developments in the graph database community bringing new, difficult problems to be solved into the space. Example projects could be graph manipulation languages, graph database languages and quantitative experiments based around these languages.

We believe the software engineering approach of developing DSLs that implement ideas embodied in models has considerable potential, and that this potential has not yet been widely exploited for many applications domains, particularly those with complex underpinning features.

References

[1] Allen, J.: Gedom(future direction) announced by family history (May 1998), https://listserv.nodak.edu/cgi-bin/wa.exe?A2=ind9805A&L=GEDCOM-L&P=R2&I=-3&T=0

[2] Baker, P.: How ancestry.com uses big data. Fierce Big Data (2014), http://www.fiercebigdata.com/story/how-ancestrycom-uses-big-data/2014-08-04

[3] Burton, J.: Genealogy issues paper. In: AITSIS Workshop on Genealogies (2002)

[4] Cong, J., Sarkar, V., Reinman, G., Bui, A.: Customizable domain-specific computing. IEEE Design & Test of Computers March/April, 6–14 (2011)

[5] Czarnecki, K., Eisenecker, U.W., Glück, R., Vandevoorde, D., Veldhuizen, T.L.: Generative programming and active libraries. In: Selected Papers from the International Seminar on Generic Programming. pp. 25–39. Springer-Verlag, London, UK, UK (2000), http://dl.acm.org/citation.cfm?id=647373.724187

[6] Czarnecki, K., O'Donnell, J.T., Striegnitz, J., Taha, W.: Dsl implementation in metaocaml, template haskell, and c++. In: Domain-Specific Program Generation. pp. 51–72 (2003)

[7] Dagand, P.E., Baumann, A., Roscoe, T.: Filet-o-fish: practical and dependable domain-specific languages for os development. In: Proceedings of the Fifth Workshop on Programming Languages and Operating Systems. pp. 5:1–5:5. PLOS '09, ACM, New York, NY, USA (2009), http://doi.acm.org/10.1145/1745438.1745446

[8] Doty, D., Patitz, M.J.: A domain-specific language for programming in the tile assembly model. In: Proc. Fifteenth Int. Meeting on DNA Computing and Molecular Programming. pp. 8–11 (Mar 2009)

[9] Fowler, M.: Domain-Specific Languages. No. ISBN 0-321-71294-3, Addison Wesley (2011)

[10] de Geest, G.: Building a framework to support Domain-Specific Language Evolution using Microsoft DSL Tools. Master's thesis, Software Engineering Research Group, Delft University of Technology (2008)

[11] de Geest, G., Savelkoul, A., Alikoski, A.: Building a framework to support domain-specific language evolution using microsoft dsl tools. In: Proc. 7th OOPSLA Workshop on Domain-Specific Modeling (DSM'07) (2007)

[12] de Geest, G., Vermolen, S., van Deursen, A., Visser, E.: Generating version convertors for domain-specific languages. In: Proc. 15th Working Conf. on Reverse Engineering (2008)

[13] Ghosh, D.: Dsl for the uninitiated - domain-specific languages bridge the semantic gap in programming. Communications of the ACM 54(7), 44–50 (2011)

[14] Guenter, B., Nehab, D.: Neon: A domain-specific programming language for image processing. Microsoft Tech Report MSR-TR-2010-175, Microsoft Research (2010)

[15] Hawick, K.A.: Engineering domain-specific languages for computational simulations of complex systems. In: Proc. Int. Conf. on Software Engineering and Applications (SEA2011). pp. 222–229. No. CSTN-123, IASTED, Dallas, USA (14-16 December 2011)

[16] Hawick, K.A.: Engineering internal domain-specific language software for lattice-based simulations. In: Proc. Int. Conf. on Software Engineering and Applications. pp. 314–321. IASTED, Las Vegas, USA (12-14 November 2012)

[17] Hawick, K.A.: Fluent interfaces and domain-specific languages for graph generation and network analysis calculations. In: Proc. Int. Conf. on Software Engineering (SE'13). pp. 752–759. IASTED, Innsbruck, Austria (11-13 February 2013)

[18] Hemel, Z.: Methods and Techniques for the Design and Implementation of Domain-Specific Languages. Ph.D. thesis, Delft University of Technology (2012), iSBN 978-90-8570-794-3

[19] Husselmann, A.V., Hawick, K.A.: Automatic high performance structural optimisation for agent-based models. In: Proc. 14th Int. Conf. on Software Engineering Research and Practice (SERP'14). pp. 1–7. WorldComp, Las Vegas, USA (21-24 July 2014), http://www.hull.ac.uk/php/466990/csi/reports/0010/csi-0010.html

[20] Husselmann, A.: Data-Parallel Structural Optimisation in Agent-Based Modelling. Ph.D. thesis, Computer Science, Massey University, Albany, North Shore, New Zealand (May 2014)

[21] Johnson, P.: Gedcom 1.19 (August 2013), https://metacpan.org/pod/Gedcom

[22] Jones, T.: Ftw gedcom (March 2009), http://www.tamurajones.net/FTWGEDCOM.xhtml

[23] Karlsch, M.: model-driven framework for domain specific languages demonstrated on a test automation language. Master's thesis, Hasso-Platner-Institute of Software Systems Engineering, Potsdam, Germany (2007)

[24] Kelly, S., Tolvanen, J.P.: Domain-Specific Modeling: Enabling Full Code Generation. Wiley (2008)

[25] Kim, N.W., Card, S.K., Heer, J.: Tracing genealogical data with timenets. In: Proceedings of the International Conference on Advanced Visual Interfaces. pp. 241–248. ACM (2010)

[26] Ledford, H.: Genome hacker uncovers largest-ever family tree. Nature (October 2013), http://www.nature.com/news/genome-hacker-uncovers-largest-ever-family-\\tree-1.14037

[27] Lengauer, C., Batory, D., Consel, C., Odersky, M. (eds.): Domain-Specific Program Generation. No. 3016 in LNCS, Springer (2003), ISBN 3-540-22119-0

[28] Martin, J.: Fourth Generation Languages: Principles. Pretice Hall (1985)

[29] Sadilek, D.A.: Prototyping and simulating domain-specific languages for wireless sensor networks. Tech. rep., Humboldt-Universitat zu Berlin, Institute for Computer Science (2007)

[30] Simonyi, C.: The death of computer languages, the birth of intentional programming. Tech. rep., Microsoft Research (1995)

[31] Singer-Villalobos, F.: Computer scientists at ut austin crack code for redrawing bird family tree. Texas Advanced Computing Center (2014), https://www.tacc.utexas.edu/-/computer-scientists-at-ut-austin-crack-code\\-for-redrawing-bird-family-tree

[32] Skoric, N.: simplepyged (February 2014), https://github.com/dijxtra/simplepyged

[33] Sprinkle, J., Karsai, G.: A domain-specific visual language for domain model evolution. Journal of Visual Languages and Computing 15, 291–307 (2004)

[34] Taha, W.: Domain-specific languages. In: Pro. Int. Conf. Computer Engineering and Systems (ICCES). pp. xxiii – xxviii (25-27 November 2008)

[35] Veale, K.J.: A doctoral study of the use of internet for genealogy. Historia Actual Online 7 pp. 7–14 (2009)

[36] Voellmy, A., Agarwal, A., Hudak, P., an Sam Burnett, N.F., Launchbury, J.: Don?t configure the network, program it! domain-specific programming languages for network systems. Tech. Rep. YALEU/DCS/RR-1432, Yale University, USA (July 2010)

[37] Wetmore, T.: The lifelines programming subsystem and report generator (2005), http://lifelines.sourceforge.net/manual.3.0.39/ll-reportmanual.html

[38] Wilson, R.: Graph-based remerging of genealogical databases. In: Workshop on Technology for Family History and Genealogical Research. vol. 1 (2001), http://dagwood.cs.byu.edu/fht/workshop01/fht2001prog.php http://dagwood.cs.byu.edu/fht/workshop01/final/Wilson.pdf

[39] Zschaler, S., Kolovos, D.S., Drivalos, N., Paige, R.F., Rashid, A.: Domain-specific metamodelling languages for software language engineering. In: Proc. Software Language Engineering (SLE 2009). LNCS, vol. 5969, pp. 334–353 (2009)

Real-Time Requirements Engineering (RTRE) as Extended to Hard and Soft Real-Time Analysis

Ahmed Tarek[1], **and Ahmed Farhan**[2]
[1]Engineering, Physical and Computer Sciences, Montgomery College, Rockville, Maryland, United States of America
[2]College of Arts and Sciences, University of Pennsylvania, Philadelphia, Pennsylvania, United States of America

Abstract— *Real-Time Systems (RTSs) have more stringent temporal requirements as compared to classical software. Requirements analysis for RTS is more complex compared to traditional software. Flowcharts as purely sequential tools are not sufficient for concurrent real-time systems. Besides, the requirements analysis leading to real-time software design varies from system to system contingent upon software needs, which has progressed into the relatively new discipline of Real-Time Requirements Engineering (RTRE). RTRE uses additional tools and techniques besides the traditional ones, such as, Process Activation Table, Decision Table, Decision Matrix, and so in the Structured Analysis leading to robust and secured software design. In this paper, we address some of the essential RTRE tools and techniques, and their significance to real-time analysis as extended to Hard and Soft RTS. Hard and soft RTS are not required to be embedded software systems. However, time still plays a crucial role in their operation. This extended approach yields with better design, coding and software reliability.*

Keywords: Real-Time System, Real-Time Requirements Engineering, Structured Analysis, Hard Real-Time Systems, Soft Real-Time Systems, Embedded Real-Time Systems

1. Introduction

There exists a wide variety of tools and techniques for requirement analysis of strictly sequential systems. For instance, Jackson System Development (JSD) [4] covers the complete technical development of a wide class of systems that are strictly sequential. Concurrency imposes additional restrictions to system development [1]. A real-time system needs to respond to stimuli that occur at different points in time. Therefore, it is necessary to organize its architecture so that, as soon as a stimulus is received, control is transferred to the correct handler. Hence, the system architecture must allow for fast transfer to the appropriate handler. This is impractical in sequential programs. As a result, real-time systems are designed as a set of concurrent, cooperating processes, with a real-time executive controlling the processes [3]. Based on the sensitivity to temporal requirements, there are two categories - soft and hard real-time systems. With real-time systems, programŠs control flow is the most important part Ű as concurrency and control comes together and plays a vital role in the system operation. Therefore, there are two critical things to consider - the actions the program should take, and also the order of taking those actions. These two factors are related to Structure of the Code Ű how the code would look like. For classical software, simple, straight forward flowcharts are sufficient, where as for Real-time systems, programs are concurrent, and flowcharts are not enough as they are used to represent sequential control flow. This gave rise to the Real-Time Requirements Engineering (RTRE). The requirements engineering of real-time software deals with how requirements should be expressed, and also determines final formats of requirements specification.

In Section 2, specific terms and notations used in this paper are briefly deliberated. Section 3 explores RTRE. Section 4 considers the real time software design steps. Section 5 surveys the tools and techniques used for Real-Time Structured Analysis. Section 6 is the conclusion based on models of RTRE discussed in this paper, which also explores the future research avenues.

2. Terms and Notation

Following terms and notations are used all throughout this paper.

Soft Real-Time Systems: Time is of utmost importance but a little deviation from time bound does not lead to disastrous effects but only degrades the system performance. An example is an online banking system.

Hard Real-Time Systems: Time is the most critical factor, and a small deviation from time bound could be disastrous. An example is the real-time system used with a chemical processing plant.

Real-Time Requirements Engineering ($RTRE$): The requirements engineering for real-time software deals with how requirements for soft and hard real-time systems should be expressed. RTRE also determines final formats of requirements specification.

RTA: Real-Time Analysis.

PSPEC: Process Specification.

CSPEC: Control Specification.

DFD: Data Flow Diagram.

CFD: Control Flow Daiagram.

DD): Data Dictionary.
STD: State Transition Diagram.
PAT: Process Activation Table.
DT: Decision Table.
DM: Decision Matrix.
FSM: Finite State Machine.
FSA: Finite State Automata.
AID: Architecture Interconnect Diagram.
IP: Input Processing.
OP: Output Processing.
UIP: User Interface Processing.
MSTP: Maintenance and Self-Testing Processing.
CD: Context Diagram.
CCD: Control Context Diagram.

3. Real Time Requirements Engineering Explored

For real-time systems, program's control flow is the most important part, as concurrency and control come together and plays a vital role in system operation. This incorporates but is not limited to the the following:

1) **Actions of the Real-Time Software:** The actions that the real-time program is supposed to take, when certain event occurs. For example, shut down the chemical processing plant in times of an emergency.

2) **The Order of Taking Actions:** The order in which actions are required to be performed is of paramount importance for real-time concurrent systems. This is in order to avoid any mishap. For instance, boiler needs to be shut down before stopping the plant operation. User information is required to be saved before logging out of an online account.

All these determine the structure of the code, which is how the code is to be designed, and how the code would look like. Therefore, for classical software, simple, straight forward flowcharts are sufficient. With real-time software operating in concurrent mode, flowcharts are not sufficient, as flowcharts are used to represent Sequential Control Flow. This has advanced the Real-Time Requirements Engineering ($RTRE$) for real-time system design and operation. The requirements engineering of real-time software (RTS) deals with how requirements should be expressed for designing concurrent, real-time systems. $RTRE$ also determines the final formats of the requirements specification.

RTSs are almost always related to the surrounding environment with which the system interacts. As a result, for RTS, the requirements are grouped into categories as follows:

1) Input requirements relating to input signals or data.
2) Output requirements relating to output signals or data.
3) Joint I/O requirements. These are related to input and output data simultaneously.
4) Processing requirements relating to internal function of software.

In addition to the above functional requirements that relate to the functionality of real-time software, there is a group of other requirements relating to certain attributes of software such as, performance, safety, reliability, security, etc. The discussion in this research paper on RTRE will highlight and illustrate these diverse issues that come naturally with real-time system design and construction.

3.1 Advantages Rendered Through RTRE

Real-Time Requirements Engineering offers added tools and techniques for robust analysis that ensures safety, security and system reliability. Following are the advantages rendered through RTRE.

Advantages of RTRE:

For Hard and Soft Real-time Systems, with RTRE:

1) Flow control analysis for software becomes easier.
2) Yields better software architecture.
3) Less chance of errors in Real-time Software Design.
4) RTRE provides with a robust analysis leading into robust and reliable software design.
5) As the design is robust, there is less chance of changing the software design or requirements later resulting in reduced cost of development.
6) Eventually, this robust process results in better real-time software testing strategies.

From Software Analysis to Software Testing with RTS, itŠs a complete process known as the Structured Approach.

4. Real Time Software Design Steps

The Hatley & Pirbhai approach as articulated in [1] is one of the best approaches to real-time software design. The modified Hatley and Pirbhai approach starts designing the software with the Context Diagram, and follows through seven design steps.

1) Perform Data Flow Decomposition using Data Flow Diagrams (DFDs).
2) Perform Control Flow Decomposition through Control Flow Diagrams (CFDs).
3) Develop Process Specifications (PSPECs) in Structured English.
4) Develop Control Specifications (CSPECs) using State Transition Diagrams (STDs).
5) Develop Response Time Specifications (RTSs) for timing thresholds.
6) Develop Data Dictionary (DD).
7) Produce Architecture Interconnect Diagrams (AIDs).

The key to developing real-time software is understanding the relationship of software with the surrounding environment, with which, the software interacts. For instance, for Airline Ticket Reservation software, the real-time software interacts with the customers, who are trying to buy the

tickets. The principal ways of expressing these interactions are:

1) A Physical Diagram (PD), representing all physical entities, including a computer system or systems, if applicable. This leads to Physical Diagram.
2) A Context Diagram (CD) is a refinement of a Physical Diagram that represents the Real-Time Software as a single bubble (circle), and all external devices as rectangles. There are arcs connecting the circle, and the rectangles representing connections, data flow, and also the directions of data flow.

The key elements in the design with this approach are based on the following:

1) Deriving DFDs and CFDs using the template of the architecture.
2) Relies on the main characteristics of the system as captured by DFDs and CFDs.
3) But the design refines DFDs and CFDs into a physical representation.

The real-time software design template is based on 5 components.

1) Functional and Control Processing (FCP), which constitutes the Central Box of the design template. This part is based on Process Model (PM) and Control Model (CM) of the real-time software.
2) Input Processing (IP), constituting the left of central box.
3) Output Processing(OP), which makes up the right of central box.
4) User Interface Processing (UIP), shown as the top of central box.
5) Maintenance and Self-Testing Processing (MSTP), represented as the bottom of central box.

Following figure is representative of these five different components and their relative positioning in the design template.

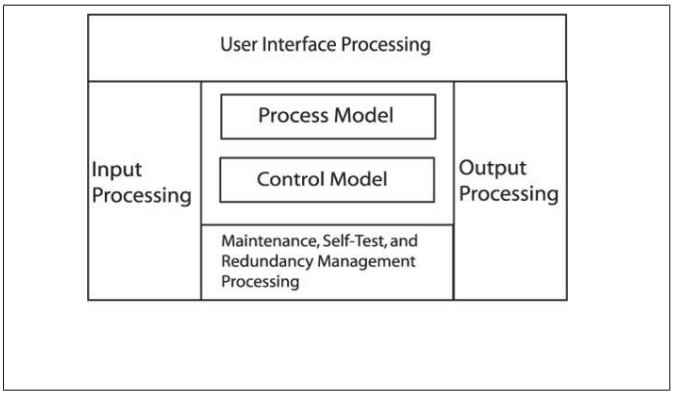


Fig. 1: Software Design Template.

Whenever it comes to the actual system implementation with the real-time embedded software, following notations and symbols play major roles. Following figure is representative of different notations used during this phase. Following

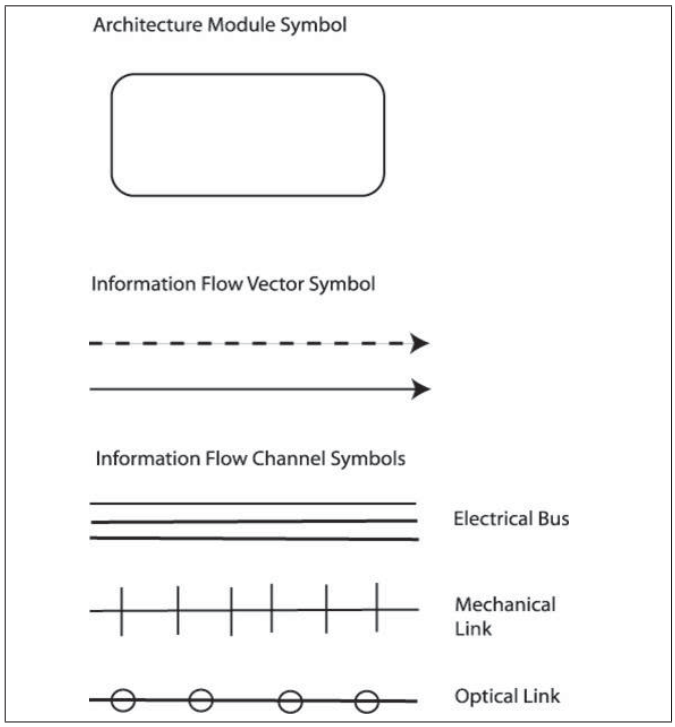


Fig. 2: System Implementation Symbols.

represents the ideal scenario in real-time embedded system design and development.

1) Software Engineers meet with the Client to collect and gather the System Requirements(SRs).
2) Based on the collected SRs, Real-Time Requirement Analysis (RTRA) takes place.
3) Once RTRA is complete. Real-time Software Design Engineers pursue the System Design.
4) Once the Real-Time Software Design (RTSD) becomes available, both the Hardware Development (HD), and the Software Development (SD) phases progress in parallel.

   Following are the typical sub-phases during HD phase.

   a) Hardware Requirements Analysis (HRA)
   b) Preliminary Hardware Design (PHD)
   c) Detailed Hardware Design (DHD)
   d) Hardware Fabrication (HF)
   e) Hardware Testing (HT)

   Following are the standard sub-phases during real-time embedded software development.

   a) Real-Time Requirements Analysis (RTRA)
   b) Preliminary Software Design (PSD)
   c) Detailed Software Design (DSD)
   d) Real-Time Software Coding and Unit Testing (CUT)
   e) Real-Time Software Integration Testing (IT)

5) Once both the hardware and software becomes available, System Integration (SI) is performed.
6) Once SI is complete, System Integration Testing (SIT) and System Testing (ST) are done to ensure safety, security and reliability.

5. Tools and Techniques for Real-Time Structured Analysis

Whereas the Classical Software Engineering (CSE) typically uses Data Flow Diagrams (DFDs), the Real-Time Requirements Engineering (RTRE) includes, but are not limited to the following tools and techniques. This is due to the nature and criticality of the real-time system design, implementation and physical operation.

1) Data Flow Diagrams (DFDs)
2) Control Flow Diagrams (CFDs)
3) Decision Tables (DTs)
4) Decision Matrices (DMs)
5) Process Activation Tables (PATs)
6) State Transition Diagrams (STDs)
7) State Transition Tables (STTs)

This paper mostly focuses on the current tools and techniques pertaining to RTSE. Using Structured Approach for RTA yields with added benefits. Some examples of real-time systems include Vending Machine, On-line Banking web sites, etc. These are soft real-time systems. With hard real-time systems, timing factor is crucial, such as the chemical processing plants.

The Structured Analysis Model (SAM) uses the following core concept: Following constitutes the basic components of

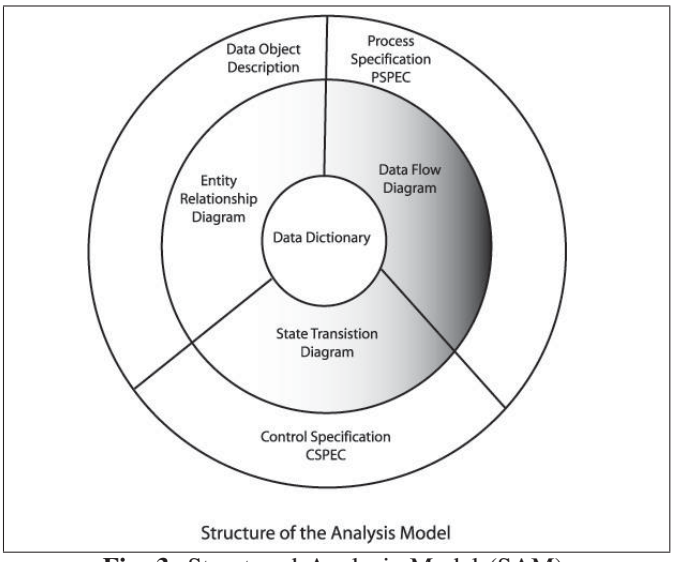


Fig. 3: Structured Analysis Model (SAM).

SAM.

1) Data Dictionary (DD): DD represents core of the model. DD is a repository containing a description of all data objects.ă
2) Entity Relationship Diagram (ERD): ERD depicts relationships between data objects formulating data attributes.
3) Data Flow Diagram (DFD): DFD serves two different purposes.
   a) DFD describes how data is transformed as it moves along the system.
   b) DFD depicts the functions for transforming the data flow.
4) Process Specification (PSPEC): The description of each transform function for data flow is contained in PSPEC.
5) State Transition Diagram (STD): STD describes how the system responses to external events.
6) Control Specification (CSPEC): CSPEC contains description on the control aspects of software.

5.1 Real-Time Analysis (RTA)

Following are important segments of Real-Time Analysis (RTA).

1) Data Flow Analysis (DFA): Done through Process Specification (PSEC).
2) State Transition Analysis (STA): Performed through Control Specification (CSPEC).
3) Data Dictionary (DD): A repository of data items useful for DFA and STA.

Following implies how the Analysis Model (AM) maps into a **hierarchy** referred to as the Design Model (DM).

1) Component Level Design (CLD): CLD constitutes the Highest Level (HL) in the entire hierarchy. CLD is derived from Process Specification (PSPEC), State Transition Diagram (STD), and Control Specification (CSPEC).
2) Interface Design (ID): ID is derived from State Transition Diagram (STD), and Control Specification (CSPEC), as well as the Data Flow Diagram (DFD).
3) Architectural Design (AD): AD is the next lower level in the hierarchy. It uses the Data Flow Diagram (DFD).
4) Data Design (DD): DD is the Lowest Level (LL) in the entire hierarchy. It is derived from Data Dictionary (DD), and the Entity-Relationship Diagram (ERD).

5.2 On Tools and Techniques for RTRE)

5.2.1 Data Flow Diagram (DFD)

Shows how data is transformed as it moves along the real-time system. DFD is a graphical aid for RTA. DFD uses following symbols. DFDs start with the Context Diagram (CD), which represents the entire system as a big bubble and proceeds through the hierarchical decomposition. Following illustrates the principle of hierarchical decomposition of designs.

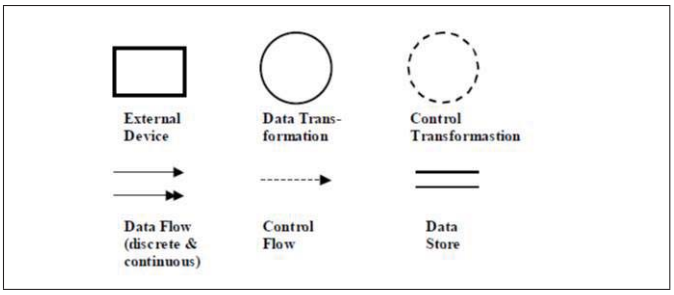


Fig. 4: Symbols Used With DFDs.

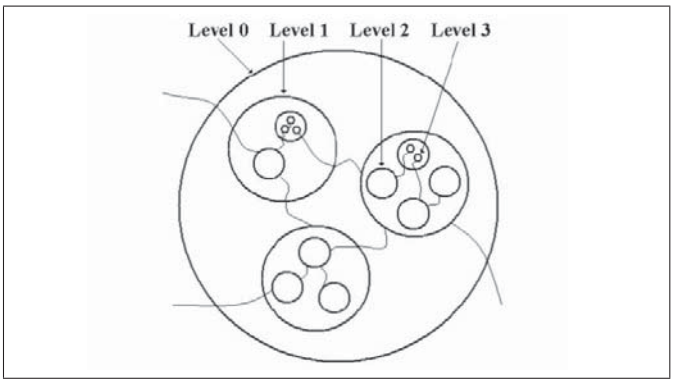


Fig. 5: Hierarchical Decomposition With DFD.

5.3 Illustration Of Data Decomposition:

To better illustrate the process of data decomposition, consider the example of a vehicle Cruise Control System (CCS). Following represents the Context Diagram (CD) of the CCS. As evident from the figure, the CCS interacts with

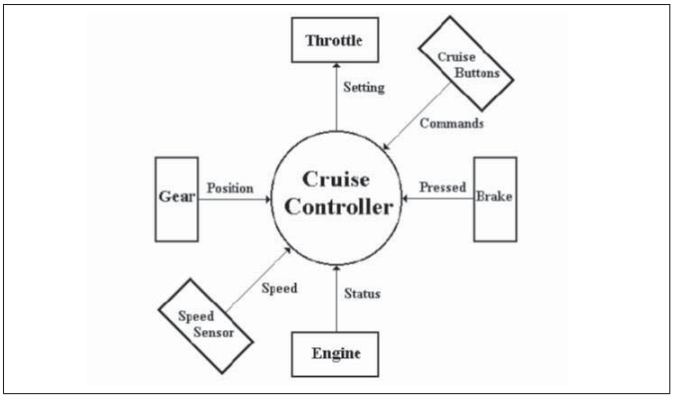


Fig. 6: CD of A Vehicle's CCS.

its surrounding environment, which in this case, consist of, throttle, cruise buttons, brake, engine, speed sensor, and the gear assembly.

Next for the control purposes, the Cruise Controller Process is decomposed into lower level functions as shown in the following. As evident from the CCP decomposition, there are three main functions of the cruise controller process, which are:

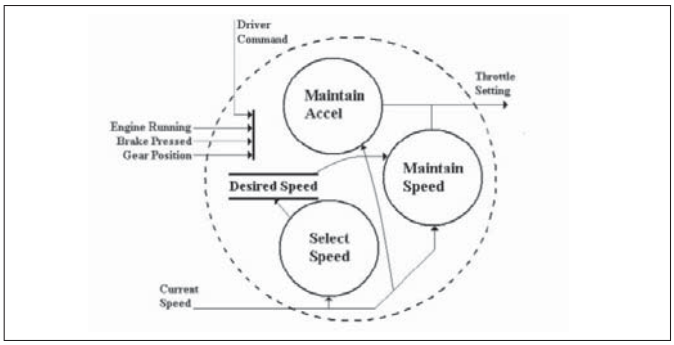


Fig. 7: Decomposition of the Cruise Controller Process.

1) Select Speed: Select a particular speed using the cruise controller button.
2) Maintain Speed: Maintain the pre-set vehicle speed selected through the cruise controller button.
3) Maintain Acceleration: Maintain acceleration to preserve the vehicle speed to the pre-set level by controlling throttle (fuel injection).

Since all three of these functionalities are related to control, the entire cruise controller in which, these three control functions are nested is shown with a broken bubble, which represents a Control Context Diagram (CCD). The PSPEC for the above three control functions follow: PSPEC rep-

Fig. 8: PSPEC for Cruise Controller.

resents the textual description of control logic related to each one of the three control functionalities. The change of state in the control application is best represented by a State Transition Diagram (STD). Following represents STD of the Cruise Controller. In context to STDs, Finite State Machines (FSMs) play a vital role. Following subsection discusses FSM in relation to real-time systems.

5.4 Finite State Machine (FSM) in RTRE

Finite State Machine (FSM) plays a major role in RTRE. Real-time systems have the property of past and present events. Also, both external and internal events can change

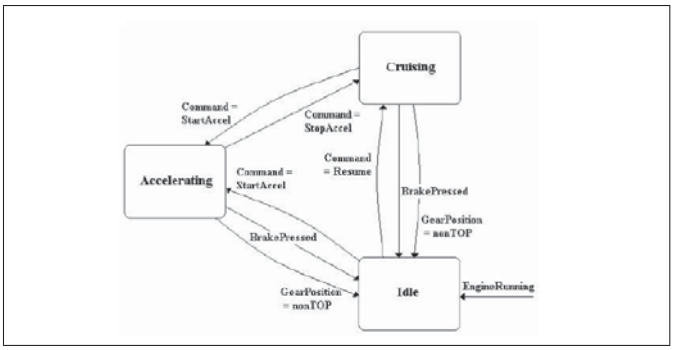


Fig. 9: STD of the Cruise Controller.

real-time behavior. Real-time systems respond to changes in their input, and also to events that occur over time with system appearing as if several different processors from one instance of time to another. The process model alone fails to express this type of action. Therefore, FSM combined with the process model is suited to modeling the change in state behavior due to internal or external events.

Continuous Machines (CMs) or Analog Machines (AMs) are capable of processing continuous-valued inputs, outputs, and internal elements. CMs are also able to receive and produce discrete-signal values. PSPECs represent continuous machines in the requirements model.

On the contrary, FSMs can only process discrete signals. Control specifications (CSPECs) are represented by FSMs. There are two categories of FSMs.

1) Combinational Machine (CM) Ű Combinational Machines have no memory. As a result, CM can not hold or represent the past states. It is capable of representing only the present states.
2) Sequential Machine (SM) Ű SM has Memory. As a result, SM is capable of representing both the past and the present states. SM requires a wide a variety of tools to show the state transitions. These include, but are not limited to STD, DT, DM, PAT, etc.

5.5 Decision Table (DT) and Decision Matix (DM) in RTRE

DTs and DMs are often used in RTRE for structured analysis. An example follows:

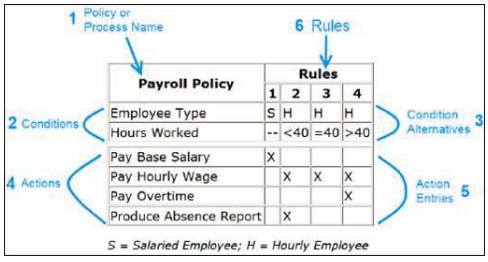


Fig. 10: DT for Employees.

The Decision Table (DT) for organizational employees helps in making decisions regarding salaries. DTs are two dimensional and composed of rows and columns. Each of the columns defines the conditions and actions of the rules. Decision tables represent a list of causes (conditions) and effects (actions) in a matrix, where each column represents a unique rule. The purpose is to structure the logic for a specific specification feature. It is possible to add new rows to a decision table and fill in its cells to create new rules. When the rules are executed, if the conditions are met, the actions in that column are performed. One can also add preconditions that apply to all the rules.

5.6 Process Activation Table (PAT) in RTRE

Decision Tables (DTs) that are used to activate processes are called Process Activation Tables (PATs). Following represents a CSPEC containing both PAT and DT.

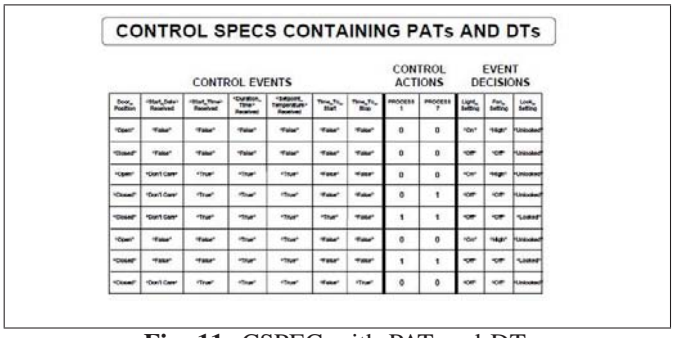


Fig. 11: CSPEC with PAT and DT.

5.7 STDs & DTs In RTRE

Following figure illustrates CFD/DFD (left diagram) and transitions among states due to the event triggering (right diagram).

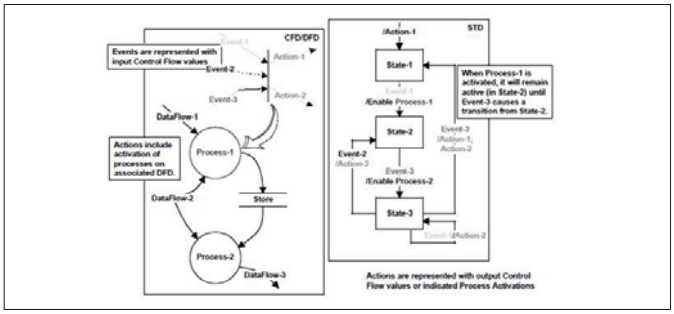


Fig. 12: STD in RTRE.

Fig. 13 is representative of CFD/DFD (left diagram) and the decisions taken due to different combinations of the occurring events (right diagram).

Sometimes DFD and CFD are combined in CSPEC to represent CSPEC as the STD. Fig. 14 is representative of this scenario.

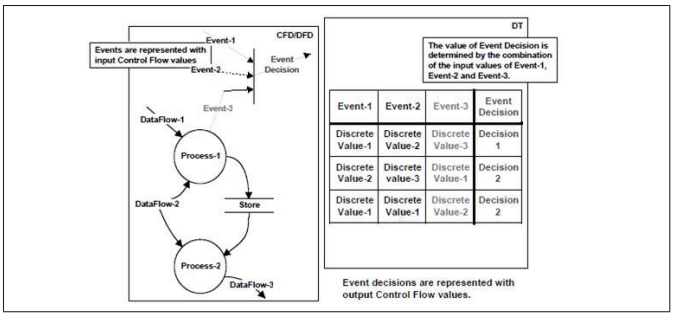


Fig. 13: DT in RTRE.

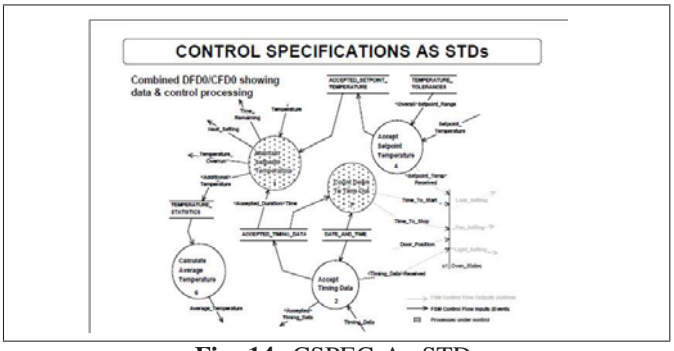


Fig. 14: CSPEC As STD.

6. Conclusion

State Transition Diagram is only suited for a limited number of states. Alternative representations used with large number of states are State Transition Matrix (STM) and State Transition Table (STT). Control Specifications (CSPECs) consists of FSMs, which map control inputs to control outputs. CSPEC combines both Combinational and Sequential Machines. Block diagram in Fig. 15 is representative of this scenario. A general real-time system model involves associ-

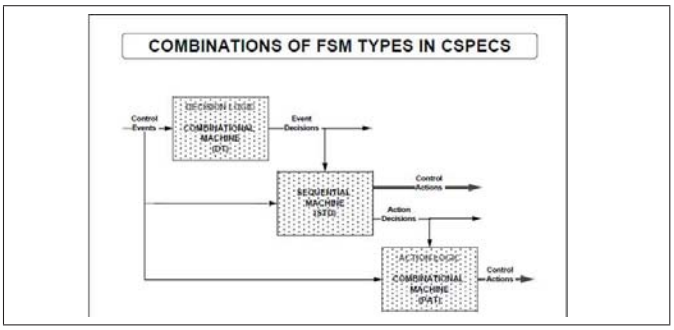


Fig. 15: CSPEC with FSMs.

ating processes with sensors for data input, and actuators for data output, which are sent as control signals. Real-time System architecture is usually designed as a number of concurrent processes. Real-time system specification involves System Requirement (SR), representing the Logical Configuration of software, and System Architecture (SA), representing the Physical Configuration of software. Both SR and SA are required to be developed in parallel.

After incorporating different components in the Software Design Template (SDT) for Cruise Controller, following Fig. 16 is the complete software architecture.

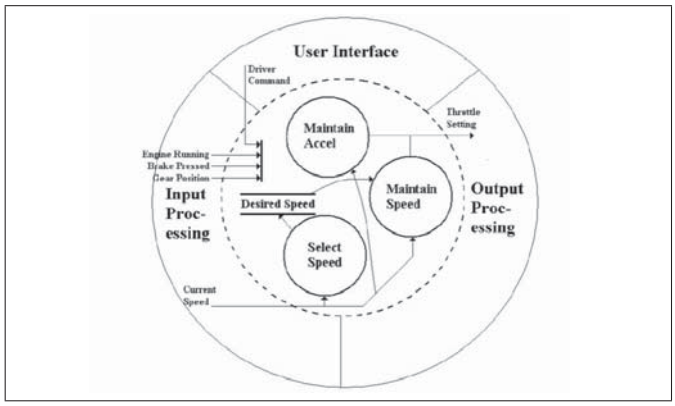


Fig. 16: Cruise Controller Software Architecture.

System functionalities are either partitioned to hardware or to software. Design decisions are made based on System Requirements. Hardware implementation delivers better timing performance compared to the software counter-part. However, functional implementation with Hardware incurs higher costs, potentially longer development time, and less flexibility for accommodating any future changes.

Due to functional requirements, sometimes RTS may not be implemented using the Object-Oriented Approach. Real-Time Requirements Engineering (RTRE) is wide. There are still unexplored avenues in this engineering domain. We tried to represent a few of the useful tools and techniques from the RTRE paradigm in connection to modern real-time systems. In future, we plan to incorporate RTRE, and the Structured Approach discussed in this paper for a realistic Patient Monitoring System (PMS) at hospitals, and also to an Online Registration Control System (ORCS) for a self-sustained small academic institution.

References

[1] Derek J. Hatley and Imtiaz A. Pirbhai, *Strategies for Real-Time System Specification*, New York, United States of America: Dorset House Publishing, 1987.

[2] Derek J. Hatley, Peter Hruschka and Imtiaz A. Pirbhai, *Process for System Architecture and Requirements Engineering*, New York, United States of America: Dorset House Publishing, 2000.

[3] Ian Sommerville, *Software Engineering*, 8th ed. New Jersey, United States of America: Pearson Education, 2007.

[4] John R. Cameron, *JSP & JSD: The Jackson Approach to Software Development*, Maryland, United States of America: IEEE Computer Society Press, 1983.

[5] Roger S. Pressman, *Software Engineering - A Practitioner's Approach*, 5th ed. New York, United States of America: McGraw-Hill Higher Education, 2001.

[6] Sebastian Altmeyer, Roeland Douma, Will Lunniss, and Robert I. Davis, "On the effectiveness of cache partitioning in hard real-time systems," *Real-Time Systems*, vol. 52, pp. 598–643, Jan. 2016.

Which Roles Ontologies play on Software Requirements Engineering? A Systematic Review

J. Valaski, S. Reinehr, and A. Malucelli

joselaine.valaski@pucpr.br,sheila.reinehr@pucpr.br, malu@ppgia.pucpr.br

PPGIa, Pontifícia Universidade Católica do Paraná, Curitiba, Paraná, Brazil

Abstract - *One of the main goals of Software Requirements Engineering is to understand customers' needs. However, some of the main flaws of Requirements Engineering are the lack of a common vocabulary, communications difficulty among stakeholders and the lack of domain comprehension of the problem domain in order to solve it. Ontologies have played a highly versatile role in the solution of these issues. The aim of this paper is to present a systematic review, identifying which roles ontologies play are on Software Requirements Engineering. This systematic review has identified 2407 distinct papers. Using the application of exclusion criteria, 60 papers have remained and were analyzed.*

Keywords: Ontology, Requirements Engineering, Software Requirements Engineering, Systematic Review.

1. INTRODUCTION

The Software Requirements Engineering (SRE) provides the appropriate mechanisms to understand customers´ needs [1]. SRE can be defined as an iterative process of discovery and analysis with the purpose of producing a clear, complete and consistent set of requirements for software [2][3]. The initial result of the SRE process consists of several not very clear views whereas the final result consists of a complete specification of the system, formally represented [4].

In the early stages of SRE, the system to be developed is often inaccurate and inconsistent [5]. The lack of understanding of the business by the requirements´ engineers and the communication breakdowns among the specialists in the business and in computing, compromise the quality of information [6]. Furthermore, the lack of consensus regarding the use of terms in the organization can lead to different meanings [7]. Considering these matters, some of the main needs during the development of software are: the use of a common language, aiding the comprehension of information obtained from different sources [8]; a broad understanding of the domain; a conceptual model that uses a common vocabulary to facilitate the discussion; and the construction of a sufficiently clear and unambiguous specification. Ontologies have been an important resource to satisfy these and other needs that result from problems related to SRE.

The application of ontologies in SRE can be motivated by different objectives: common shared vocabulary [9]; reuse and structuring of knowledge [10]; understanding the problem domain [11]; analysis of the expressivity of the language [12]; the closest possible representation of the problem to the real world [13]; and better communication among specialists from different domains [14]. Ontologies play an important and diverse role when it comes to handling problems in SRE. The aim of this study is to understand in a broader view what the roles of ontologies are on the SRE. It can be found in the literature some studies about the application of ontologies on SRE [15][16]. However, this review has found distinct proposals and answers that increase the understanding of the use of ontologies on SRE.

The remainder of this paper is organized as follows. Section 2 presents the main concepts discussed in this work. Section 3 shows the steps taken during the systematic review. Section 4 presents and discusses the results. Section 5 brings the paper to a close.

2. THEORICAL BACKGROUND

In this section, the definitions and objectives of the two main concepts involved in this systematic review are discussed: SRE and Ontology.

2.1. Software Requirements Engineering

The development process of a requirements specification is known as SRE. However, there is no single definition of SRE. Pohl [17] discusses the SRE in four main tasks: Elicitation, Negotiation, Specification & Documentation and Verification & Validation of requirements. The SWEBOK [18], the guide to a Software Engineering Body of Knowledge, discusses the SRE in knowledge areas and presents the following topics related to the SRE area: Software Requirements Fundamentals, Requirements Process, Requirements Elicitation, Requirements Analysis, Requirements Specification, Requirements Validation, Practical Considerations. On the other hand, Kotonya and Sommerville [19] state that most Requirements Engineering process can be described by the activities: Requirements Elicitation, Requirements Analysis and Negotiation, Requirements Documentation and Requirements Validation.

As it is a process that occurs during the initial phases of software development, communication among those involved are informal and the use of natural language is common. Therefore, some of the recurring problems in this process are: the lack of a commonly used vocabulary, the lack of understanding of the problem domain and the difficulty in communication among those involved. In this context,

ontologies are believed to play a fundamental role in the solution of such problems.

2.2. Ontology

In the field of computing, one of the most known definitions is that of Gruber [20]: "An ontology is a formal specification of a conceptualization". The definition of ontology can also be based on the complexity of its structure [21]. Ontologies are used to enable the reuse of domains of knowledge, to make explicit assumptions of a domain and separate the knowledge domain from the operational domain [22].

There is a wide range of uses for ontologies, from their conceptual to an approach for implementation in computers. Considering these aspects, Guizzardi [23] suggests the need for two classes of language for representation in ontology engineering. The first class, referred to in the present study as "conceptual" has to do with philosophically well-founded languages that focus on expressivity and conceptual clarity. The second one, referred to in this study as "computational" languages, focus on computational-oriented concerns such as decidability and automated reasoning.

Regardless of the type of ontology, its application has become a trend in several fields and applications. That being the case, it is considered important to obtain a more in-depth and wider view which roles ontologies play on the SRE. To obtain this view, a systematic review was conducted.

3. METHOD

The research method applied was the systematic review of the literature (SLR). The main goals of a SLR are to identify, evaluate and interpret the available and relevant studies on a single, particular question [24]. This review was conducted in the following stages.

3.1. Planning the review

In this stage, the protocol was specified, with the processes and methods for the application of the systematic literature review. According to Kitchenham [24] one of the reasons to perform a SLR is to summarize the existing evidences concerning a treatment or technology. Considering the versatility of ontologies in the solution of problems regarding to SRE, there is a need for a SLR to identify and understand which roles ontologies play on the SRE. To answer this main question, these research questions have been established: **RQ1:** What are the SRE activities in which ontologies are being applied on?; **RQ2:** Which are the functions of ontologies on the SRE?; **RQ3:** What are the languages being used by ontologies on the SRE?; **RQ4:** What knowledge domains are being represented by ontologies? and **RQ5:** What are the contributions of ontologies to SRE?

3.2. Research identification

The main goal of a systematic review is to find the highest number of primary studies related to the research questions. For this purpose, keywords were used to identify as many relevant works as possible. The final string used for the searches was: *(Ontologies OR Ontology) AND ("Requirements Development" OR "Requirements Engineering" OR "Requirements Analysis" OR "Requirements Definition" OR "Requirements Modeling" OR "Requirements Elicitation" OR "Requirements Inspection" OR "Requirements Management" OR "Requirements Negotiation" OR "Requirements Process" OR "Requirements Specification" OR "Requirements Traceability" OR "Requirements Validation" OR "Requirements Verification" OR "Requirements Reuse" OR "Software Requirements" OR "Quality Requirements" OR "Non-functional Requirements" OR "Functional Requirements" OR "Late Requirements" OR "Early Requirements")*.

The searches were conducted in the following scientific databases: Science Direct, Springer Link, IEEE and ACM Digital, where publications were filtered according to the type of journal and conference. The study was conducted in January 2013 and there was no limitation on the time of publications. In addition to the conferences indexed in the aforementioned digital bases, the Workshop on Requirements Engineering (WER) was also considered. As a result, 2407 papers formed the base for the selection of primary studies.

3.3. Primary studies selection

The process to select the primary studies followed four steps. In Step 1 the set of keywords on the defined database was applied. This returned 2.419 papers. After removing 12 duplicate titles, 2.407 distinct papers remained. Exclusion criteria were defined in the protocol for suitably selecting the studies in next three steps. In Step 2, the abstracts of all 2.407 papers were read to eliminate those that clearly did not discuss ontology. Papers were excluded only when there was no doubt that the theme was not discussed. Most of the papers (2.148) were excluded in this second step. In Step 3, with 259 papers remaining, each paper was searched for the keyword "ontology" or "ontologies", seeking evidence of its application to a theme of SRE. This strategy was used because the term ontology would be more restrictive than all the other terms used for SRE. In this step, more 123 papers were excluded.

Finally, in Step 4, the 136 remaining papers were fully read to confirm the application of the ontology concept in the area of SRE. During this step, 76 papers were excluded. With the three steps of exclusion concluded, 60 papers remained.

3.4. Classification

After the papers had been read and selected, they were classified according to the following categories: SRE activity [17-19): Elicitation, Analysis, Specification, Validation, Negotiation, Management, and, Activity undefined; Ontology function; Language class [23]: Conceptual and Computational; Knowledge domain: SRE domain, Problem domain, and Other; Evaluation type [25]: Empirical, Not Empirical, and Not applied; Empirical Evaluation Context: Academic and Industry; and Ontology contribution.

4. RESULTS AND DISCUSSION

The results will be discussed and presented according to the research questions presented in Section 3.1.

RQ1: What are the SRE activities in which ontologies are being applied on?

The objective of this question was to identify in which activities of SRE ontologies are being applied. Table 1 presents the summarized results related to this question.

Table 1 Ontologies application in SRE activities

SRE activity	ID	Qty.	%
Analysis	[5][8][14][26-53]	31	51,66
Elicitation	[6][15][28][54-64]	14	23,33
Specification	[31][56][61][65-72]	11	18,33
Management	[73-77]	5	8,33
Validation	-	0	0,00
Negotiation	-	0	0,00
Activity undefined	[78-80]	3	5,00

According to the obtained results, most of the analyzed proposals (51,66%) have been applied in the analysis activity. It is possible to observe the expressive application of ontologies specifically to modeling activity, such as: evaluate the quality or expressivity of the modeling language [14][44-48][51][53]; integration/transformation of models [28][32-33][37][41][49-50]; detect conflicts or inconsistencies in models [30][39-40][42]; validate the generated diagrams [27]; represent design rationale during the requirements modeling [38]; and, modeling of the system to be developed [8]. In the analysis activity, proposed ontologies have also been identified in order to: provide the reuse of attributes [31][34][36]; define the grammar of the language for requirements representation [5][52]; improve the vocabulary and the meaning of models elements [35][43]; and, identify variability in user solicitations [26].

The elicitation activity was the second most proposed one (23,33%). In this activity, the ontologies have been applied in a general fashion in order to: support domain understanding [6][15][54-55][60-62]; improve communications among stakeholders [15][55][63]; extract an initial list of requirements [56][58-59][64]; reuse knowledge [57][62]; and, identify the organizational objectives [28]. The third activity (18,33%) with most proposals was the requirements specification. Ontologies have been applied in order to: verify inconsistencies from the requirements document [56][65-66][69][71]; facilitate integration between specification and others activities of requirements [71-72]; improve how sentences are written [61][68]; reuse attributes requirement specification [67][31]; and structure the quality attributes [70]. The Management activity was the fourth one with the most amount in related proposals (8,33%). Ontologies have been applied in order to: allow traceability among the produced artifacts[73][75-76]; verify the risks associated with security requirement [74]; and, search for requirements artifacts [77]. Proposals have not been found to the Validation and Negotiation activities.

However, proposals not related to a specific activity have been identified (5%). In this classification, ontologies have been applied in order to: represent non-functional requirements patterns [78]; clarifying the meaning of process transparency [79]; and, structuring forms of requirements representation [80]. Most of the analyzed proposals have been classified in a single activity, except the proposals [28][31][56][61].

RQ2: Which are the functions of ontologies on the SRE?

A formal classification has not been used. The categories emerged from the text, after an interactive annotation process of the functions highlighted by the authors. Functions were put together and seven generic functions were obtained. A single proposal presented more than one ontology function. The results are presented in Table 2.

Table 2. Ontology function on the SRE

Ontology function	ID	Qty.	%
Structuring and recovery of knowledge	[26][29][32][34][38][63-64][66-70][75-78][80]	17	28,33
Verification and validation	[8][27][30][33][37][39-40][42][56][65-66][68-69][71][74]	15	25,00
Support to understand/identify concepts	[6][15][32-33][43][46-47][53][60-64][70][79]	15	25,00
Control/share of vocabulary	[5][8][32][35][52][54-55][58-59][61-62][65][68][76]	14	23,33
Integration and transformation models	[8][28][31-33][37][41][49-50][66][71-73][75]	14	23,33
Evaluation of representation language	[5][14][44-48][51-53]	10	16,66
Reuse	[8][26][31][34][36-37][57][62][67][78]	10	16,66

According to Table 2, the most highlighted functionality is structuring and recovery of knowledge (28,33), whereas the least identified ones were evaluation of representation language (16,66%) and reuse (16,66%). However, it is possible to observe that the percentage among identified functionalities is well distributed.

RQ3: What are the languages being used by ontologies on the SRE?

The proposals were put together according to the language class [23] and the language used on the ontology representation. In the conceptual language class, some proposals have not used any specific language. It was possible to identify the ontological model used. Table 3 presents the summarized results related to this question.

The computational language class was the most applied to represent ontologies (55%). The conceptual language class was applied in 41,67% of the proposals. Two proposals (3,33%) have been found to be applied in both classes.

In the proposals that applied the computational language class, the OWL language was the most used one. The inference mechanisms are one of the main resources used in this class. These mechanisms are mainly applied in order to provide interoperability, reuse and automatic inconsistency identification. Some examples that use the computational language class are as follows: enable interoperability and

validation of the models generated [8]; eliminate ambiguities in the document [61]; provide the reuse of attributes of quality in quality requirement specification [31]; help stakeholders in the construction of a complete, consistent, organized, traceable and unambiguous specification [71]; extract knowledge to be reused in the business modeling process [36]; enable the transformation of conceptual models into logical models [41]; enable traceability among the produced artifacts [75]; and, promote a common structure to facilitate the search for requirements artifacts [77].

Table 3 Languages used by ontologies

Language class/LanguageModel	ID	Qty.	%
Computational		33	55,00
OWL	[8][29][31][36][38-	13	
RDF/OWL	[27][34]	2	
RDF	[32][59]	2	
Alchoin	[42]	1	
Not informed	[30][37][56-57][65][67][70][73]	8	
FCA (Formal Concept Analysis) [74], FOL (First-order Logic)[76], Predicate logic[55], Temporal logic[66], Prolog[6], SIN (Description Logic)[72], XML[68]		7	
Conceptual		25	41,67
Bunge Model	[43][47-49][51][53]	6	
BWW Ontology	[14][44-45]	3	
UML	[28][62]	2	
Not informed	[5][15][26][35][50][54	10	
AORML[46], DOLCE[52], i*[64], StarUML[78]		4	
Hybrid		2	3,33
UML/OWL[33], i*; KAOS/Frame[69]		2	

In the proposals that applied the conceptual language class, the Bunge Model and the BWW ontology were the most used ones. In these proposals, ontologies are mainly applied with the goal of evaluating the quality and language expressivity and the models used for the artifact representation produced in the RE process. The BWW ontology has been used, for instance in order to: evaluate the quality of the AIML [14]; point out improvements in the information systems modeling, identifying deficiencies in the language [44]; and, to evaluate the modeling techniques and how to compare them [45]. The Bunge Model on the other hand has been applied in order to: suggest accurate meanings for the elements of the conceptual model and define formal semantics for the conceptual modeling language [43]; evaluate the syntax of a modeling language [47]; identify deficiencies in the grammar used in the entity-relationship diagrams [48]; and, evaluate existing process models [51].

RQ4: What knowledge domains are being represented by ontologies?

In order to answer this question, proposals were classified according to the knowledge domain represented by ontology. In the SRE process, ontologies may represent knowledge related to: SRE domain (SRE resources, methods, tools, models etc.), problem domain and other knowledge. Table 4 presents the summarized results relating to knowledge domain.

Table 4 Knowledge domain represented by ontologies

Domain	ID	Qty.	%
SRE	[5][8][14][30-31][34-53][64][66-80]	41	68,33
Problem	[6][15][26-29][32-33][54-60][62-63][65]	18	30,00
Other	[61]	1	1,67

Most of the analyzed proposals (68,33%) use ontologies in order to represent resources produced during the SRE process. In order to present more details of the classified proposals in the SRE knowledge domain, the resources represented by ontologies were also identified. The result is presented in Table 5.

Table 5 SRE domain knowledge represented by ontologies

SRE resource/ Sub-type	ID	Qty.	%
Model		21	51,22
Business model	[36][42][44][49][51]	5	
General	[37][41][47][50]	4	
Conceptual model	[14][43][46]	3	
Activity diagram	[30][45]	2	
Use case	[40][68]	2	
Class diagram[53], Collaborative[8], Entity-relationship diagram[48], Pattern[38], Product family[34]		5	
RNF		11	26,83
Quality	[31][39][70]	3	
Accessibility[67], Confidentiality[69], General[66], Pattern[78], Security[74], Transparency[79], Trust[35], Usability[64]		8	
General artifact	[73][75-77]	4	9,75
SRE core concept	[5][52][72][80]	4	9,75
Document	[71]	1	2,44

In general terms, the models (51,22%) are the most represented resources by ontologies. One of the justifications for an increase of the ontology application in models is that ontologies and models have the common goal of a formal representation of knowledge. Some examples of proposals that have applied ontologies for the modeling representation are: model for collaborative processes to represent concepts in business modeling [36]; ontology for MOBMAS, a methodology for the development of agent-oriented systems, using several models to be developed from the analysis of an agent-oriented system to the design and architecture of the system [37]; ontology to formalize the keywords of conceptual models [41], ontologies were proposed to represent exceptional flows in business process models [42]; use of ontological concepts to propose the AORML (Agent-Object Relationship Modeling Language) [46]; and, ontology to identify the components of a business model [49].

There are also an expressive number of proposals (26,83%) for the representation of non-functional requirements. Ontologies are also used in order to represent general artifacts (9,75%), with the goal of allowing artifacts tracking and searching for the artifacts. There are also ontologies in order to represent the core concepts of the SRE (9,75%) and just one ontology in order to represent

requirement specification document knowledge (2,44%). However, according to Table 4, it is also observed the relevant use of ontologies to the representation of a problem domain (30%). The problem domain representation is applied mainly in the initial phases of the SRE (e.g. elicitation). The problem domain normally describes the domain to which the software will be developed.

RQ5: What are the contributions of ontologies to SRE?

Firstly, the proposals were classified according to the type of evaluation, as can be seen in Table 6.

Table 6 Types of evaluation related to ontologies application

Type	ID	Qty.	%
Not empirical	[5][8][26][28-30][32][34-36][38][43-45][47][50-52][54-55] [57][59][62][64][66-68][71-72][74][77-79]	33	55,00
Empirical	[6][14-15][31][33][37][39-42][48-49][53][56][58][60-61] [63][65][69-70][75-76]	23	38,33
Not	[27][46][73][80]	4	6,67

Three general evaluation groups were identified: Not empirical (55,00%), Empirical (38,33%) and Not applicable (6,67%), when no type of evaluation was presented. This result indicates that the SRE area still demands more empirical evaluation works. Among the empirical evaluations the ontology application context was identified. According to Table 7, most evaluations (78,26%) were applied in the academic context, whereas a small percentage (21,74%) was applied in the industry context.

Table 7 Empirical evaluation context related to ontology application

Context	ID	Qty.	%
Academy	[6][14][33][39-42][48][53][56][58][60-61][63] [69][70][75-76]	18	78,26
Industry	[15][31][37][49][65]	5	21,74

To each one of the empirical evaluation proposals, the ontology contribution to the SRE area was identified. The most important contributions and how they are related to the evaluations performed are discussed as follows: 1) Identifying problems in specification and models: the contributions were reported as follows: detection of missing information in scenarios [65]; improvement in incomplete and ambiguous specifications [69]; identification of requirements inconsistency [33]; identification of entities in specifications [33]; identification of omitted details in models [37][40]; checking of modeling inconsistency [41]; and, identification of specification violations [42]; 2) Improving communication and building models more complete: the contribution was reported as follows: improvement of understanding among software developers [6]; improvement in communications among stakeholders [14-15][63], identification of the correct construct language [14]; building of more precise domain models [60]; performance improvement of domain analysts as far as accuracy and the covering of conceptual models are concerned [61]; and, improvement of quality of the models built by novice analysts [39]; and, 3) Allowing traceability among artifacts and Quality requirement identification: the contribution was reported as follows: allowing traceability among artifacts [75][76]; improvement in the requirement identification quality [56]; and, support in the analyst quality attributes [70].

5. CONCLUSION

Ontologies have been playing a versatile role in solving problems related to SRE. Considering this versatility, a systematic review of the literature was conducted in order to obtain a broader view of which roles ontologies play in SRE. Some of the main results relating to the role of ontologies in the SRE are discussed as follows.

Ontologies have an effective role in the Analysis, Specification and Elicitation activities. Although no proposals were identified to the Negotiation and Validation activities, it is considered that ontologies have potential to be also applied in these activities. Ontologies have a more conceptual application when the focus is not yet the software implementation, but the understanding of the domain to which the software will be developed to. On the other hand, it has a more computational application when the produced artifacts aim the software implementation. There is an important concentration of ontology proposals applied to models (conceptual, business, activity, process etc.). Models have been an important resource not only to the requirements analysis, but it is also important to the model transformation in the software design phase. Although a great number of ontology proposals have been identified, there is no universal ontology that integrates all the artifacts produced in the process.

The most important contributions of ontologies are: identifying problems in specification and models, improving communication, building models more complete, allowing traceability among artifacts and improving the quality of requirements identification. It has more ontology contributions in the academic than in the industry context. This result may be evidence that ontologies are not too widespread in industry. If ontologies are indeed still not too applied in industry, this review becomes an import source of academic research and also a source of knowledge transfer between the academic and industry context.

References

[1] Thayer RH, Dorfman M (1997) Software Requirements Engineering, 2d Ed. IEEE Computer Society Press
[2] Robinson W, Pawlowski S (1999) Managing requirements inconsistency with development goal monitors. IEEE Transactions on Software Engineering 25(6):816-835
[3] Loucopoulos P, Karakostas V (1995) System Requirements Engineering. McGraw-Hill
[4] Pohl K (1994) The three dimensions of requirements engineering: a framework and its applications. Information systems 19(3):243-258
[5] Jureta IJ, Borgida A, Ernst N, Mylopoulos (2010) Techne: Towards a New Generation of Requirements Modeling Languages with Goals, Preferences, and Inconsistency Handling. In: International Requirements Engineering Conference pp 115–124

[6] Oliveira KM, Zlot F, Rocha AR, Travassos GH, Galotta C, Menezes CS (2004) Domain-oriented software development environment. Journal of Systems and Software 72:145–161
[7] Garrido JL, Noguera M, González M, Hurtado MV, Rodríguez ML (2007) Definition and use of Computation Independent Models in an MDA-based groupware development process. Science of Computer Programming 66:25–43
[8] Lee SW, Gandhi RA (2005) Ontology-based active requirements engineering framework. In: 12th Asia-Pacific Software Engineering Conference (APSEC '05)
[9] Aranda GN, Vizcaíno A, Cechich A, Piattini M (2008) A Methodology for Reducing Geographical Dispersion Problems during Global Requirements Elicitation. In: Workshop on requirements engineering
[10] Girardi R, Leite A (2008) A knowledge-based tool for multi-agent domain engineering. Knowledge-Based Systems 21:604–611
[11] Li L (2005) Ontological modeling for software application development. Advances in Engineering Software 36:147–157
[12] Harzallah M, Berio G, Opdahl AL (2012) New perspectives in ontological analysis: Guidelines and rules for incorporating modelling languages into UEML. Information Systems 37:484–507
[13] Zhang H, Kishore R, Sharman R, Ramesh R (2007) Agile Integration Modeling Language (AIML): A conceptual modeling grammar for agile integrative business information systems. Decision Support Systems 44:266–284
[14] Kilov H, Sack I (2009) Mechanisms for communication between business and IT experts. Computer Standards & Interfaces 31:98–109
[15] Castañeda V, Ballejos L, Caliusco ML, Galli MR (2010) The use of ontologies in requirements engineering. Global Journal of Researches in Engineering 10(6):2–8
[16] Dermeval, Diego, et al. "Applications of ontologies in requirements engineering: a systematic review of the literature." Requirements Engineering(2015): 1-33.
[17] Pohl K (1997) Requirements engineering: An overview. In Encyclopedia of Computer Science and Technology. A. Kent, and J. Williams, NY, vol. 36, suppl. 21
[18] Abran A, Moore, J W (2004) Guide to the Software Engineering Body of Knowledge. Society. IEEE Press.
[19] Kotonya G, Sommerville L. (1998) Requirements Engineering: Processes and Techniques. John Wiley & Sons.
[20] Gruber TR (1995) Toward Principles for the Design of Ontologies Used for Knowledge Sharing. International Journal of Human and Computer Studies 43(5-6):907-928
[21] Guarino N (1998) Formal Ontology and Information Systems. In the Proceedings of Formal Ontology in Information Systems, Washington, DC: IOS Press p 3-15
[22] Noy NF, Mcguinness DL (2001) Ontology Development 101: A Guide to Creating your First Ontology, Stanford Knowledge Systems Laboratory Technical Report KSL-01-05
[23] Guizzardi G (2007) On ontology, ontologies, conceptualizations, modeling languages, and (meta) models. Frontiers in artificial intelligence and applications pp 18–28
[24] Kitchenham B (2004) Procedures for performing systematic reviews. Keele, UK, Keele University, v 33
[25] Wohlin C, Höst M, Henningsson K (2003), Empirical Research Methods in Software Engineering, in Empirical Methods and Studies in Software Engineering, Lecture Notes in Computer Science, R. Conradi and A. I. Wang, Eds.: Springer, pp. 7-23
[26] Santana B, Diniz S, Barbosa J, Leite JCP (2008) A Language-Based Approach to Variability Analysis. In: Workshop on Requirements Engineering pp 179–190
[27] Chen X, Yin B, Jin Z (2010) Dptool: A Tool for Supporting the Problem Description and Projection. International Requirements Engineering Conference pp 401–402
[28] Hilaire V, Cossentino M, Gechter F, Rodriguez S, Koukam A (2013) An approach for the integration of swarm intelligence in MAS: An engineering perspective. Expert Systems with Applications 40:1323–1332
[29] Cañete-Valdeón JM, Galán FJ, Toro M (2009) The intentional relationship of representation between the constructs of a language and reality. Data & Knowledge Engineering 68:173–191
[30] Liu, C (2010) CDADE: Conflict detector in activity diagram evolution based on speech act and ontology. Knowledge-Based Systems 23:536–546
[31] Ovaska E, Evesti A, Henttonen K, Palviainen M, Aho P (2010) Knowledge based quality-driven architecture design and evaluation. Information and Software Technology 52:577–601
[32] Vongdoiwang W, Batanov DN (2006) An ontology-based procedure for generating object model from text description. Knowledge and Information Systems, 10(1):93–108
[33] Pires PF, Delicato FC, Cóbe R, Batista T, Davis JG, Song JH (2011) Integrating ontologies, model driven, and CNL in a multi-viewed approach for requirements engineering. Requirements Engineering, 16:133-160
[34] Lindoso AN, Girardi R (2006) The SRAMO Technique for Analysis and Reuse of Requirements in Multi-agent Application Engineering. In Workshop on Requirements Engineering pp 1–10
[35] Bimrah KK, Mouratidis H, Preston D (2008) Modelling Trust Requirements by Means of a Visualization Language. Requirements Engineering Visualization pp 26–30
[36] Rajsiri V, Lorré JP, Bénaben F, Pingaud H (2010) Knowledge-based system for collaborative process specification. Computers in Industry 61:161–175
[37] Beydoun G, Low G, Tran N, Bogg P (2011) Development of a peer-to-peer information sharing system using ontologies. Expert Systems with Applications. 38:9352–9364
[38] Santos EG, Medeiros AP (2011) Design Rationale Representation in Requirements Engineering using the KAOS meta-model In: Workshop on Requirements Engineering
[39] Bolloju N, Sugumaran V A (2012) Knowledge-based object modeling advisor for developing quality object models. Expert Systems with Applications 39:2893–2906
[40] Bolloju N, Schneider C, Sugumaran V (2012) A knowledge-based system for improving the consistency between object models and use case narratives. Expert Systems with Applications 39:9398–9410
[41] Fan S, Zhao JL, Dou W, Liu M (2012) A framework for transformation from conceptual to logical workflow models. Decision Support Systems 54:781–794
[42] Ghidini C, Francescomarino C, Rospocher M, Tonella P, Serafini L (2012) Semantics-Based Aspect-Oriented Management of Exceptional Flows in Business Processes. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 42:25–37
[43] Wand Y, Monarchi DE, Parsons J, Woo CC (1995) Theoretical foundations for conceptual modelling in information systems development. Decision Support Systems 15:285–304

[44] Green P, Rosemann M (2000) Integrated process modeling: an ontological evaluation. Information systems 25(2):73-87
[45] Rosemann M, Green P (2002) Developing a meta model for the Bunge–Wand–Weber ontological constructs. Information Systems 27:75–91
[46] Wagner G (2003) The Agent–Object-Relationship metamodel: towards a unified view of state and behavior. Information Systems 28:475–504
[47] Evermann J, Wand Y (2005) Toward formalizing domain modeling semantics in language syntax. IEEE Transactions on Software Engineering 31:21–37
[48] Gemino A, Wand Y (2005) Complexity and clarity in conceptual modeling: Comparison of mandatory and optional properties. Data & Knowledge Engineering 55:301–326
[49] Etien A, Rolland C (2005) Measuring the fitness relationship. Requirements Engineering 10(3):184–197
[50] Perez GC, Sellers, BH (2007) Modelling software development methodologies: A conceptual foundation. Journal of Systems and Software 80:1778–1796
[51] Dreiling A, Rosemann M, Wil MPA, Sadiq W (2008) From conceptual process models to running systems: A holistic approach for the configuration of enterprise system processes. Decision Support Systems 45:189–207
[52] Jureta IJ, Mylopoulos J, Faulkner S (2008) Revisiting the Core Ontology and Problem in Requirements Engineering. In: International Requirements Engineering Conference pp 71–80
[53] Bera P, Evermann J (2012) Guidelines for using UML association classes and their effect on domain understanding in requirements engineering. Requirements Engineering
[54] Regoczei S, Plantinga EPO (1987) Creating the domain of discourse: ontology and inventory. International Journal of Man-Machine Studies. 27:235–250
[55] Jin Z (2003) Automatically multi-paradigm requirements modeling and analyzing: An ontology-based approach. Science in China 46(4):279-297
[56] Kaiya H, Saeki M (2006) Using Domain Ontology as Domain Knowledge for Requirements Elicitation. In: International Requirements Engineering Conference pp 189–198
[57] Girardi R, Marinho LB (2006) A domain model of Web recommender systems based on usage mining and collaborative filtering. Requirements Engineering 12(1):23–40
[58] Breaux, T (2009) Exercising Due Diligence in Legal Requirements Acquisition: A Tool-supported, Frame-Based Approach. In: International Requirements Engineering Conference pp 225–230
[59] Biletskiy Y, Ranganathan GR (2010) A semantic approach to a framework for business domain software systems. Computers in Industry 61:750–759
[60] Tacla CA, Freddo AR, Paraiso EC, Ramos MP, Sato GY (2011) Supporting small teams in cooperatively building application domain models. Expert Systems with Applications 38:1160–1170
[61] Bagheri E, Ensan F, Gasevic D (2012) Decision support for the software product line domain engineering lifecycle. Automated Software Engineering 19(3):335–377
[62] Cossentino M, Gaud N, Hilaire V, Galland S, Koukam (2009) A ASPECS: an agent-oriented software process for engineering complex systems. Autonomous Agents and Multi-Agent Systems 20(2): 260–304
[63] Aranda GN, Vizcaíno A, Piattini M (2010) A framework to improve communication during the requirements elicitation process in GSD projects. Requirements Engineering 15(4):397-417
[64] Cysneiros LM, Kushniruk A (2003) Bringing Usability to the Early Stages of Software Development Usability Ontology References. In: International Requirements Engineering Conference
[65] Kof L (2007) Scenarios: Identifying Missing Objects and Actions by Means of Computational Linguistics. In: International Requirements Engineering Conference pp 121–130
[66] Katz S, Rashid A (2004) From aspectual requirements to proof obligations for aspect-oriented systems. International Requirements Engineering Conference pp 43–52
[67] Masuwa-Morgan K, Burrell P (2004) Justification of the need for an ontology for accessibility requirements (Theoretic framework). Interacting with Computers 16:523–555
[68] Cabral G, Sampaio A (2008) Formal Specification Generation from Requirement Documents. Electronic Notes in Theoretical Computer Science. 195:171–188
[69] Weber-Jahnke JH, Onabajo A (2009) Finding Defects in Natural Language Confidentiality Requirements. In: International Requirements Engineering Conference pp 213–222
[70] Rago A, Marcos C, Diaz-Pace JA (2011) Uncovering quality-attribute concerns in use case specifications via early aspect mining. Requirements Engineering, 18(1):67–84
[71] Castañeda V, Ballejos L, Caliusco ML (2012) Improving the Quality of Software Requirements Specifications with Semantic Web Technologies. In Workshop on requirements engineering
[72] Verlaine B, Dubois Y, Jureta IJ, Faulkner S (2012) Towards conceptual foundations for service-oriented requirements engineering: bridging requirements and services ontologies. IET Software. 6(2):85-102
[73] Richter H, Gandhi R, Liu L, Lee S, Carolina N (2006) Incorporating Multimedia Source Materials into a Traceability Framework. In: International Workshop on Multimedia Requirements Engineering p 5
[74] Gandhi RA, Lee SW (2007) Discovering and Understanding Multi-dimensional Correlations among Certification Requirements with application to Risk Assessment. In: International Requirements Engineering Conference. 231–240
[75] Zhang Y, Witte R, Rilling J, Haarslev V (2008) Ontological approach for the semantic recovery of traceability links between software artefacts. IET Software 2(3) pp 185-203
[76] Assawamekin N, Sunetnanta T, Pluempitiwiriyawej C (2009) Ontology-based multiperspective requirements traceability framework. Knowledge and Information Systems 25(3):493–522
[77] Chicaiza J, López J, Piedra N, Martínez O, Tovar E (2010) Usage of social and semantic web technologies to design a searching architecture for software requirement artifacts. IET Software 4(6): 407–417
[78] Supakkul S, Chung L (2010) Visualizing non-functional requirements patterns. In: International Workshop on Requirements Engineering Visualization pp 25–34
[79] Cappelli C, Leite JCSP, Oliveira APA (2007) Exploring Business Process Transparency Concepts. In: International Requirements Engineering Conference pp 389–390
[80] Kaindl H, Svetinovic D (2010) On confusion between requirements and their representations. Requirements Engineering, 15(3):307–311

On Non Functional Requirements, Their Evolution and Impact on Safety

Abdelaziz Babiker[1], Abd-El-Kader Sahraoui[2]

[1]. Sudan University of Science and Technology (SUST) Khartoum Sudan
[2]. LAAS-CNRS, University de Toulouse, CNRS, UT2J, Toulouse, France

Abstract: *The paper is on preliminary work on systems engineering concepts and their deployment. We consider requirement evolution with a systems engineering approach can be modelled by systems engineering processes at least for the on development phases of the product. The paper is very technology oriented and wholly based on EIA-632 standards; the paper is extended with requirement evolution issues and a methodological approach is proposed.*

Keywords: requirements engineering, systems engineering, requirements change, safety requirements.

1- Introduction

The actual systems are more and more complex, because they integrate a variety of technologies. The system need often long period of development. When we take a change on requirements the objectives are to improve the functionality, the cost or the delay of systems; but unfortunately, this modification can be affect a problem with others requirements those involved with the safety of the system.

The requirements change occurs in two main cases. The first concerns revising/updating existing requirements that led to an actual version of the systems to adapt to new environment (Larsen and Buede, 2002). The second is when new technology is being developed and new requirements are implemented consequently for reasons of cost or feasibility.

When we consider the change of requirements, we focus mainly in this paper to study the effect of the change on the security of the system; we try to see if any change cannot harm the system and the risk to affect the systems security. You have two type of change for requirements. The first such as the requirement change and the system is in the phase of development, the second when we take a modification for a realize requirements

The main idea in this part is to present the effect of change on the system if one or many requirements change, and to present what are the considerations we must make, and to know the importance of safety requirements.

A recent survey produced Customer Focus group (CFG) of an aircraft manufacturer indicated how airliners keen on the integration of new technologies as requirement change for more functionalities rather integrating new technologies that may not be appropriate for safety issues and show that new technologies that succeeded in some application may not do for other application as there was no assessment of side effects on performance and safety for the whole system.

Our goals in this paper are to study:

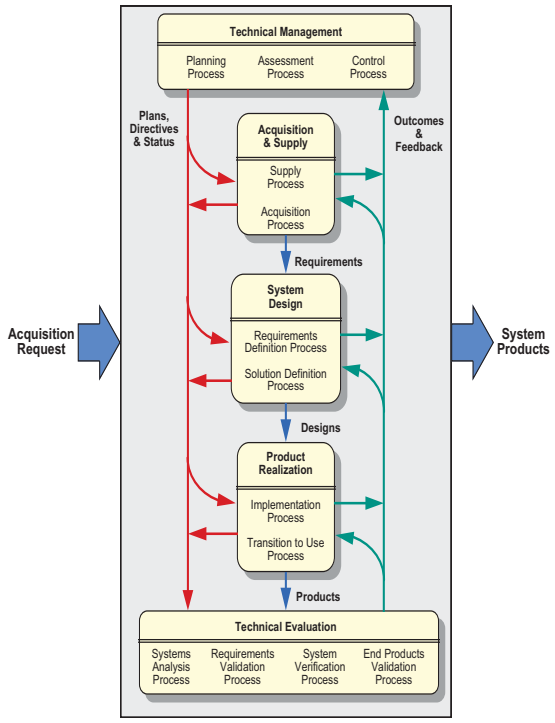
1. The problematic if we take a modification (on one or many) requirements, how can be sure that we have not any problem on safety of the system?
2. The problematic of How can we take a study of the effect on safety when we have a modification?
3. The problematic that how can we know the requirements have link with safety, and how can we know the safety requirements?
4. What type of change (functional requirements, non-functional requirements, physical requirements, operational requirements…) make a problem of security of the system?

2- Requirements engineering

Requirements Engineering covers all activities related to the elicitation, modelling, analysis, specification and management of requirements throughout the systems development life cycle. RE is a multi-disciplinary and communication rich part of software development that utilises a variety of techniques and tools at different stages of development and for different kinds of application domains.

The processes are applicable for the engineering or reengineering of the end products that make up a system, as well as the development of enabling products required to provide life-cycle support to system end products. The next graphic shows the

relationships between the processes.



Our context study will, through the system engineering framework, give importance to end products aggregation of end products.

Enabling products are used to perform the associated process functions of the system develop, produce, test, deploy, and support the end products; train the operators and maintenance staff of the end products; and retire or dispose of end products that are no longer viable for use. Both the end products and the enabling products are either developed or reused, as appropriate. The relationship of these system elements is shown in Figure 2.

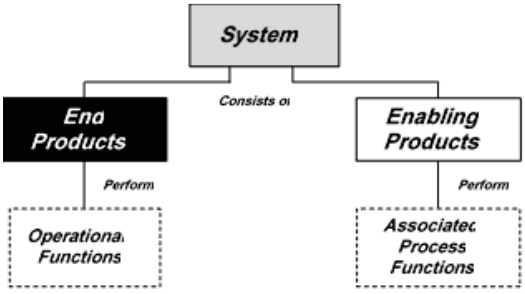


Figure 2: System concept

The system forms the basis for a larger structure, called the building block, shown in Figure 3. The building block. Effectively, it can be seen the end product is separated from other enabling product issues; this framework will help companies to enhance their RE process development when considering critical issues that often get mixed up with final product; the infrastructure for agile development relies highly on what is often called logistics but in fact they are the enabling product in systems engineering view: training, deployment products, test, development, product etc. as shown in figure 3.

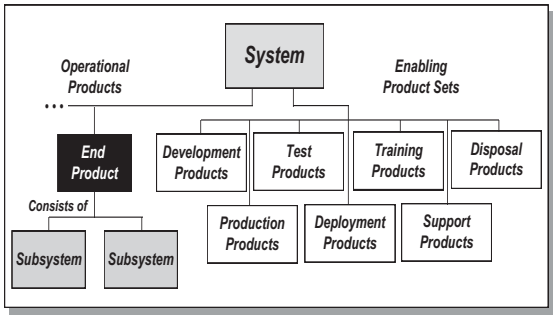


Figure 3: Building block

For such objective, standard EIA-632 will be considered as the mostly popular and effective standard being used; such standard has been deployed in major space, manufacturing, military and aeronautic industries.

Errors in requirements specifications can have a major impact on software costs and project duration. It is evident that early detection and correction of potential problems during requirement analysis may alleviate many much larger problems later on during testing and maintenance.

Requirements engineering is a process from system engineering, only active in an upstream phase, the objectives of this process is to identify the stakeholders, to capture, to explain, to formalise, to analyse, to validate, to allocate requirements. One specification for any system contains a highly number of requirements. The specification is necessary to make requirements very coherent, and to control, to change, to manage requirements.

Many studies in Requirements Engineering show that 90% of costs are engage in the first phase of development of System Engineering, and that is better to change one or many requirements in the first phase of development of the system. But when we apply a modification on our specification we have an impact not only in cost but also in safety. In Concurrent Engineering (CE) there should be a virtual data bus communicating between every process so to enable to

activate parallel processes when a change is being made during systems development.

3-Mapping SE to CE

The mapping we prone is how to customise the SE processes to CE. Such mapping is not systematic as it depends how CE is deployed in every enterprise.

As stated in European research for the integration in manufacturing Information technologies already have a pervasive influence on modern manufacturing and engineering. This ranges from supporting the design and engineering process, through production planning and control, and on the toe control of manufacturing equipment and distribution systems. However, there remains enormous scope for IT growth in the manufacturing sector.

The main topics covered within the Integration in Manufacturing research area include:

1. Product data exchange and modelling which deals largely with the internal representation of all product-related design and manufacturing data.
2. Factory automation, which includes the development of software tools to support human-centered production concepts; advanced work in robotics; and software developments in the field of simulating products, the manufacturing process, and the design and layout of manufacturing cells, assembly work-stations and entire plants.
3. Communications and logistics, to improve the integration of distributed manufacturing applications, both within a plant and between enterprises collaborating in a supply chain.

Unifying processes for the needs until the disposal cannot be attained sequentially in any product or service development.

The systems development is based on unified process. These processes make abstraction of the systems nature. In [Sahraoui et al, 2004) we prone numerous research issues related to the subject.

3.1 A Decision focused framework for Life-Cycle Based Architecting and design in SE

3.1.1 Definition of the problem

Systems engineering is a multi-disciplinary problem definition and problem solving process that is implemented by people. There are as many definitions of this process as there are systems engineers with no real agreement on an underlying theory that unifies the process. Most systems engineers will agree to the following characterization of systems engineering:

1. Focus: a process and systems management focus that will result in the engineering of a trustworthy product or operational system that will satisfy user and customer requirements and for which these stakeholders will pay
2. Scope: entire life cycle of the system, including the definition of user and customer requirements, development of the system products and enabling products, and deploying them in an operational environment. These enabling product systems include test system, deployment system, training system, operational support (logistics, maintenance, etc.), refinement system, and retirement system
3. Products: Systems Engineering Management Plan, Operational Concept for the product, hierarchy of requirements documents for each key system (starting with the system-level requirements document and following the physical decomposition of the system), architectures and hierarchy of interface control documents that define the interfaces at each level of the physical decomposition
4. Characteristics of SE Process: Combination of qualitative, quantitative, and executable models to examine the behavioral (functional) and system-wide (non-functional) characteristics of alternate designs and architectures.

3.1.2 Research approach

A design process is characterized by a collection of decisions. In this, we use the fundamentals of decision analysis in which a decision is characterized by alternatives (what you can do – designs), values (objectives hierarchy with a quantitative value model to describe the trade-offs of the stakeholders across the key measures of effectiveness), and facts about what is known and not known. Within this context view systems engineering as a risk mitigation strategy that includes architecture, design, and testing. We must recognize that the entire process must adhere to the following principles:

1. Coherent value structure across all decisions.
2. Top-down, decentralized (distributed, asynchronous) decision making.
3. Managed by an adaptive, feedback-control process for decision making.

4. Focused, cost-effective, risk management of both the (life cycle) design and design process.

3.1.3 Expected results

1. Integration of values across all decisions for the system's life cycle
2. Architecture and design framework for an integrated and coordinated decision-making framework with a schedule that identifies serial and concurrent decision-making activities
3. Structure for reviews of key products that is based on the principles of feedback-control systems as well as the coordinated decision framework and is sensitive to the uncertainties
4. Framework for risk management that is sensitive to the integrated values across the system's life cycle and the decision framework.
5. Process that can be generalized to other problem solving situations.

4- Refining the approach : EIA Standard application to CE

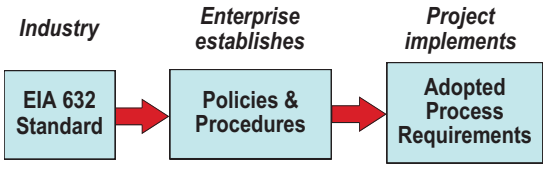
The EIA 632 as briefly presented in part 1 is common standard adopted in aeronautics and space industry and still to be deployed in other sector with other standard as IEEE –P1220 and ISO-15288.

Our work in such limited only to the design and verification processes (8 processes from the total 13 processes); the left out processes concern the management, supply and acquisition processes.

We are concerned also by the products and enabling products, that is really implicit in concurrent engineering and recommended in EIA 632.

MIL-STD-499B was intended to be the first military standard to address systems engineering as a whole. MIL-STD-499A addressed the management of systems engineering and, thus, had a different focus, although MIL-STD-499B was intended to supersede it. MIL-STD-499B uses a lengthy definition for systems engineering which concentrates on integration of disciplines, full life-cycle coverage, assurance of interface integrity, management of technical risks, and validation that the system meets the needs and requirements. The actual work of systems engineering is clarified only by reading the full standard. This work tends to lean heavily toward the Technical Management definition of systems engineering.

The standard begins with five pages of definitions and follows that with one chapter on General Requirements and one on Detailed Requirements. The General Requirements chapter is ordered by life-cycle phase, while the Detailed Requirements call out individual systems engineering work products. These work products clearly imply a large military contract environment, calling out numerous customer reviews, for example, and including things like Survivability tasks and Integrated Logistics support.



Application of the standard

While this standard was never released, the May 1992 copy in particular has enjoyed a long applicable life in that it served as the basis for both the IEEE 1220 standard and the EIA IS 632 standard, both of which were fairly minor modifications.

4.1 Main Product

The product to be designed: from requirement to retrieval; such product can be decomposed in sub products and so on.

4.2 Enabling Product

5- The requirement evolution in CE context.

Our approach and contribution will focus mainly on impact of requirement change and development a methodology for requirements change. This will be carried on the basis of:

a) traceability model
b) The concurrent processes
c) A formal framework for the requirement change

5.1 General approach

We are investigating many approaches to such issue. However, the global approach is thought as an operational view as illustrated by the following figure. The formal basis for such approach is not tackled yet but some items are thought to be useful and to be discussed in latter section.

This preliminary approach is systems engineering context characterised by the interaction by four models; these involve respective processes. Our concern is the development of the change model and its interaction with all other models. We will present the traceability model dynamics that have been used in earlier work [Hellouin1 et al.] [Hellouin2], and make abstraction of the development and system configuration management models as their basic characteristics are known for long time. We know that any requirement change will concern and trigger all four models. In our first approach we will be concerned the change, traceability and development models. However, some principles will guide towards the deepening of the approach as future work will focus mainly on refining the approach:

1. Any change request either at any step of development model suppose the availability of a traceability model.
2. A change request for an operation module will necessarily require tracing back the original requirement
3. Make distinction between functional and non-function requirements
4. Identify security/safety requirements.
5. Create link between associated function and safety requirement.

5.2 Traceability model

As discussed earlier, providing traceability of requirements to their sources and the outputs of the system development process can be along several dimensions. Different stakeholders contribute to the capture and use of traceability information, often with different perspectives. A user has a different vision from an audit specialist, a system designer or a validation engineer. Some typical questions are often asked:

What are the systems components that are affected by a specific requirement?

Why are the components affected by such requirements?

How are the components affected by such requirement?

What are the sources of a low level requirement?

Why and how two requirements are related?

And so on …

An object can belong to one of the following classes: requirement, design, components, system/subsystem, etc. Attributes and operations (activities) are associated with each class, subclass.

Sources are all available information as documents, phone call, E-mail about the object lifecycle. Traceability concerning specific decision made can be found through the relation documents.

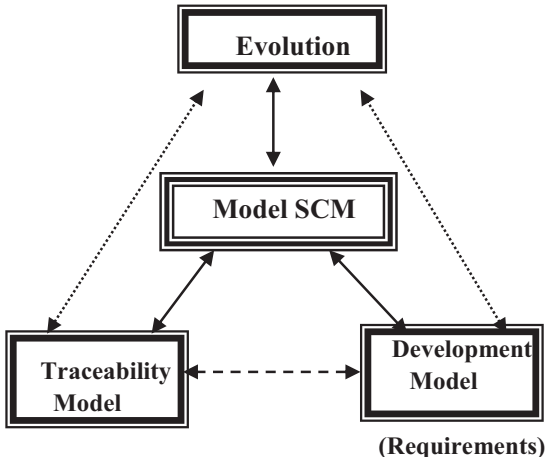


Figure 3: Traceability at low level

We can use this traceability model to identify any link that may be subject and constrained by a requirements change.

Conclusion

An extension of the approach for key non-functional requirement is the objecting of future research.

References

[1] Barry, E.J.; Mukhopadhyay, T.; and Slaughter, S.; Software Project Duration and Effort: An Empirical Study, 2002, Information Technology and Management, vol. 3, pp. 113-136.

[2.] Buren, J.V.; cook, D.; 1998, Experiences in the Adoption of Requirements Engineering Technologies, Journal of Defence Software Engineering, December, pp.3-10.

[3] Hellouin, L.; Beaugrand, J.L.; and Sahraoui, A.E.K.; 2001, Requirements process and traceability issues, 11th Annual INCOSE Symposium, Melbourne.

[4] Hellouin, L.; 2002, Contribution à l'ingénierie des exigences et à la traçabilité, PhD Thesis, LAAS-CNRS and INPT, LAAS-Report 02074.

[5] El-Jamal, H.M: Requirements evolution and impacts on safety. IFIP 18th World Computer Congress, August 2004, Toulouse.

[6] El-Jamal, H.M, Sahraoui, A.E.K: requirements evolution methodology and impacts on safety. Quality conference, Bordeaux, Mars 2005.

[7] Hughes, T.; Cindy, M.; Design traceability of complex systems. In Human Interaction with Complex Systems, pages 37–41, March 1998.
[8] Krasner, H.; 1989, Requirements Dynamics in Large Software Projects, 11th World Computer Congress (IFIP89), Amsterdam, Netherlands.

[9] LuizMarcio, C.; Julio Cesar, S. d. P.L., Nonfunctional Requirements: From Elicitation to Conceptual ModelsIEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 30, NO. 5, MAY 2004 Engineering, Fort Collins.

[10] Nurmuliani, N.; ZowghiD.; and Fowell S., 2004, Analysis of Requirements Volatility during Software Development Life Cycle, proceedings of the Australian Software Engineering Conference (ASWEC), April 13-16, Melbourne, Australia.

[11] Sahraoui, A.E.K, Buede, D., Sage, A: Issues in systems engineering research. Incose Int'l Symposium, June 2004, Toulouse.

[12] Stevens R;Brook, P.; Systems Engineering - Coping with complexity. Prentice Hall, 1998
[13] Traoré, I.; El Jamal, M.H.; Yanguo Liu, M.; Sahraoui, A.E.K.; 2004, UML-PVS for requirements specification. Incose symposium, Toulouse, France.

[14] Traoré, I.; Sahraoui, A.E.K.; 1997, A Multiformalism Specification Framework with Statecharts and VDM, 22nd IFAC/IFIP Workshop on Real Time Systems,Lyon, France.

[15] Zowghi, D.;Nurmuliani, N.; 2002, A Study of the Impact of Requirements Volatility on Software Project Performance, 9th Asia-Pacific Software Engineering Conference, Gold Coast, Australia.
[Sheard, 2003] SheardS.A : Three types of systems engineering application. Incose symposium, 2003

Use Case Mapping vs. Thing-Oriented Modeling of Software Requirements

S. Al-Fedaghi
Computer Engineering Department, Kuwait University, Kuwait

Abstract - *Use case maps (UCMs) have been introduced in software engineering as a means to bridge the gap between requirements and design in software development. This problematic gap originated from ineffective elicitation of system requirements. A great deal of difficulty in developing software can be traced to the initial phase of the development life cycle, when analyzed requirements and design schemata come together. UCMs relate use cases in a maplike diagram built from components and associated responsibilities linked by the path of the scenario they depict. This paper proposes adopting an alternative to UCM conceptualization that could contribute to the alleviation of some of the problems posed by the requirements/design gap. The proposed approach is based on so-called Thing-Oriented modeling utilizing flow-based diagrammatic methodology. Accordingly, some examples from the literature are reformulated using the proposed flow-based representation. The resultant schemata seem more suitable for developing structures in the design phase.*

Keywords: Software development life cycle, elicitation of system requirements, use case mapping, conceptual representation

1 Introduction

A great deal of difficulty encountered in development of software can be traced to the initial phase of the development life cycle, when *analyzed requirements* and *design schemes* are brought together. Low user involvement and participation as well as improper requirements specifications are the top causes of more than a third of IT project cancellations and cost increases. On average, barely more than one in six software projects are ever completed on-time and on-budget [1, 2, 3, 4].

Many methodologies have been proposed to handle such a problem, including the so-called component bus system and properties [5, 6], study of cognitive challenges of design [7], text analysis approaches, and natural language processing [8].

Use Case Map (UCM) notation has been used to bridge the requirements/design gap in software development [9] and to describe scenario-based aspects at the requirements level. UCM is a visual standard representation for the materialization of scenarios including both static and dynamic aspects of a system [10]. It can be constructed from informal requirements or from use cases expressed in natural language and diagrams. UCMs relate use cases in a *maplike diagram* built from components and associated responsibilities linked by the path of the scenario they depict. UCM is used to describe relationships between *responsibilities* (e.g., actions, activities, operations, tasks to perform) and *components* (e.g., objects, processes, databases, servers, functional entities, network entities, users, actors, processors), which may potentially be bound to underlying organizational structures of abstract components [11]. Some works have extended UML with UCM concepts [12, 13].

UCM notation comprises a set of start-points, filled circles representing preconditions; responsibilities, representing the tasks, are denoted by crosses; end-points are indicated by bars that represent post-conditions; and components are shown as boxes to represent a software entity containing responsibilities.

It is claimed that UCMs can clarify the functional description of a system and eliminate possible errors in user requirements. They have the advantage of being flexible and maintainable, and they encapsulate different types of information in a single view [14].

> One of the advantages of using the UCM notation is that it is easily understandable by any stakeholder of the system, allowing designers to reduce the gap between client needs and requirement analysts. [8]

This paper proposes adopting an alternative to UCM conceptualization that could contribute to the alleviation of some of the problems encountered in the gap between requirements and design. The proposed approach is based on so-called *thing-oriented modeling* and utilizes flow-based diagrammatic methodology. The resultant schemata seem more suitable for developing structures in the design phase. To substantiate such a claim, the paper contrasts the two approaches side-by-side by reformulating (a) a UCM from the literature and (b) its representation. This comparison serves as evidence of the benefit of the latter methodology.

Thing-oriented modeling is used to model a portion of reality through *things* that flow in "river basins," called flow systems, that are part of greater "territories" called spheres and subspheres. Operations on *flowing things* are categorized and limited to creation, release, transfer, receiving and processing. Relationships among spheres are represented in Venn-like diagrams. Elements of such a model are matched with elements of the domain system. According to Frigg [15],

> A system is a "compact" and unstructured entity and we have to *carve it up* in order to impose a structure on it. Structures do not really exist until the scientist's mind actually "creates" them or, to put it in a less pretentious way, ascribes them to a system. Structures are not "ready-made" but result from a way of taking, or demarcating, the system. (Italics added)

This thing-oriented modeling utilizes a diagrammatic language called the Flowthing Model (FM). FM has been used in several applications (e.g., [16, 17, 18, 19, 20]). For the sake of a complete presentation, the basic notions in FM are briefly described in the next section.

2 Flowthing Model

The Flowthing Model (FM) is a diagrammatic language that uses *flowthings* to represent a range of items, for example, electrical, mechanical, chemical, and thermal signals, circulating blood, food, concepts, pieces of data, and so on. *Flowthings* are defined as what can be created, released, transferred, processed, and received (see Figure 1). Hereafter, flowthings are referred to as *things*. Note that what we call a thing is not necessarily a substance in the philosophical sense, e.g., heat is a thing that is created, released, transferred, received, and processed.

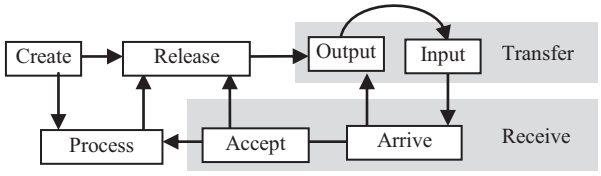


Fig. 1. Flow system

FM depicts *processes* using *flow systems* that are formed by up to seven stages (states) depending on the requirements and details of a system (Figure 1). These stages can be embedded in a network of assemblies called spheres in which the processes of flow systems take place.

The stages in Figure 1 can be described as follows:

Arrive: A thing reaches a new flow system.

Accepted: A thing is permitted to enter a machine. If arriving things are always accepted, *Arrive* and *Accept* can be combined as a **Received** stage.

Processed (changed): The thing goes through some kind of transformation that changes it without creating a new thing.

Released: A thing is marked as ready to be transferred outside the flow system.

Transferred: The thing is transported somewhere from/to outside the flow system; here, two separate stages of Output and Input can be defined instead of the single stage of Transfer, if needed.

Created: A new thing is born (created) in a flow system.

The flow system of Figure 1 is a generalization of the typical system model of input-process-output used in many scientific and engineering fields. In general, a flow system is thought to be an abstract machine that receives, processes, creates, releases, and transfers things. The stages are mutually exclusive (i.e., a thing in the Process stage cannot be in the Create stage or the Release stage at the same time). An additional stage of *Storage* can also be added to any machine to represent the storage of things; however, storage is not an exclusive stage because there can be *stored processed* flowthings, *stored created* flowthings, etc.

FM also uses the notions of *spheres and subspheres*. These are the network environments and relationships of machines and submachines. Multiple flow systems can exist in a sphere if needed. A sphere can be a person, an organ, an entity (e.g., a company, a customer), a location (a laboratory, a waiting room), a communication medium (a channel, a wire). A flow system is a subsphere that embodies the flow; it itself has no subspheres.

FM also utilizes the notion of *triggering*. Triggering is the activation of a flow, denoted in FM diagrams by a *dashed arrow*. It is a (causative) dependency among flows and parts of flows. A flow is said to be triggered if it is created or activated by another flow (e.g., a flow of electricity triggers a flow of heat), or activated by another point in the flow. Triggering can also be used to initiate events such as starting up a flow system (e.g., remote signal to turn on). Multiple flow systems captured by FM can interact by triggering events related to other flow systems in their spheres and stages.

3 Applying FM to analyzed requirements and design schemata

This section is the main contribution of this paper. It demonstrates that FM can be applied as an alternative to use case mapping; hence, it could help to alleviate the problem of failure to effectively elicit system requirements. Such a demonstration will show two reformulated examples from the literature.

3.1 Use case mapping utilizing natural language mining

Casamayor et al. [8] introduce an approach for mining and grouping functionality starting from informally written requirements using a combination of natural language processing and text clustering algorithms. The purpose is to identify potential software responsibilities and components of the system to be developed. A set of rules is applied to every *verb* phrase in the requirements description to select candidate responsibilities. An initial partition of the candidate responsibilities is produced to give the designer some insight into possible conceptual components of the desired architecture.

For example, consider the requirement "Add New Client," which can be transcribed as follows:

> *The manager selects "add new client" from the system menu. The system displays a blank client form. The manager enters the social security number of the new client. The system retrieves the client list and checks that the client does not exist. The manager enters the required client information fields: name, date of birth, postal address and credit card number. The manager selects "save changes". The system updates the client information.* [8]

Accordingly, from the sentence *The manager selects "add new client" from the system menu*, we can extract:
Noun: *the manager*, identified as the *actor* of this requirement
Verb: *selects* "add new client" from the system menu, where the verb is considered an external input to the system.

Consequently, the *responsibility of the system* would be to wait for user input, i.e., from the manager.

Such a mining process is followed by grouping of the identified responsibilities, e.g., a cluster, as shown in Figure 2. A complete description of the method is detailed in [8].

Suppose that we model this "add new client" scenario using FM. Can the FM diagram accomplish the same aim of identifying potential software responsibilities and components of the system to be developed? Figure 3 shows such a representation.

The system displays a blank client form.
NP: the system
VP: displays a blank client form
→ Candidate responsibility: display blank client form
The manager enters the social security number of the new client.
NP: the manager
VP: enters the social security number of the new client
→Candidate responsibility: user input (''social security number of the new client'')
…

Fig. 2. Grouping of identified responsibilities (partial, from [8]).

The manager generates (1) a request for the menu of tasks that flows to the system (2), where it is processed (3). This triggers (4) creating such a menu (5) that flows to the manager (6). The manager processes (7 – e.g., browses) the menu and triggers the creation (8) of a selection ("Add New Client") that flows to the system (9). The system processes the selection and sends the blank form to the manager (10). The manager enters the social security number of the new client, which flows to the system (11).

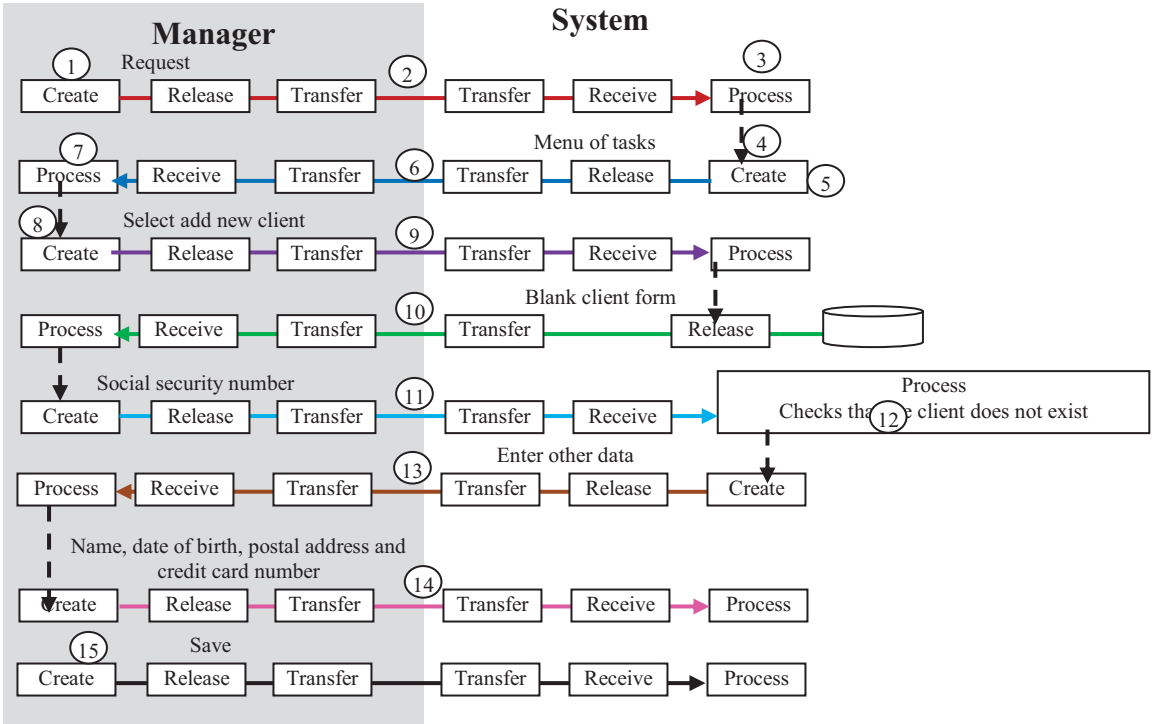


Fig. 3. FM representation of "add new client"

The system processes the input data to check that the client does not exist (12). It also sends a message to enter the other data related to the new client (13). Accordingly, name, date of birth, postal address, and credit card number are input and flow to the system (14). Then, the manager saves the changes (15).

The FM diagram is almost a flowchart of a student assignment in a programming course. Accordingly, moving to the design level to identify software components and procedures is a straightforward task. The FM representation can be extended to the user's interface level. For example, assuming that the manger wants to perform the *Add New Client* task, Figure 4 shows a partial view of such an interface with FM utilized at this level.

3.2 Use case mapping in object-oriented modeling

Object-Z is a language that facilitates the specification of systems in an object-oriented style [21]. Dongmo and van der Poll [14] developed a framework to transform a UCM into Object-Z to facilitate the construction of a formal specification.

A correct Z specification could be used in a reverse-engineering approach that would in turn enhance the original UCM. A more correct UCM may be vital, since system designers may prefer to develop a system from a set of UCMs instead of a formal description. [14]

Dongmo and van der Poll [14] describe a case study to illustrate their ideas, as follows.

Consider a scenario comprising the following activities:
(1) A company employee (say, Helper) to whom a customer returns a purchased item receives the item and temporarily puts it aside, waiting to forward it to the provider.
(2) The Helper forwards the returned item to the provider.
(3) The provider collects the returned item.

Figure 5 shows the corresponding UCM model.

> *The diagram is constructed with UCM elements that may be grouped into the categories Paths, Path elements, Path connectors, and abstract components. A UCM path indicates the progression route of one or more scenarios in execution. ... Elements encountered along a path segment are: Responsibility points, Path connectors, Timers, etc.* [14]

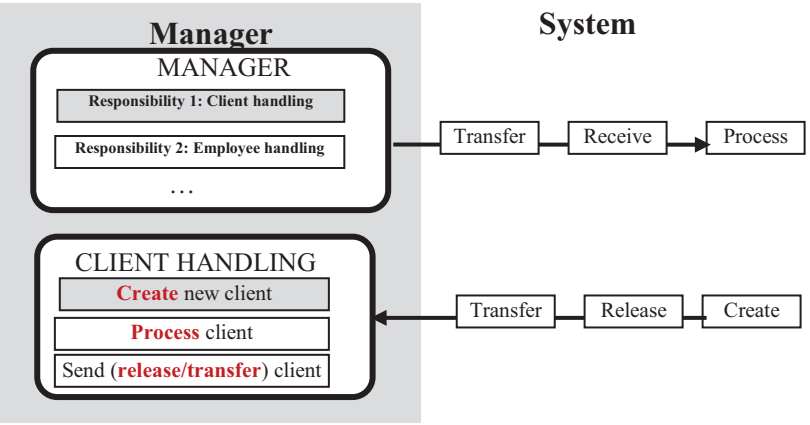


Fig. 4. A glimpse at the design level where FM is used at manager interfaces, with stages create, process, and release/transfer

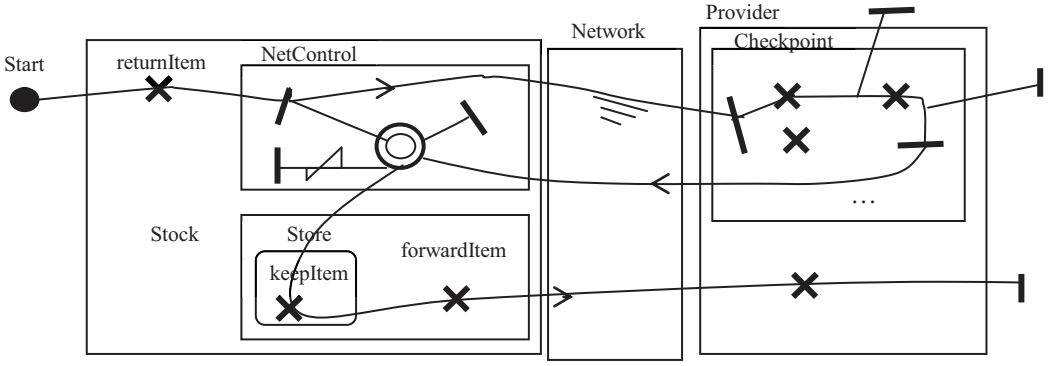


Fig. 5. Partial view redrawn from [14] that illustrates diagramming of the given scenario

Accordingly, the resultant Object-Z classes are identified as shown in the hierarchical structuring of schemata depicted in Figure 6. The arrows pointing to a class indicate the classes originating from that class.

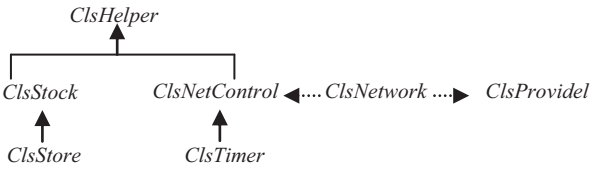


Fig. 6. Structure of classes (partial, redrawn from [14])

Such a description realizes our aim of showing the type of diagramming methodology and some of the results of the work of Dongmo and van der Poll [14]. We also contrast it side by side with a model of the same scenario using FM.

Figure 7 shows the FM depiction of our understanding of this case study. If some details of the scenario have been misunderstood, the general diagramming methodology is not affected; the figure can easily be modified.

In the figure, first, the *returned item* is received by the helper (1) and processed (2). Such processing triggers the following:

- Creation of a request to validate the item invoice (3)
- Creation of an Internet request to initiate communication (4)

The request to communicate flows to the network (5), where it is processed (6) to create a permission (7) that flows to the helper (8).

Upon receiving such permission, the helper releases the request to validate the item invoice (9) that flows to the network (10) and is delivered to the provider (11). Processing the request (12) creates a response (13) that we assume is OK; this response flows to the network (14) and then to the helper (15).

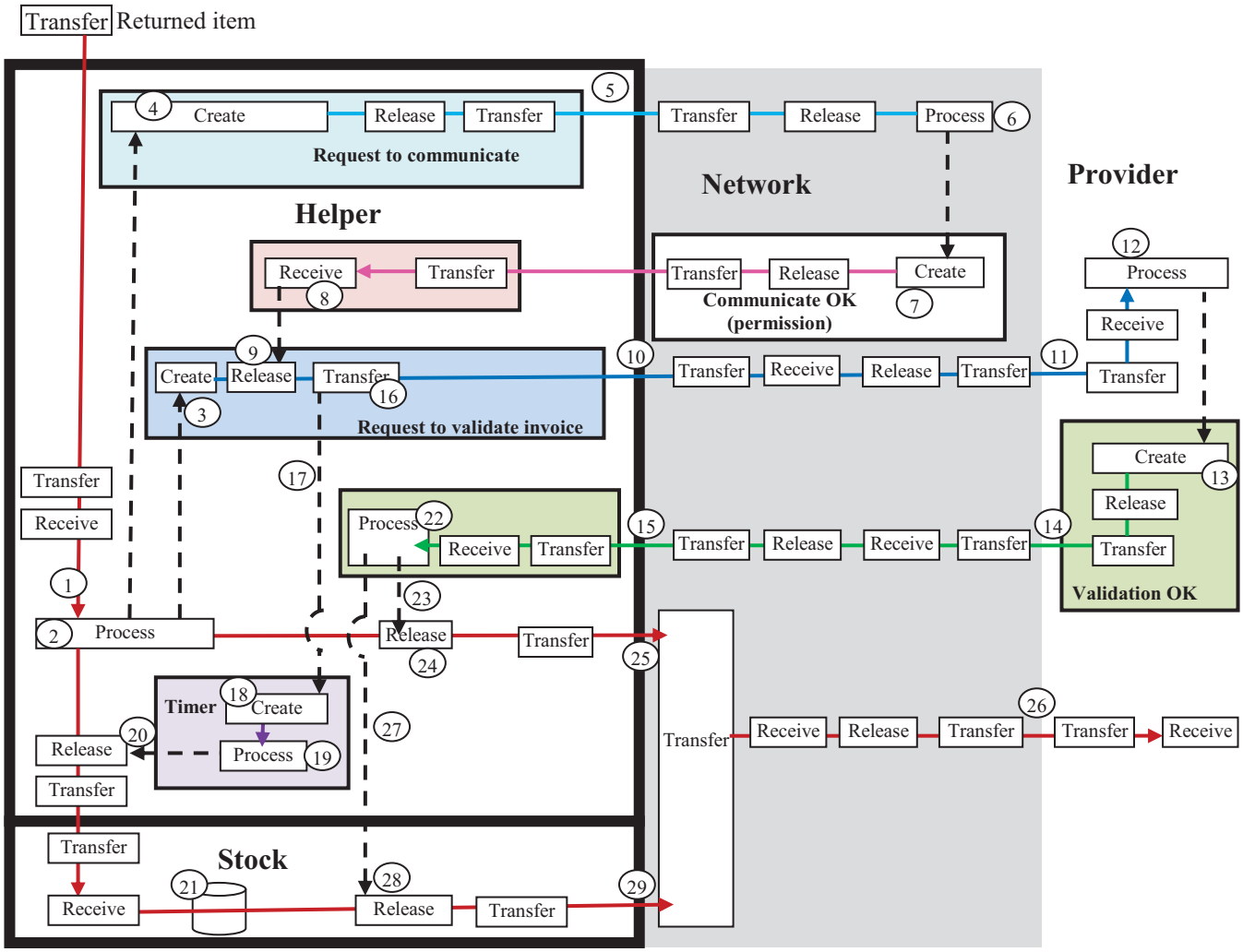


Fig. 7. FM representation of the case study

Returning to the point when a request to *validate the item invoice* (16) is sent, this triggers (17) the generation of that point in time (18) at which the time is processed (19); if this time has expired, the returned item is released (20) to be returned to stock (21).

Now we return to the point where the response *validation OK* is received by the helper (15). It is processed (22), and depending on whether or not the item is in the store, the following occurs:

- If there is no delay (the item is not in stock), then the item is triggered (23) to be released (24) to the network (25), then to the provider (26).
- It the item is in storage (delayed), then this triggers (27) releasing it (28) from storage to the network (29), then to the provider (26).

Again, the resultant description is a high-level flowchart that can be translated into a design schema. Figure 8 shows the identifying structure of classes shown in Figure 6.

4 Conclusion

This paper contributes to the area of requirements analysis and design by adopting a new modeling method that can alleviate some of the problems of lack of user input and incomplete requirements specifications. The approach advocates developing requirements analysis and design upon a *thing-oriented* foundation that integrates a diagrammatic representation upon which details of function, structure, and behavior can be built. The results point to the feasibility of using this type of modeling as a multilevel diagrammatic language with simple notions. Its features suggest that it is also appropriate for the *design* phase. Its apparent complexity can be attributed to the details of the specification as applied in engineering schemata.

Further research would expand the application of the approach to various theoretical and actual systems. Furthermore, research could lead to injecting other formalized modeling techniques (e.g., Petri net) into the method.

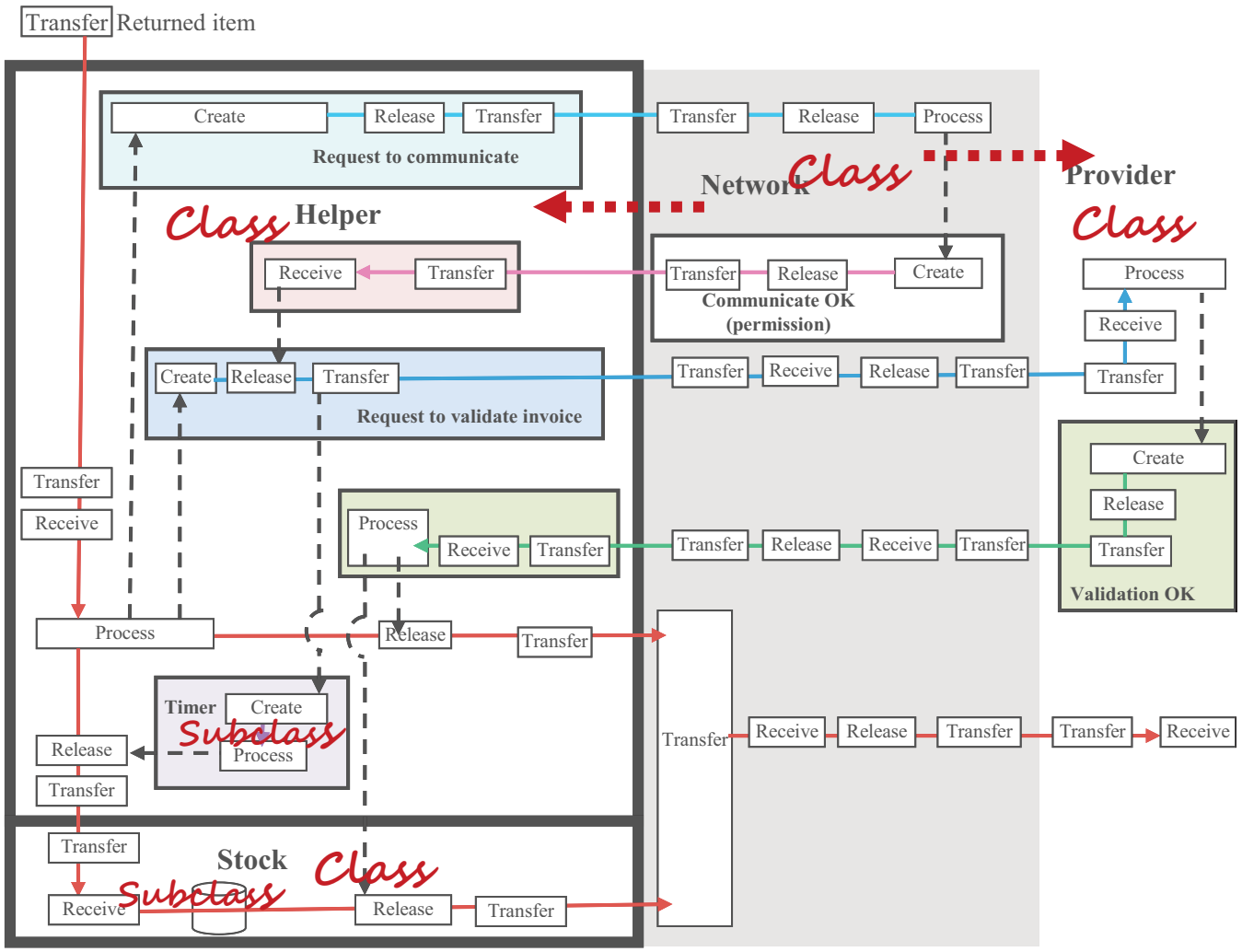


Fig. 8. Identifying structure of classes of Figure 6 in the FM representation

5 References

[1] M. Liu. "What Makes IT Projects Succeed?", Keep Touch News Letter.
http://www.csb.gov.hk/hkgcsb/eon/183/183_5.html

[2] Standish Group. "Chaos", The Standish Group Report, (2014).

[3] Y. Yang, F. Xia, W. Zhang, X. Xiao, Y. Li, X. Li. "Towards semantic requirement engineering", in: Proceedings of the IEEE International Workshop on Semantic Computing and Systems (WSCS'08), Washington, DC, USA, 2008, pp. 67–71.

[4] E. Hull, K. Jackson, D. Jeremy. "Requirements Engineering". Springer-Verlag, 2005.

[5] P. Grünbacher, A. Egyed, N. Medvidovic. "Refinement and evolution issues in bridging requirements and architecture – the CBSP approach", in: Proceedings of the 1st International Workshop, From Software Requirements to Architectures (STRAW'01), 2001, pp. 42–47.

[6] P. Grünbacher, A. Egyed, N. Medvidovic. "Reconciling software requirements and architectures with intermediate models"; Software and System Modeling, Vol. 3, No. 3, 235–253, 2006.

[7] J. E. Robbins, D. F. Redmiles. "Software architecture critics in the Argo design environment"; Knowledge-Based Systems, Vol. 11, No. 1, 47–60, 1998.

[8] Agustin Casamayor, Daniela Godoy, Marcelo Campo. "Functional grouping of natural language requirements for assistance in architectural software design"; Knowledge-Based Systems, Vol. 30, 78–86, June 2012. doi:10.1016/j.knosys.2011.12.009

[9] D. Amyot, G. Mussbacher. "Bridging the requirements/ design gap in dynamic systems with use case maps (UCMs)", in: Proceedings of the 23rd International Conference on Software Engineering (ICSE'01), Toronto, Canada, 2001, pp. 743–744.

[10] G. Mussbacher, D. Amyot, M. Weiss, "Visualizing early aspects with use case maps"; Transactions on Aspect-Oriented Software Development, vol. 3, Springer, pp. 105–143, 2007.

[11] D. Amyot, A. Eberlein. "An Evaluation of Scenario Notations for Telecommunication Systems Development", Proceedings of the Ninth International Conference on Telecommunication Systems (9ICTS), Mar. 2001.

[12] R. J. A. Buhr. "Use case maps as architectural entities for complex systems"; IEEE Transactions on Software Engineering, Vol. 24, No. 12, 1131–1155, 1998.

[13] Daniel Amyot. "On the Extension of UML with Use Case Maps Concepts"; 3rd International Conference on the Unified Modeling Language, York, UK, October 2000. Lecture Notes in Computer Science, No. 1939, pp. 16-31

[14] Cyrille Dongmo, John Andrew van der Poll. "Addressing the Construction of Z and Object-Z with Use Case Maps (UCMs)"; International Journal of Software Engineering and Knowledge Engineering, Vol. 24, No. 2, 285–327, 2014.

[15] R. Frigg. Models and representation: why structures are not enough. Measurement in Physics and Economics Project Discussion Paper Series, DP MEAS 25/02, London School of Economics (2002).

[16] Sabah Al-Fedaghi. "Scrutinizing UML Activity Diagrams"; 17th International Conference on Information Systems Development (ISD2008), Paphos, Cyprus, August 25-27, 2008.

[17] Sabah Al-Fedaghi, "Privacy as a Base for Confidentiality", Presented at the Fourth Workshop on the Economics of Information Security, Harvard University, Cambridge, MA, 2005.

[18] Sabah Al-Fedaghi, "Scrutinizing the Rule: Privacy Realization in HIPAA"; International Journal of Healthcare Information Systems and Informatics (IJHISI), Vol. 3, No. 2, 32-47, 2008.

[19] Sabah Al-Fedaghi. "Conceptualizing Effects, and Uses of Information", The Information Seeking in Context Conference (ISIC 2008), September 17-20, 2008, Vilnius. Lithuania.

[20] Sabah Al-Fedaghi. "Awareness of Context and Privacy"; The American Society for Information Science & Technology (ASIS&T) Bulletin, Vol. 38, No. 2, 2011.

[21] John Derrick, Eerke A. Boiten. "Refinement in Z and Object-Z", Springer-Verlag London, 2014.

Merging Software Specifications Focusing on Different System Boundaries

Lan Lin, Yufeng Xue
Ball State University
Department of Computer Science
Muncie, IN 47396, USA
{llin4, yxue2}@bsu.edu

Abstract

Field application of sequence-based software specification has earlier identified the need to address complexity and scalability both theoretically and practically for larger and more complex applications. Previous strategies focus on a clean partitioning of system inputs as dictated by the application, which is not always easy or possible. In this paper we present a formal and systematic process and workflow to merge partial specification products that focus on different system boundaries. The subsets of inputs entailed in the partial work products may or may not interact (or communicate) with each other, as opposed to the previous clean decomposition. The new approach will prove useful and effective in field applications towards the construction of a complete system model that forms a basis for system level software specification, testing, and certification. We report five case studies (all the applications are from published literature) that we have performed to test out our new theory and implementation.

Keywords: *rigorous software specification, sequence-based specification, requirements engineering, change management, Mealy machine*

1 Introduction

Modern software development processes for safety- and mission-critical systems rely on rigorous methods for code development and testing to support dependability claims and assurance cases [15] that provide the justified and needed confidence. *Sequence-based software specification* [26, 28, 25], as a rigorous method for requirements analysis and specification development developed by the University of Tennessee Software Quality Research Laboratory (UTK SQRL) in the 90's, derives a formal model of a software-intensive system from the original descriptions of functional requirements through a systematic and constructive *sequence enumeration* process. Since its inception it has been successfully applied to a variety of industry and government projects ranging from medical devices to automotive components to scientific instrumentation, to name a few [7, 6, 14, 27, 8, 28].

Field application of the method has identified one problem common to all aspects of computing: scalability. To reduce the specification to a manageable size, a human specifier has to clarify the extent to which inputs can be partitioned into subsets that do not interact (or communicate), and therefore, need not be enumerated together, as suggested by the application. However, this is not always easy or possible. This paper proposes a formal and systematic merging process and workflow with tool support, to overcome this limitation and combine specifications built on subsets of inputs that focus on different system boundaries. These subsets of inputs only need to cover but not necessarily partition the complete set of inputs; they may or may not have interactions with each other. The proposed workflow improves on current strategies that handle complexity and scalability, and will prove useful in field applications.

This paper is structured as follows. In Section 2 we introduce the sequence-based specification method. Section 3 elaborates the merging process and workflow with a symbolic example. In Section 4 we report five case studies in which we applied the merging process to the derivations of complete system specifications with our observations. Section 5 summarizes related work. Finally, Section 6 concludes the paper.

2 Sequence-based software specification

Sequence-based software specification [26, 28, 25] is a rigorous method that systematically converts ordinary functional requirements of software to a precise specification. The specification automatically translates to a formal system model for both development and testing.

To apply the method, one first identifies a *system boundary* that defines what is inside and outside the software-

intensive system. This usually consists of a list of *interfaces* between the system and the *environment* of the software. From the interfaces one further collects stimuli (inputs) and responses (outputs). *Stimuli* refer to events (inputs, interrupts, invocations) in the environment that can affect system behavior. *Responses* refer to system behaviors observable in the environment. Then one explicitly *enumerates* finite stimulus sequences (representing scenarios of use), first in increasing order of length, and within the same length lexicographically. The sequence enumeration process is guided by a few rules.

As an example let us consider a simple industrial "cooker" [2]. Requirements for the cooker controller are collected in Table 1, with each sentence numbered (tagged) for easy reference. The last three requirements whose tags begin with "D" correspond to derived requirements not originally given but identified in the specification process. We further identify all stimuli and responses in Table 2. Notice the last two responses introduced by the theory: the *null* response (denoted 0) representing no observable behavior across the system boundary (the software may only have an internal state update), and the *illegal* response (denoted ω) representing a sequence of inputs not practically realizable (an instance of this is the power-on event being placed after other inputs in the sequence).

Table 3 shows a sequence enumeration of the cooker controller up to Length 3. We start with the empty sequence λ, and proceed from Length n to Length $n+1$ sequences (n is a non-negative integer). Within the same length we enumerate sequences in lexicographical order. For each enumerated sequence the human specifier makes two decisions based on the requirements:

1. They identify a unique response for the most current stimulus given the complete stimulus history. For instance, the sequence SG corresponds to: software started, followed by a pressure good reading. By Requirements 1, 2, 3, and 5, the valve should be functioned first and closed, hence the response "cv". A sequence is *illegal* if its mapped response is ω; otherwise, it is *legal*.

2. They consider if the sequence takes the system to the same situation that has been encountered and explored by a previously enumerated sequence. If so they note the previous sequence in the "Equivalence" column, and treat the later sequence as *reduced* to the earlier sequence. The two sequences are called *Mealy equivalent* as they correspond to the same state when the system is modeled as a Mealy machine. For instance, the sequence SO corresponds to: software started and then turned off. This takes the system to the same situation as the empty sequence as in both cases software is not started yet, hence SO is reduced to λ. From this

Table 1. Cooker controller requirements

Tag	Requirement
0	Consider an industrial "cooker" that is subject to various factors (external heat, internal chemical reactions, etc.) that change the pressure in the cooker. A control unit attempts to keep the pressure near a specific set point by opening and closing a pressure release valve and turning a compressor on and off, according to the following rules.
1	When started, the controller does not know the status of the valve or compressor, and there is no hardware capability to poll their status.
2	Only one output signal can be sent per input signal.
3	If both the valve and the compressor appear to need changes at the same time, since only one signal can be sent, function the valve first.
4	If the input signal says the pressure is Low, then the valve should be closed and the compressor on.
5	If the input signal says the pressure is Good, then the valve should be closed and the compressor off.
6	If the input signal says the pressure is High, then the valve should be open and the compressor off.
7	When turned off, the controller does not have time to generate any output signal.
D1	Sequences with stimuli prior to system initialization are illegal by system definition.
D2	When started, the controller does not produce any externally observable response across the system boundary.
D3	Once started, the system cannot be started again without being turned off first.

Table 2. Cooker controller stimuli and responses

Stimulus / Response	Short Name	Description	Requirement Trace
Stimulus	S	Started	1
Stimulus	L	Pressure low	4
Stimulus	G	Pressure good	5
Stimulus	H	Pressure high	6
Stimulus	O	Turned off	7
Response	ov	Open valve	0
Response	cv	Close valve	0
Response	co	Compressor on	0
Response	cf	Compressor off	0
Response	0	Null	Method
Response	ω	Illegal	Method

point on SO and λ will behave the same for any future non-empty sequence of inputs. A sequence is *unreduced* if no prior sequence is declared for its "Equivalence" column; otherwise, it is *reduced*. In theory we also treat an unreduced sequence as reduced to itself.

The enumeration process is constrained by the following rules:

1. If a sequence (say u) is illegal, there is no need to extend u by any stimulus, as all of the extensions must be illegal (i.e., physically unrealizable). For instance, no extensions of the sequence G are enumerated because G is an illegal sequence.

2. If a sequence (say u) is reduced to a prior sequence (say v), there is no need to extend u, as the behaviors of the extensions are defined by the same extensions of v. For instance, no extensions of the sequence SO are enumerated because SO is reduced to λ.

3. When reducing a sequence (say u) to a sequence (say v), we require v be both prior (in length-lexicographical order) and unreduced (otherwise we could follow the reduction chain of v and get to the sequence that is unreduced). For instance, SGO is reduced to λ and not SO.

Therefore, only legal and unreduced sequences of Length n get extended by every stimulus for consideration at Length $n+1$. The process continues until all sequences of a certain length are either illegal or reduced to prior sequences. The enumeration becomes *complete*. This terminating length is discovered in enumeration, and varies from application to application. The cooker controller enumeration terminates at Length 5.

Application of the method is facilitated with two tools developed by UTK SQRL: Proto_Seq [3] and REAL [4]. To produce a specification in either tool, one only needs to give stimuli and responses short names to facilitate enumeration; no other notation or syntax is required. The tools enforce enumeration rules by the recommended workflow and maintain internal files (XML format) current with every action.

Table 3. Cooker controller sequence enumeration up to Length 3

Sequence	Response	Equivalence	Trace
λ	0		Method
G	ω		D1
H	ω		D1
L	ω		D1
O	ω		D1
S	0		D2
SG	cv		1, 2, 3, 5
SH	ov		1, 2, 3, 6
SL	cv	SG	1, 2, 3, 4
SO	0	λ	7
SS	ω		D3
SGG	cf		5
SGH	ov	SH	2, 3, 6
SGL	co		4
SGO	0	λ	7
SGS	ω		D3
SHG	cv	SG	2, 3, 5
SHH	cf		6
SHL	cv	SG	2, 3, 4
SHO	0	λ	7
SHS	ω		D3

3 Merging specifications

In sequence-based specification the derived specification takes the form of a complete sequence enumeration. An enumeration is *complete* if every *legal and unreduced* (also called *canonical*) sequence has been extended by every single stimulus. Let us consider merging two enumerations focusing on different system boundaries: $\mathcal{E}_1$ with the stimulus set S_1, and $\mathcal{E}_2$ with the stimulus set S_2, where $S_1 \cup S_2 = S$ (the complete stimulus set). Notice that the two stimulus sets S_1 and S_2 do not have to be disjoint (if they are, they correspond to a partitioning of the complete set of system inputs), and can share a few stimuli that are common to the two different system boundaries (for example the power on input). Let C_1 and C_2 denote the sets of canonical sequences in $\mathcal{E}_1$ and $\mathcal{E}_2$, respectively. We define the process for merging in Table 4, with the following assumptions:

- If $\mathcal{E}_1$ and $\mathcal{E}_2$ share a stimulus, it has the same definition in both.

- The same holds for a stimulus sequence that is defined in both $\mathcal{E}_1$ and $\mathcal{E}_2$.

When merging two specifications that focus on different system boundaries new enumeration entries (stimulus sequences) will need to be added as a result of the enlarged stimulus set (Steps 2-3 in Table 4). These sequences entail interactions of stimuli from the different system boundaries, and need to be defined for the correct system behavior based on the requirements. Since these sequences are inserted into all possible places in an ordered enumeration table, defining these entries requires the application of change algorithms (Step 6) to automatically compute and enforce the rippling effect of a single response or equivalence change.

Since the sub-specifications were constructed based on

Table 4. The process for merging two complete enumerations $\mathcal{E}_1$ and $\mathcal{E}_2$, with the stimulus sets S_1 and S_2, and the canonical sequence sets C_1 and C_2, respectively; (H) indicates the step requires human effort; (T) indicates the step has automated tool support.

Step	Description
1	Construct $\mathcal{E}$ by taking rows of $\mathcal{E}_1$ and $\mathcal{E}_2$ and re-ordering them based on the length-lexicographical ordering of stimulus sequences. (T)
2	Extend each sequence in $C_1 - C_2$ by each stimulus in $S_2 - S_1$. Map the new extensions to illegal and highlight these rows (as a reminder to the human specifier that they need to be re-defined based on the requirements). The new extensions are inserted in $\mathcal{E}$ based on the length-lexicographical ordering of stimulus sequences. (T)
3	Extend each sequence in $C_2 - C_1$ by each stimulus in $S_1 - S_2$. Map the new extensions to illegal and highlight these rows (as a reminder to the human specifier that they need to be re-defined based on the requirements). The new extensions are inserted in $\mathcal{E}$ based on the length-lexicographical ordering of stimulus sequences. (T)
4	Check if two canonical sequences in $\mathcal{E}$ are Mealy equivalent (i.e., check for redundant states). (H) If so, reduce the later in length-lexicographical order to the prior one using an equivalence change algorithm (i.e., eliminate redundant states). (T)
5	Check if a previously reduced sequence in $\mathcal{E}$ should be declared unreduced considering the combined stimulus set S (i.e., check if any state needs to be split). (H) If so, declare the sequence as unreduced using an equivalence change algorithm (i.e., split the state; essentially the sequence will be extended by every stimulus in S; the new extensions will be mapped to illegal and highlighted, and re-defined by the human specifier based on the requirements). (T)
6	Define all highlighted rows one after another in the order they appear in $\mathcal{E}$ based on the requirements. Essentially each highlighted sequence starts from the default definition of being illegal, and proceeds as follows: (H/T)
6.1	If the sequence is indeed illegal, only fill in the requirements trace column. (H/T)
6.2	If the sequence is legal and reduced, define it as so using a response change algorithm followed by an equivalence change algorithm, and fill in the requirements trace column. (H/T)
6.3	If the sequence is legal and unreduced, define the new response using a response change algorithm. Check if the sequence is Mealy equivalent to a sequence that comes later in length-lexicographical order. If so define the equivalence between these two sequences using a forward reduction algorithm (another case of eliminating redundant states; another change algorithm), and fill in the requirements trace column. (H/T)

a partial system boundary, when they are combined, it is possible that the merged specification contains redundant states (Step 4) or states that need to be split considering an enlarged stimulus set (Step 5). In such cases change algorithms are also needed to automatically enforce all the comprehensive impact of a single response or equivalence change.

All the response and equivalence change algorithms have been developed [18] using state machines as an intermediate and visualizing tool to comprehend and analyze the impact of changes. More recently the forward reduction algorithm used in Step 6.3 was developed [20] along the same line. Derivation of the algorithms is out of the scope of this paper (for details please see [20, 18]). In all the cases with one intended response or equivalence definition of a specific stimulus sequence, the algorithms compute and apply all the changes that need to be incurred to the specification and guarantee it still conforms to all the enumeration rules. Table 5 describes the algorithms that facilitate the merging process. They were all developed using an axiom system for sequence-based specification [20, 19, 18], proved to be correct, implemented and provided with tool support.

Figure 1 shows Steps 1-3 of the merging process with two symbolic enumerations: $\mathcal{E}_1$ with the stimulus set $S_1 = \{a, b, c\}$, the response set $R_1 = \{r_1, r_2, 0, \omega\}$, and the canonical sequence set $C_1 = \{a, c\}$, and $\mathcal{E}_2$ with the stimulus set $S_2 = \{c, d\}$, the response set $R_2 = \{r_1, r_2, r_3, 0, \omega\}$, and the canonical sequence set $C_2 = \{c, cd\}$. Three new extensions are added to the merged enumeration at the end of Step 3: sequences ad, cda, and cdb. They are all mapped to illegal by default, and highlighted (in different shades of orange) as a reminder to the human specifier for re-definition. Suppose there are no redundant states after the merge (i.e., the two sequences a and cd are not Mealy equivalent), the merged enumeration remains the same after Step 4. Suppose a previously reduced sequence aa needs to be re-considered for state splitting, an equivalence change algorithm will need to be called on aa to make it unreduced (Step 5). In defining the three new sequences if for instance the sequence ad is found to be Mealy equivalent to the future sequence cd which is already extended, the forward reduction algorithm will need to be called to define the extensions of ad, as well as their future extensions if needed, based on the extensions of cd, enforce all the change impacts, and reduce cd to ad in the resulting specification (Step 6.3).

We have implemented the merging behavior and the forward reduction algorithm in our enumeration tool Proto_Seq [3] that runs on Windows, Linux, and Mac OS X. In Table 4 we specify for each step of the workflow whether it requires human effort, and/or has automated tool support.

Table 5. Algorithms that facilitate the merging process

Algorithm	Description
Equivalence change: for a legal and unreduced sequence	Make a previously legal and unreduced (hence extended) sequence reduced, and make all corresponding changes to the resulting enumeration.
Equivalence change: for a legal and reduced sequence; make it unreduced	Make a previously legal and reduced sequence unreduced, and make all corresponding changes to the resulting enumeration.
Response change: from illegal to legal	Make a previously illegal sequence legal, and make all corresponding changes to the resulting enumeration.
Forward reduction	Simulate the effect of "forward reducing" a prior legal and unreduced sequence to a later legal and unreduced sequence in length-lexicographical order, and make all corresponding changes to the resulting enumeration. (In the enumeration after the change all reductions are still to prior sequences by the enumeration rules.)

4 Case studies

We performed case studies on five published applications:

- An automotive lighting system (ALS) [1]
- A car door mirror electronic control unit (CDMECU) [6]
- An insulin pump controller software (IPC) [29]
- A mine pump controller software (MPC) [17]
- A piece of the satellite operations software (SOS) [28]

In each case study we applied sequence-based specification on two subsets of stimuli focusing on different system boundaries that cover the complete stimulus set for each application, and merged the specifications using the theory we have developed and the tool support.

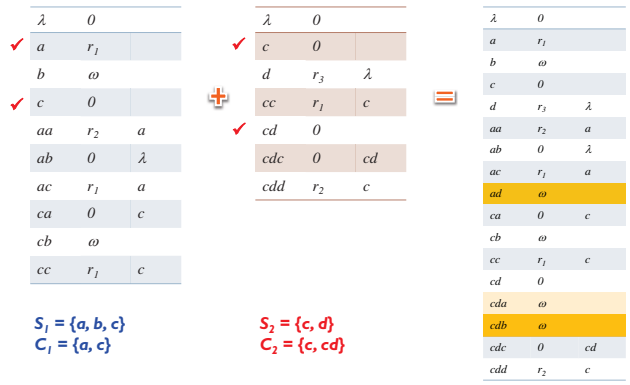


Figure 1. An example of merging two symbolic enumerations (table columns are for sequences, responses, and equivalences, respectively)

Figure 2 shows the enumeration statistics for the five case studies before and after the merge. For each of the two enumerations (sub-specifications) before the merge, we recorded the number of stimuli, the termination enumeration length (i.e., at which length the enumeration / sub-specification terminates with completion), how many stimulus sequences are extended (i.e., the number of both legal and unreduced sequences in the enumeration / sub-specification), and how many stimulus sequences are analyzed (i.e., the total number of enumerated sequences in the enumeration / sub-specification). For the enumeration (specification) after the merge (after the consolidation of redundant states but before state splitting and the definition of any highlighted new extensions) we recorded the number of stimulus sequences that need to be defined at the top level (which might get further extended). Merging works correctly as expected for all the case studies.

Figure 3 details for each case study how we divided up the system boundary (to determine the inputs that need to be included in each sub-specification). Selection of the system boundary for sub-specifications is in all cases intuitive and application-specific. It also lists for each case study in the merging process whether there is any consolidation of redundant states, state splitting, and application of forward reductions, and if so how much is handled by the built-in change algorithms. These steps in the merging process work correctly as expected for all the case studies, although the need for such considerations depends on the application.

Figure 4 shows the final enumeration statistics for each

Application		ALS	CDMECU	IPC	MPC	SOS
First enum	# of stimuli	7	8	9	4	15
	Term. enum length	4	4	6	2	6
	Seqs extended	7	6	9	2	7
	Seqs analyzed	50	57	85	9	106
Second enum	# of stimuli	3	7	4	6	8
	Term. enum length	5	5	3	5	1
	Seqs extended	5	5	3	16	1
	Seqs analyzed	16	36	13	129	9
Merged enum	# of first-level new extensions	5*	45	21	66	48

*After consolidation of redundant states

Figure 2. Enumeration statistics for the five case studies before and after the merge

Application	ALS	CDMECU	IPC	MPC	SOS
First enumeration	Battery inputs, dashboard inputs	Ignition, mirror movement inputs	Switch on/off, auto mode inputs	Human (operation or supervisor) inputs	GCS (Ground Control System) commands, on-board system signals
Second enumeration	Battery inputs, steering wheel inputs	Engine start, mirror heating inputs	Switch on/off, manual mode inputs	Sensor inputs	Uplink inputs, downlink inputs
Consolidation of redundant states	Yes (3 pairs of redundant states / 1 equivalence change)	No	No	No	No
State splitting	No	No	No	No	No
Forward reductions	No	No	No	Yes (2 forward reductions)	No

Figure 3. Details on system boundary selection, consolidation of redundant states, state splitting, and forward reductions for the five case studies

case study after we defined all the new extensions, as well as their further extensions if needed, based on the requirements. For each completed final specification we recorded the number of stimuli, the termination enumeration length, how many stimulus sequences are extended, how many stimulus sequences are analyzed, as compared with how many stimulus sequences potentially need to be considered up to the termination enumeration length. It can be observed that sequence enumeration is effective in controlling the combinatorial growth of the number of input sequences to be examined in order to specify the correct system behavior and construct a formal system model for design, implementation, and testing.

5 Related work

Sequence-based specification emerged from the functional treatment of software as described by Mills [23, 21, 22]. The development was most directly influenced by the

Application	ALS	CDMECU	IPC	MPC	SOS
Number of stimuli	8	14	11	10	23
Termination enumeration length	4	7	7	5	9
Sequences extended	7	21	17	22	11
Sequences analyzed	57	326	191	265	254
Potential sequences	4,681	113,522,235	21,435,888	111,111	1.8830232E12

Figure 4. Enumeration statistics for the final specifications for the five case studies

trace-assertion method of Parnas [24, 5] and the algebraic treatment of regular expressions by Brzozowski [9]. Foundations of the sequence-based specification method were established by Prowell and Poore in [26, 28, 25]. An axiom system for sequence-based specification was developed more recently [19] for a formal treatment. The axiom system was used in developing a theory and a set of algorithms that manage requirements changes and state machine changes [18]. The forward reduction algorithm used in the merging process can be added to that set [20].

The primary distinction of sequence-based specification from its nearest neighbors (the *trace-assertion method* [12, 10, 11, 16, 24, 5] and *software cost reduction* [13]) is that we evolve the discovery and invent the state machine from requirements through systematic enumeration of input sequences and recording sequence equivalences based on future behavior. Likewise, the theory underlying the forward reduction algorithm used in the merging, as well as other change algorithms, differs from the conventional state change theory in that it is designed for human interactive sequence enumeration and revision, and targeted at an appropriate subset of Mealy machines that can be obtained through enumeration.

6 Conclusion and future work

Field application of sequence-based software specification has earlier identified the need to address complexity and scalability both theoretically and practically for larger and more complex applications. This paper proposes a merging process and workflow with automated tool support that combines specifications focusing on different system boundaries. The subsets of inputs considered in smaller specifications may or may not have interactions with each other, as opposed to a clean partitioning of system inputs (dictated by the application) by the previous strategy. We present a formal and systematic process to explicitly merge partial work products towards a complete system model, forming a basis for system level software specification, test-

ing, and certification. We report five case studies (all the applications are from published literature) that we have performed to test out our new theory and implementation.

Future work is along the line of managing complexity and scalability for larger and more complex systems. Work is under way that explores an alternative, orthogonal approach by limiting the number of states being explored in partial work products.

Acknowledgements

The authors would like to thank Tom Swain (previous manager of UTK SQRL) for helping set up an environment for new development and testing, and Xin Guo for earlier code contribution. This work was generously funded by Rockwell Collins, Air Force Research Laboratory, and Ontario Systems through the NSF Security and Software Engineering Research Center (S^2ERC).

References

[1] 2007. David L. Parnas. Trace function method: Exercise.

[2] 2010. Jesse H. Poore. Private communication.

[3] 2016. Prototype Sequence Enumeration (Proto_Seq). Software Quality Research Laboratory, The University of Tennessee. https://sourceforge.net/projects/protoseq/.

[4] 2016. Requirements Elicitation and Analysis with Sequence-Based Specification (REALSBS). Software Quality Research Laboratory, The University of Tennessee. http://realsbs.sourceforge.net.

[5] W. Bartussek and D. L. Parnas. Using assertions about traces to write abstract specifications for software modules. In *Proceedings of the 2nd Conference of the European Cooperation on Informatics*, pages 211–236, Venice, Italy, 1978.

[6] T. Bauer, T. Beletski, F. Boehr, R. Eschbach, D. Landmann, and J. Poore. From requirements to statistical testing of embedded systems. In *Proceedings of the 4th International Workshop on Software Engineering for Automotive Systems*, pages 3–9, Minneapolis, MN, 2007.

[7] L. Bouwmeester, G. H. Broadfoot, and P. J. Hopcroft. Compliance test framework. In *Proceedings of the 2nd Workshop on Model-Based Testing in Practice*, pages 97–106, Enscede, The Netherlands, 2009.

[8] G. H. Broadfoot and P. J. Broadfoot. Academia and industry meet: Some experiences of formal methods in practice. In *Proceedings of the 10th Asia-Pacific Software Engineering Conference*, pages 49–59, Chiang Mai, Thailand, 2003.

[9] J. Brzozowski. Derivatives of regular expressions. *Journal of the ACM*, 11(4):481–494, 1964.

[10] J. Brzozowski. Representation of a class of nondeterministic semiautomata by canonical words. *Theoretical Computer Science*, 356:46–57, 2006.

[11] J. Brzozowski and H. Jürgensen. Representation of semiautomata by canonical words and equivalences. *International Journal of Foundations of Computer Science*, 16(5):831–850, 2005.

[12] J. Brzozowski and H. Jurgensen. Representation of semiautomata by canonical words and equivalences, part II: Specification of software modules. *International Journal of Foundations of Computer Science*, 18(5):1065–1087, 2007.

[13] C. L. Heitmeyer. Software cost reduction. In J. J. Marciniak, editor, *Encyclopedia of Software Engineering*. Wiley-Interscience, 2001.

[14] P. J. Hopcroft and G. H. Broadfoot. Combining the box structure development method and CSP for software development. *Electronic Notes in Theoretical Computer Science*, 128(6):127–144, 2005.

[15] D. Jackson, M. Thomas, and L. I. Millett, editors. *Software for Dependable Systems: Sufficient Evidence?* National Academies Press, 2007.

[16] R. Janicki and E. Sekerinski. Foundations of the trace assertion method of module interface specification. *IEEE Transactions on Software Engineering*, 27(7):577–598, 2001.

[17] M. Joseph, editor. *Real-Time Systems: Specification, Verification and Analysis*. Prentice Hall International, London, United Kingdom, 1996.

[18] L. Lin, S. J. Prowell, and J. H. Poore. The impact of requirements changes on specifications and state machines. *Software: Practice and Experience*, 39(6):573–610, 2009.

[19] L. Lin, S. J. Prowell, and J. H. Poore. An axiom system for sequence-based specification. *Theoretical Computer Science*, 411(2):360–376, 2010.

[20] L. Lin and Y. Xue. An algorithm for forward reduction in sequence-based software specification. In *Proceedings of the 28th International Conference on Software Engineering and Knowledge Engineering*, pages 309–316, Redwood city, San Francisco Bay, CA, 2016.

[21] R. C. Linger, H. D. Mills, and B. I. Witt. *Structured Programming: Theory and Practice*. Addison-Wesley, 1979.

[22] H. D. Mills. The new math of computer programming. *Communications of the ACM*, 18(1):43–48, 1975.

[23] H. D. Mills. Stepwise refinement and verification in box-structured systems. *IEEE Computer*, 21(6):23–36, 1988.

[24] D. L. Parnas and Y. Wang. The trace assertion method of module interface specification. Technical Report 89-261, Queens University, 1989.

[25] S. J. Prowell and J. H. Poore. Sequence-based software specification of deterministic systems. *Software: Practice and Experience*, 28(3):329–344, 1998.

[26] S. J. Prowell and J. H. Poore. Foundations of sequence-based software specification. *IEEE Transactions on Software Engineering*, 29(5):417–429, 2003.

[27] S. J. Prowell and W. T. Swain. Sequence-based specification of critical software systems. In *Proceedings of the 4th American Nuclear Society International Topical Meeting on Nuclear Plant Instrumentation, Controls and Human-Machine Interface Technology*, Columbus, OH, 2004.

[28] S. J. Prowell, C. J. Trammell, R. C. Linger, and J. H. Poore. *Cleanroom Software Engineering: Technology and Process*. Addison-Wesley, Reading, MA, 1999.

[29] I. Sommerville. *Software Engineering*. Addison-Wesley, Harlow, England, 9th edition, 2010.

Requirements Engineering Practice in Developing Countries

Elicitation and Traceability Processes

Ayman Sadig[1], Abd-El-Kader Sahraoui[2]
[1]. Ahfad University for Women and SUST Khartoum Sudan
[2]. LAAS-CNRS, Université de Toulouse, CNRS, U2J, Toulouse, France

***Abstract*—**This is a preliminary work on such new research topic. The paper is on analyzing requirement process in developing countries and develop requirement management methodology and mainly on elicitation, traceability and validation. We know that requirement engineering is as stated by Goguen "requirement engineering reconciliation" from that we can say social aspect affect the requirement process. We propose an approach to deal with these issues.

***Keywords*—** requirements traceability, requirement elicitation.

1. Introduction and problem statement

The complexity in systems development is observed when linking artifacts between themselves, these artifacts items can pieces of requirements, properties, pieces of design and stakeholders. We address two issues that were used separately in previous studies. The first issue is part of requirements engineering as the first sub-process.: Requirements elicitation to make difference with the requirements acquisition.

The second concerns part of requirements managements named traceability. There are two types traceability and syntactic links between items and the more semantic based concerns the coverage that item as design does Implement the re stated requirements.

Elicitation of requirements is a long process that differs from requirement acquisition as said previously. We want to link these issues of requirements management and deploy these by adaptation in the context of developing countries. Huge project in developing country has start with only handful of requirement then writing more requirement as while working on the project. Also 'requirement cannot be gathered out of the social context' [1]

- Traceability is a new issue that reinforces the requirements elicitation as we can always trace backward and forward to search for rational. Traceability provides critical function in the development and maintenance of a software system. Projects using traceability has 21% extra success [12]. Most middle to large software companies in developed country uses traceability [11]. It is not the case in developing countries where a lot more work need to be done to convince companies that money spend in traceability is well worth it.

I am doing a poll to to contact institutions and companies how they do requirements elicitations and traceability in developing countries. Top IT company in Sudan develops products then adjusted them to several customers needs using agile method but that required extensive communication and collobration where the problems lies.

2. Requirement elicitation approach

Requirements elicitation can be broadly defined as the activities, typically performed after project initiation and before system design. They are related to the acquisition and elaboration of goals, constraints, and features for a proposed system, by means of investigation and exploration. Furthermore, it is generally understood and accepted that requirements are elicited rather than just captured or collected. This implies both a discovery and development element to the process. In practice requirements elicitation is often performed poorly, the major reasons being inadequate expertise on the part of the participating requirements engineer, and the insufficient allocation of time and resources within the larger development project. The consequences of this situation frequently include costly rework, schedule overruns, project failure, poor quality systems, and general stakeholder dissatisfaction [3]. Failing to gather the right requirement count for 90% of large software projects failures [9], "Poor requirements management lead to 71 % of software projects that fail; greater than implementation

problem, missed deadlines, and change management issues. (Lindquist,2005, p. 54)

In response, much of the relevant research performed over the past two decades has focused on the development of numerous techniques for requirements elicitation as surveyed in [1], and more recently in [8]. In developing country like brazil reason for incomplete requirement was people 40%, input 20% and issue affecting customer 19% while in Germany, people 12%, input 6% and no effect for issuers affecting customer. One of the more successful in producing quality requirements has proven to be facilitated workshops [2]. However, most projects typically require more than one technique to be used for requirements elicitation [6]. Furthermore, a major problem in requirements elicitation today is the significant gap between expert and novice analysts. A lack of awareness by analysts of the state of the art techniques and tools for requirements elicitation, combined with a general unwillingness to adopt them is largely responsible for this situation. This situation is further aggravated by the current shortage of systematic guidelines and flexible methods.

Subsequently the work described in this paper investigates how an improved approach for the early stages of requirements elicitation can be developed that combines various techniques based on a detailed information meta-model and process framework for collaborative workshops. The approach was developed based on an extensive literature review, seven structured interviews with practitioners widely regarded within the Requirements Engineering community as elicitation experts, and a review of requirements related documentation produced within fifteen successful system development projects. The paper is therefore structured as follows. Section 2 describes the information types contained in the meta-model used as the foundation of the approach. Section 3 presents the approach with an overview of the structure, content, and process. Section 4 offers a broad discussion of the approach, and finally Section 5 provides some general conclusions on the research.

3. A meta model for requirements elicitation

The foundation of the proposed approach is based on a knowledge meta-model consisting of a specified set of information types with corresponding attributes. As the name implies, information types are different types or categories of information or knowledge that must be addressed during the requirements elicitation phase of the software development lifecycle in order to collect and capture all the necessary details to produce a quality requirements specification document. As can be seen in Table 1 below, fifteen 'core' information types have been identified from the review of current theory and practice as being relevant to most application domains, and typically necessary in most software engineering projects.

Table 1: The Fifteen Core Information Types

No.	*Title*	*Description*
1	Project	Problem, mission, vision, context, and scope of the project
2	Deliverable	Desired result of the process, its audience, objectives, and overview
3	System	Background, perspective, context, and scope of the system
4	Objectives	Objectives of the business with respect to the project and system
5	Assumptions	Underlying assumptions upon which the project and system are based
6	Constraints	Constraints that must be applied to the project and system
7	Environment	Social and physical environmental characteristics of the project and system
8	Opportunities	Possible opportunities for improvements to be delivered by the system
9	Challenges	Possible challenges which may be encountered during the project
10	Risks	Potential risks to both the project and the system
11	Stakeholders	Stakeholders in the project, and sources of

		system information
12	Processes	Detailed work process which the system must support
13	Functional	Functional aspects which must be provided by the system
14	Non-functional	Non-functional aspects which must be provided by the system
15	Implementation	Implementation details relating to the system including design solutions

Information types can have multiple levels of detail, and relationships between these different information types and levels, such as linking individual non-functional requirements to system objectives, are handled by specific attributes within the template for those information types. Table 2 provides an example of a template for one of the core information types. Within the approach, an individual template has been developed for each of the information types with specific attributes and instructions

Table 2: Information Type Template Example – '4. Objectives'

Attribute	***Description***
ID	Unique numerical identifier for the objective
Name	Unique textual name for the objective
Description	Detailed description of the objective
Type	Classification of the objective selected from a standard list or specified by the analyst
Source	Source of the objective, possibly a document, a person, or an organization
Rationale	Justification for the objective in terms of reasons for its inclusion
Priority	Importance of the objective selected from a standard rating or specified by the analyst

In order to promote a more rigorous approach and resultant document from the requirements elicitation process, a number of additional information types are required to provide all the necessary support information for the knowledge elicited for the core types. These can be seen in Table 3 below.

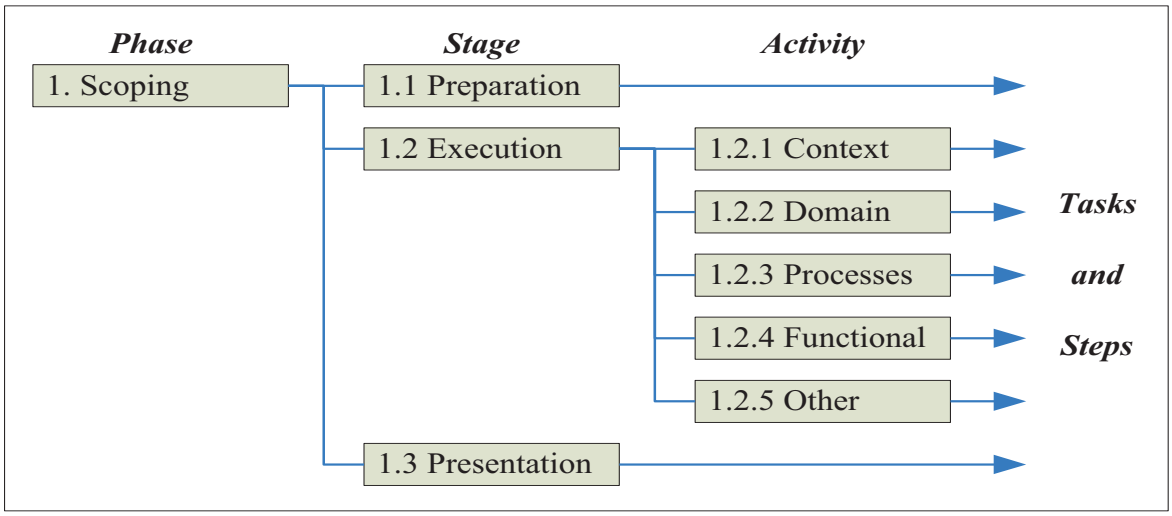
Table 3: The Seven Support Information Types

No.	***Title***	***Description***
1	Glossary	Definition of terms, abbreviations, and acronyms
2	Dictionary	Data definitions relevant to the system including type, size, and format
3	Issues	Prioritized list for project and system related issues
4	Actions	Prioritized list for project and system related actions
5	Ideas	Possible suggestions and potential solutions related to the project and system
6	References	Cited references made to information in other documents and sources
7	Appendixes	Required appendixes for the resultant document

The combination of these information types, as well as additional ones that may be specified by the requirements engineers based on the needs of the individual projects, form the information meta-model used as the foundation for the workshops and guidelines.

A guided requirements elicitation workshop

The proposed approach consists of three key workshop phases being 1) Scoping, 2) High-level, and 3) Detailed, as explained in the following sub sections. As can be seen in the example of the Scoping phase shown in Figure 1 below, the Execution stage of each phase is divided into five activities. These activities, as well as the Preparation and Presentation stages for each phase, are composed of a set of tasks in a prescribed sequence (100 tasks in total for all 3 phases). The steps for these tasks, being the next and final level in the process hierarchy, are determined by which of the techniques within the approach is selected to perform that particular task.



Structured Workshop Process Hierarchy – '1. Scoping Phase'

Each of the phases may be completed over a number of sessions depending on the complexity of the project, and the availability of the relevant stakeholders, facilitated by a requirements engineer, also referred to as the analyst. Furthermore, the same information type may be addressed by more than one task in different stages. In these cases the level of detail investigated and the attributes elicited are different but complimentary. Each task has one or more 'available' techniques, meaning that established instructions exist within the approach as a set of sequences steps for that technique, which can be utilized to perform that particular task within a workshop environment.

4. A general Traceability model

What is prone here is the separation of concerns principles, as the model can be made generic for new systems and enhanced for existing systems. The approach to be discussed is illustrated with the following figure fig1

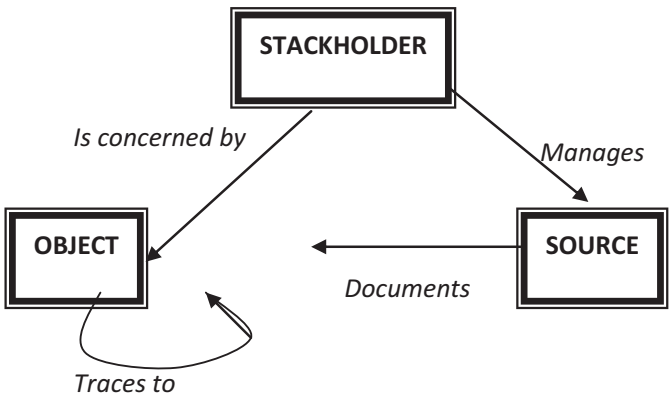
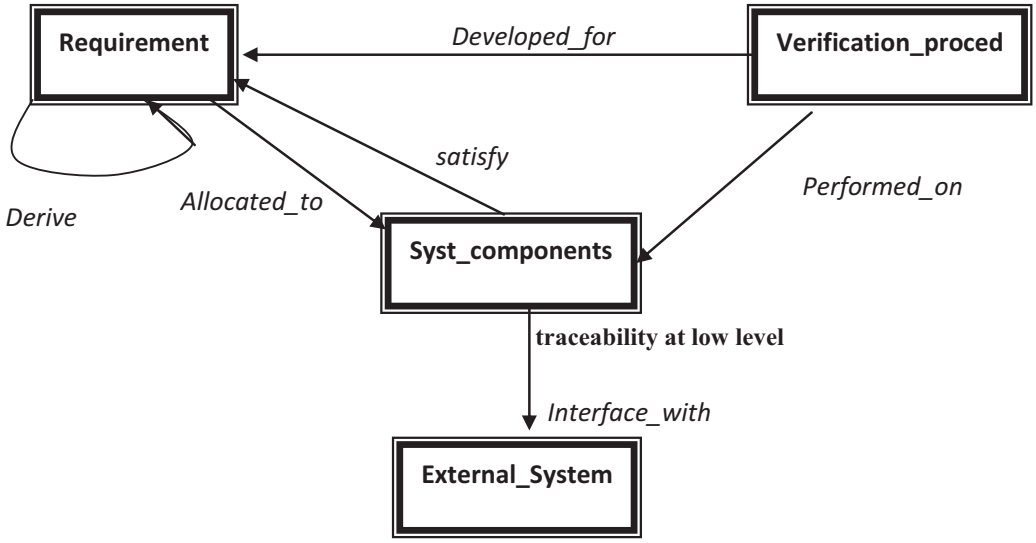


Fig1 : Traceability model

Such a reference model is discussed later in the paper. The main idea behind such a class diagram is to set the main essence of traceability that covers not only the requirement models in terms of basic and refined requirements but also others models as for implementation and design. The other

advantage is to ease the traceability implementation in any tool based on object analysis.

Requirement traceability deals with tracing requirements at two orthogonal aspects.



The first aspect is in the requirement refined/derivation up and down. It means low level requirements (child requirement) can be traced back to at least a high level requirement (ancestor or parent requirement). This traceability is denoted through abstraction. On the contrary, requirements induced through refinement by a high level requirement can be traced from it. Every requirement has an identified origin (source) : it can be another requirement or coming from the external context known as stakeholders, standards, accumulated knowledge, etc.

The other orthogonal aspect concerns links with design and implementation. Two directions are also distinguished. The forward direction concerns traceability from a requirement to design elements and components. This traceability is denoted development traceability for design and respectively for implementation. The backward direction is to trace back from either a designed module or a component to original requirements. Thus it is denoted the reverse traceability from design and respectively from components.

A traditional techniques for surveyed in [13] and more recently in [14].

As discussed earlier, providing traceability of requirements to their sources and the outputs of the system development process can be along several dimensions. Different stakeholders contribute to the capture and use of traceability information, often with different perspectives. A user has a different vision from an audit specialist, a system designer or a validation engineer. Some typical questions are often asked :

What are the systems components that are affected by a specific requirement?

Why are the components affected by such requirements? How are the components affected by such requirement? What are the sources of a low level requirement? Why and how these two requirements are related?

5. A requirements Meta model for an original approach

5.1 Requirement model

We apply the original requirement model developed and presented in previous part. Sources are all available information as documents, phone call, e-mail about the object lifecycle. Traceability concerning specific decision made can be found through the relation documents.

Stakeholder represents all actors involved in producing the source related to an object; Requirements R1 has been captured by user_1 and being document in requirement file Doc_R1. All three meta-classes can be used to create specialized classes in order to adapt the Meta model to any needs for a traceability model for any requirement process as the following basic traceability model which shows the traceability link through refinement/abstraction

5.2 Traceability at low level: An important use of requirements traceability is to ensure that the system meets the current set of user requirements. Representing this information is considered to be critical from a project management perspective. It is usually accomplished by creating a link between the set of requirements and a set of system components that SATISFY them. Such links may also be indirectly derived. An important concern of the study participants was the lack of support in many CASE tools for the automated identification of derived links ("I don't have the time to link every requirement with

everything produced at different stages. These links must be automatically derived whenever possible"). For example, requirements may be linked to designs that are intended to satisfy them. Designs, in turn, may be linked to relevant system components. Then, a derived link is created from requirements to system components.

Such a model is used to identify all traceability links related to requirement-requirement, requirement-implementation (component). A link can be added on system_component to develop decomposition relation at the system, subsystem and component level.

High-level traceability can be modelled by integrating other classes as organization, system mission and standards. Change proposal can be a specialized class.

5.3 Traceability in requirement elicitation process

The model can be deployed with respect all sub-processes mentioned in part 2.

Therefore, what is needed to improve our understanding of requirements elicitation is a more detailed investigating into the common and underlying activities of typical requirements elicitation processes. To this end and to present our own overview of the requirements elicitation process, as once again there is very little uniformity in the research literature and practice concerning the names given to the activities often performed during requirements elicitation. Subsequently, we have divided the various individual requirements elicitation tasks into five fundamental and interrelated activities as listed below and described in the following subsections. The five requirements elicitation activities described are:

1. Understanding the Domains
2. Identifying the Sources
3. Selecting the Methods
4. Eliciting the Requirements
5. Organizing the Information

Traceability customized model

Conclusions and future work

The presented approach takes advantage of the benefits gained from using facilitated collaborative workshops whereby all the relevant stakeholders can cooperatively contribute to the results, and the combination of complementary techniques used to support the main activities, as well as within the actual requirements elicitation workshop environment; traceability remaining as a tool for managing relations between requirements artifacts till possible IEEE SRS model for requirements specification. These strengths are further enhanced by the integration of the entire process into a prescribed set of detailed guidelines based on the underlying knowledge meta-model of information types, thereby ensuring that the process is systematically performed in order to produce a high quality requirements document. We are of the opinion that the resultant approach can produce a requirements elicitation process that is profitable in terms of offering value for effort, therefore encouraging its acceptance and adoption into industry by organizations and analysts. Organizations and analysts are culture dependent.

This paper can be used anywhere in the world especially as a template for getting requirement elicitation and requirement traceability in developing countries. Prospective work is on identifying case studies to enhance and adapt such requirement management processes, elicitation and traceability of requirements with considering keys factors based on society from organization aspects as well institutions culture's key issues

References

[1] Goguen, J. A., Linde, C. (1993): Techniques for Requirements Elicitation, International Symposium on Requirements Engineering, pp. 152-164, January 4-6, San Diego, CA.

[2] Gottesdiener, E. (2002): Requirements by Collaboration: Workshops for Defining Needs, Addison-Wesley: Boston, MA.

[3] Hickey, A. M., Davis, A. M. (2002): The Role of Requirements Elicitation Techniques in Achieving Software Quality, International Workshop of Requirements Engineering: Foundation for Software Quality, September 9-10, Essen, Germany.

[4] IEEE (1998): Std 830 – Recommended Practice for Software Requirements Specifications.

[5] IEEE (1998): Std 1362 – Concept of Operations Document.

[6] Maiden, N. A. M., Rugg, G. (1996): ACRE: selecting methods for requirements acquisition, Software Engineering Journal, 11(3), pp. 183-192.

[7] Sahraoui, AEK, Jones D. A framework for requirements management. Internation conference systems engineering, Las vegas, Oct 1999

[8] Coulin C. , Zowghi,D Sahraoui, AEK: A situational method engineering approach to requirements elicitation workshops in the software development process. Software Process: Improvement and Practice 11(5): 451-464 (2006)

[7] Sahraoui, AEK, Jones D. A framework for requirements management. International conference systems engineering, Las Vegas, Oct 1999

[8] Sahraoui, AEK "Requirements Traceability Issues: Generic Model, Methodology And Formal Basis", ; International Journal of Information Technology and Decision Making, 2005, pp.59-80.

[9] Davis, A. M., Dieste, O., Hickey, A. M., Juristo, N., & Moreno, A. M. (2006). *Effectiveness of requirements*

elicitation techniques: Empirical results derived from a systematic review. 14th IEEE International Requirements Engineering Conference (RE'06).

[10] Lindquist, C. (2005). Required: Fixing the requirements mess; The requirements process, literally, deciding what should be included in software, is destroying projects in ways that aren't evident until its too late. Some CIOs are stepping in to rewrite the rules. *CIO, 19*, 53-60.

[11] Mäder, Patrick, Orlena Gotel, and Ilka Philippow. "Motivation matters in the traceability trenches." *Requirements Engineering Conference, 2009. RE'09. 17th IEEE International*. IEEE, 2009.

[12] Mader, Patrick, and Alexander Egyed. "Assessing the effect of requirements traceability for software maintenance." *Software Maintenance (ICSM), 2012 28th IEEE International Conference on*. IEEE, 2012.

[13] Gotel, Orlena CZ, and Anthony CW Finkelstein. "An analysis of the requirements traceability problem." *Requirements Engineering, 1994., Proceedings of the First International Conference on*. IEEE, 1994.

[14] Rochimah, Siti, Wan MN Wan-Kadir, and Abdul H. Abdullah. "An Evaluation of Traceability Approaches to Support Software Evolution." *ICSEA*. 2007.

[15] Fernandez, Daniel Mendez, et al. "Naming the Pain in Requirements Engineering: Comparing Practices in Brazil and Germany." *IEEE Software* 5 (2015): 16-23.

Towards a Multi-views Approach for Software Requirement Engineering: Requirements Management Tools

Omer Salih Dawood[1], Abd-El-Kader Sahraoui[2]
[1] Department of Computer Science, College of Arts and Science, Wadi Aldawasir
Prince Sattam Bin Abdulaziz University, KSA
Sudan University of Science and Technology, Khartum, Sudan
o.dawood@psau.edu.sa, omercomail@gmail.com
[2] LAAS-CNRS, Université de Toulouse, CNRS,UT2J, Toulouse, France

ABSTRACT
This paper is on Requirements methods and associated tools. It provides general overview of requirement engineering approaches and propose a multi-view approach, and deep comparison on tools and techniques used to manage requirements. The comparison is based on five items, requirement traceability, requirements integration, requirement prioritizing, requirement status, and customer satisfaction. By this comparison it becomes easy to assist the requirement tools and techniques. The paper concentrates on tracing requirement and proposes to develop a model that can be used to handle and manage requirement traceability on large and complex systems.

Keywords
Requirement Engineering; Traceability; requirements tools RTM; DOORS.

1. INTRODUCTION
Requirement engineering is first step in software development. It has many steps elicitation, analysis, and requirement management validation. Its aimed to collect and managed the requirements in a good manner and best way to ensure that all requirements are gathered and analyzed in the way that allow to produce both products and services that satisfying quality attributes [1]. Requirements management (RM) is process of managing changes in the requirements throughout procedure of requirement engineering. Requirement management contains activities related to identification of change, maintenance of change, traceability and changes management of requirements [2]. Requirements management Tools are used to manage the requirements, and shows the relationship between the requirements, and so on. The following section introduces a tools and technologies used to manage the requirements and performs a simple comparative study between three tools DOORS, RTM, and Volere, by this comparative study we want to specify the properties and ability of each, and the problem of each one so that we can enhance the tools and methodology to produce a tool that satisfy the all quality attributes, and we provides multi-views for software requirement engineering, and idea for research in this area. The paper concentrates on requirement engineering traceability.

2. Volere Requirement Specifications:
Volere is used for requirement specifications. It is developed to manage and trace the requirements using volere shell. The shell consist from main attribute that needed when specifying requirement like requirement Id, type, priority, and dependency between requirements.

Volere divides the requirements to the five types each type has the sub components as in the flowing figure (1).

1. Functional requirements are the Basic functions of system that perform core operations of the system. The concrete means are used to measure these functional requirements.
2. Non-functional requirements are the properties that the specify behavior of function such as usability, look and feel, performance, etc.
3. Project constraints describe how the product delivered and fit into the world. The constraint in involves many things like needed interface with existing component for both software and hardware, practice of business, and budget defined at starting project or be ready by a defined date [3].
4. Project drivers are the related forces of business, that drives or control on the process of project development. The purpose of the product is project driver, for all stakeholders in the system in different levels and reasons.
5. Project issues define the conditions under which the project will be done[3]

3. Dynamic Object Oriented Requirements (DOORS)
It is the requirements management tool developed by Telelogic to support the software engineering lifecycle. DOORS is mentioned in several papers and is often referred to as very capable requirements management tool[4].It allows many Users to work together at the same time, and enabling access to the database server that contains information about requirement and links[4].

DOORs database contains many project and each project has its own users and information model. The information model contains a set of module used to keep information about actual requirements and link. DOORs has main three modules:

1. Formal modules Requirements has many artifacts, the artifact contains smaller object. Formal module used to store information about representation of requirement artefact.
2. Link module each formal module has relationship; this relationship is stored in the link module.
3. Descriptive module this module not used .basically to store actual requirement, but now actual requirement stored in formal module [4].object consists from the following [5]

1. General used for describing heading, short text, object text values for the object
2. Access is manage access right to the object
3. History is used as log for changing in object
4. Attributes: is value of object attributes
5. Links: are used to handle the relationships with other objects

4. REQUIREMENT TRACEABILITY MATRIX (RTM)

The requirement engineering has two parts, fist one is requirement development, this part is responsible for requirement elicitation, analysis, specification, and validation. Software Requirement Specification (SRS) is document that produced as output of the requirement development. It contains the requirement specification and it ready to the design phase .The second part of requirement engineering is requirement management, which is responsible for managing requirements and it has two part change management and traceability. Traceability process produces Requirement Tractability Matrix (RTM) as output [6]. The RTM handles requirements and relationships between these requirements in a single document [7].

REQUIREMENT TRACEABILITY

Traceability recognizable association between two or more logical entities like requirements, verifications, elements of system, or tasks. The main two types of traceability are horizontal and vertical traceability but there are other sub types [6].

Vertical Traceability: it shows the source of items and traces these items to Work Breakdown Structure (WBS), to project team and finally to the customers. It insures that the requirement can be traced till satisfied [6].

Horizontal Traceability: It shows the relationship between the related item and work group. It is aimed to avoid the conflicts.

Bidirectional Traceability: It is an association between two or more logical entities that is discernible in either direction. It effectively manages the relationship between requirement sources and requirements of product and its components. In other meaning bidirectional traceability happens between requirement to end product and vise versa [6].

Indirect Traceability: There are two directions for traceability. Firstly is Forward traceability, trace the requirements to its source. Secondly is backward traceability is trace to from product to requirement source [6]. Requirement Traceability Matrix (RTM): Is matrix that used to handle the complete user and system requirements, or a part of the system.

Requirement traceability is helpful in software engineering activities like validation of requirements, and impact analysis, also is useful in tracking the logical sequences and trade-offs for each Requirement.

No.	Comparison Topic	Volere	DOORs	RTM
1	Requirement Traceability	Known as dependability	Support different traceability types	Support different traceability types
2	Requirements Integration	No requirements integration	Support integration, on different levels	Not fully Supporting integration
3	Requirement Prioritizing	No priority	Support priority	Support priority
4	Requirement Status	No requirement status	Has requirement status	Has requirement status
5	Customer Satisfaction	Has customer Satisfaction	___	Depend on RTM Application

Table (1): Comparison of requirements tools and techniques

5. RELATED WORK

Renuka and et al [8] designed a novel methodology to design traceability of onboard software known as Software Requirements to Design Traceability Technique (SoRDeTT). Their methodology is based on two templates, Software Requirements Specification (SRS) and Software Design Document (SDD) as input to the methodology. SoRDeTT represent a common template for both requirement and design which is used to handle the data and information from these documents. There are two trace items; SRS Trace Item (SRSTI) is the template that populates data from SRS. The other one is SDD Trace Item (SDDTI) which if filled with data from SDD. This

methodology applied to satellite system because it's complex and contains many subsystems. onboard software requirements and software design of many subsystems Represented in SRS and SDD. The main purpose of this methodology is to ensure the software design is done according to SRS. The methodology works as follow, data captured from SRS to build SRSTI, SDDTI, comparing SRSTI and SDDTI, if one to one mapping is not found then make tag mismatch, then generate mismatch report, analyze the inconsistency to make a correction, and finally repeat SoRDeTT for all new changes.

Filho and et al[9] propose traceability framework. The model aimed to visualizing the traceability among different tools , and they assume the models are represented in XML to support heterogeneity of tools and models, so that XML is became de facto standard for data interchange, XML supported by many tools, and they use XQuery as standard for traceability rules expression. They assume the model is generated in Native Format Models and converted to (XML_based Models) by using Model Translator.XML model and rules used as input to Traceability_Completeness_Checking Engine.

The traceability relations between the models are generated by the engine, also the sngine identify missing elements based on the rules. Engine uses WordNet component to ensure the identification of synonyms between the of element's names in the models. Traceability_Relations_Missing_Elements document is used to handle the traceability relationship and identified missing elements. This document is important because preserve the original models, to let the use of these models by other different applications and tools. The document used as input to Traceability_Completeness_Checking Engine component that used to support generation of dependent traceability relations. They developed simple prototype tool that shows the traceability relations, and they showed that there are many traceability relations can be generated .

Fab´ıola and Michel [10] extend the SysML requirements diagrams concentrating on traceability of both functional and non-functional requirements. Real-Time Systems can be modelled by this extension of the requirements. The metamodel of SysML is extended with new relationships and stereotypes, and applying the specification of a Road Traffic Control System using proposed metamodel is applied to a set of requirements for the specification of a Road Traffic Control System. SysML is a UML profile and Class diagram stereotype extended new attributes. It decomposes the requirements into smaller related elements in form of hierarchy so that the complexity is managed early. The hierarchy is based on master and slave who allow reusing the requirement. They propose seven new stereotypes to extend the relationships like, copy relationship is represented by master/slave relationship, derive relationship (deriveReqt), satisfy requirement shows how model satisfies requirements, a test case, represented by verify relationship, refine relationship show how a model element used to refine a requirement. The trace relationship act as a general purpose relationship.

6. Ontology for requirements elicitation

Such work based on research roadmap in systems engineering [11] that will be integrated to partially in our work as requirement ontology if well defined and formalised can give rise to better requirements tools . This can make changes to Volere requirements templates.

Problem Definition It is very important to build requirements elicitation on the form used for requirements expression. With the evolution of the Internet and electronic commerce, future business services will often be delivered by autonomous and collaborating parts or software agents inside or across organizational boundaries through negotiation and information exchange over a distributed data network. Efforts are needed to develop collaborative requirement engineering t as an associated need.

The semantics of different information sources are collected by their ontologies, i.e., both terms and the relationships between these sources. In many applications, the intended meaning of a term is often implicit, and understanding this in a collaborative environment necessarily is reliant upon mutual agreements and understandings. In an open environment mutual agreement is hard to achieve. Thus it is very important for the vocabulary, that describes the domain model, to be specified and maintained in such a way that other systems can process them with minimum human intervention. Ontology is used to manage and deal with this task. The ontology research now has more attention from both academia and industry.

It is generally very difficult to build a machine-definable ontology (vocabulary). The semantics of a term varies from one context to another and across different stakeholders. Ideally we need an approach that reduces the problem of knowing the contents and structure of many information resources to the problem of knowing the contents of specific domain of ontologies that user familiar with the domain and easily understands.

Not all requirements are known at starting of the system development. They cannot be specified completely up front in one voluminous document. But rather will evolve during analysis phases of a project and beyond. requirements elicitation involve all stakeholders: users, developers and customers; all see their way matured in the way the requirements are expressed from this step till maintenance; such acquired added value by the elicitation is used to improve the system instead of maintaining the myth that the requirements are to remain static.

Requirement elicitation is one of requirement engineering process. It represents one of the first critical phases. Requirement process is the first phase in systems development. The specific nature of such process is that the

word "elicitation" is new as a technical term; its equivalent does not even exist in some natural languages as French; we use some times the close definition: capture or acquire requirements. Such phase was often neglected as the requirements were only expressed; effectively, we considered were considered to be available and needs only to be expressed. For long period till the 90's the research community was focussing on notation, methods and languages for expressing requirements. The debate was best to use for the sake for genuine expression and also for validation and verification. The debate was transported on the formal versus semi-formal specification of requirements. Most RE researchers have been concerned by such work on taxonomy of methods and adequacy of such methods and notations for expression requirements for various types of applications: in formation, systems,

The requirements elicitation is one of the most important steps in requirements engineering project. Experience over the last decennia has shown that incorrect, incomplete or misunderstood requirements are the most common causes of poor quality, cost overruns and late deliveries. The ability to use an adequate approach thought a method or systematic process is therefore one of the core skills in systems development. The GAO survey is a demonstration through figures on nine projects totaling about $7 millions.

A terminology (CMU): The procedure of understanding systems requirements can be defined and described by many terms. Requirements engineering can be used as a general terms including all activities related to requirements. In fact, requirements engineering consists from four specific processes

Requirement elicitation: Is first process allowed to understand, discover, reveal, and articulate the requirements to customers, buyers, users of a system.

Requirements analysis: This process is based on the requirement elicitation. It is reasoning the elicited requirements; it involves some activities such as checking requirements to ensure from both conflicts and inconsistencies, combining requirements that related to each other's, and specify missing requirements.

Requirements specification: In This process the requirements are recoding in the forms, this including the may be done in natural language, symbolic, formal, and diagrammatically representing the requirements, also the product that is the document produced by that process.

Research approach. The suggested research approach involves development of a shared ontology: A shared ontology can be in the form of a document or a set of machine interpretable specifications. Among possible contemporary research projects that deal with ontology-based approaches to resolving the semantic issues, the following seem especially appealing.

Requirements validation: In this process the requirements are confirmed with the users of systems, and customers to ensure that the specified requirements are valid, complete and correct.

In an actual situation, these four processes cannot be strictly separated and performed sequentially; they are interleaved and performed iteratively.

The term elicitation is not universally accepted for the process described; there are no similar term in other language, in example French language; the term acquisition, capture is often used; some companies use gathering, expressing, formulating. Each term has a different connotation. Acquisition supposes the requirements are already there like sensor value acquisition by I/O system of a computer system. Apart from the term used, all of these terms address implicitly the elicitation term.

i. Common domain model: although participating agents share a common domain that is the basis of their cooperation, they often have different views of the domain. In order for them to collaborate, a common domain model is required to facilitate their communications.
ii. Different levels of abstraction: different levels of information abstraction are required by a flexible enterprise. At the agent level, only high-level business process and service concepts are needed to form service level agreements, i.e., contracts. At the task scheduling level, processes and services must be viewed in term of individual tasks and task interfaces (methods and conditions). At the execution level, data representation must be explicit so that data can be transformed and fused correctly.
iii. Dynamic information integration: the underlying information systems are potentially large. New services may require only parts of the information systems to be integrated. Dynamic information integration is required as which parts to be integrated for what purposes cannot be determined beforehand.
iv. Service and contents description: agent services and information system contents must be formally described. The descriptions must be accessible and meaningful to all participating agents.
v. Information heterogeneity reconciliation: as flexible enterprises operate in an open environment, participating agents often use conflicting terms. In order for them to collaborate, the heterogeneity must be reconciled.

Expected results. The suggested research should result in several needed and useful outcomes.

i. Developing a requirements domain ontology environment for effective and efficient requirements elicitation will represent a considerable advance in requirements engineering. This will necessarily involve identification of appropriate support environments needed to assist ontology designers with the tasks involved in ontology management. It is

envisaged that such an environment would maintain an ontology repository that can be accessed. During the design phase to enable this, tools will be available to browse and reuse the terms from the repository. When new terms need to be added, checks should be performed to see that they do not cause inconsistency in the repository. This environment should also have a set of tools that help extract ontological information that is embedded in existing systems.

ii. Develop appropriate methods and tools to support the integration of process models and information systems from multiple organisations during requirements change.

iii. Extending XML in requirements for data sources and ontology extraction and retrieval. Integrate the ontology for requirements elicitation into a general framework and context to support systems engineering in a computer supported cooperative work environment.

7. RESEARCH OBJECTIVE

1. Better understanding of requirement management tools and techniques.
2. Evaluate mentioned tools to become easily when requirements management tool is needed.
3. Improving software quality by determine each tool capabilities to minimize the problems risk of requirement management.
4. Evaluate requirement traceability in each tool and techniques.

8. RESEARCH PROBLEM

From literature review there are many researches in requirement traceability but not detailed covered the one of the important topic in requirements validation and verification through traceability. It is expected to develop a requirement traceability model or tool that allows enhancing and improving software quality through tracing requirement in a good and best manner.

9. DISCUSSION

Requirement engineering is first step of software development its aim to collect and document requirements. The cost of detecting and managing errors in earlier stages is less than detecting errors in later stages. Traceability is very important to handle and managing requirements, because its allows easily tracking requirements. There are many research covered this area but still some gaps and missing are found. The previous research concisely covered the traceability issues and concentrate on small to medium systems. This research aimed to full the missing points on the previous studies and develop a model for requirement traceability that allow handle the traceability in big and complex system.

By developing the new model its expected to produce highly and well requirement modeling and techniques that produce software with high level of quality, and requirements of the complex system can be managed in easily way.

10. CONCULISION

This paper covered the concept of requirement engineering and comparing between some of the current tools and techniques that used to manage requirements. In the area of requirements engineering we concentrate to requirement traceability because it very important to manage and handle the requirement. Many previous researches are reviewed and concluded in this paper, and some interested areas are shown and discussed .this paper introduces some research problems like requirement traceability and requirement ontology. Its expected to solve the problem of requirements elicitations verification and validation through requirements traceability.

11. ACKNOWLEDGMENTS

The authors and mainly second author are indebted to many colleagues who contributed directly or indirectly to this work and mainly Late Professor Andy Sage from Georges Mason University and Professor Dennis Buede from New Jersey University.

12. REFERENCES

[1] Hummera, Yasir, Sohail, Muhammad, Asma, 2013.Effective Usage of AI Technique for Requirement Change Management Practices. 5th International Conference on Computer Science and Information Technology CSIT (Aman, 27-28 March 2013).978-1-4673-5825-5, 2013. DOI: 10.1109/CSIT.2013.6588769

[2] Dhirendra, U. Dhirendra, A. K. Ramani. 2010. An Effective Requirement Engineering ProcessModel for Software Development and Requirements Management. 2010 International Conference on Advances in Recent Technologies in Communication and Computing(Kottayam. 16-17 Oct. 2010) DOI: 10.1109/ARTCom.2010.24

[3] James and Suzanne Robertson. Volere Requirements Specification Template. (London, Aachen & New York).DOI: http://homepages.laas.fr/kader/Robertson.pdf

[4] B.J.M. Abma. 2009. Evaluation of requirements management tools with support for traceability-based change impact analysis.Master Thesis, University of Twente.

[5] Vassilka Kirova and Darshak Kothari. 2004. Getting the most out of DOORS for requirements management. Technical Report. Software Technology Center, BLAT.

[6] Surachet and Yachai.2010. Enhancement of Requirements Traceability with State Diagrams. 2010 2nd International Conference on Computer Engineering and Technology (ICCET) (Chengdu. 16-18 April 2010) DOI: 10.1109/ICCET.2010.5485417

[7] Software Testing-Requirements Traceability Matrix.DOI: http://www.etestinghub.com/requirements_traceability_matrix.php

[8] Renuka and et al.2014. NOVEL METHODOLOGY FOR REQUIREMENTS TO DESIGN TRACEABILITY OF ONBOARD SOFTWARE. 2014 International Conference on Advances in Electronics, Computers and Communications (ICAECC), (Bangalore, 10-11 OCT 2014)DOI: 10.1109/ICAECC.2014.7002386

[9] Fab´ıola and Michel. 2013. A Metamodel for Tracing Requirements of Real-Time Systems. 16th IEEE International Symposium on Object/component/service-oriented Real-time distributed Computing (ISORC 2013)(Paderborn, 19-21 June 2013).DOI: 10.1109/ISORC.2013.6913189

[10] Gilberto and Maria. 2012. Towards a Traceability Visualization Tool. 2012 Eighth International Conference on the Quality of Information and Communications Technology. (Lisbon, 3-6 Sept. 2012).DOI: 10.1109/QUATIC.2012.60

[11] SAHRAOUI and et al. 2008. SYSTEMS ENGINEERING RESEARCH. Journal of Systems Science and Systems Engineering.(16 July 2008). DOI: 10.1007/s11518-008-5083-9

SysML State Machine Diagram to Simple Promela Verification Model Translation Method

Takahiro Ando[1], Yuya Miyamoto[2], Hirokazu Yatsu[1], Kenji Hisazumi[3], Weiqiang Kong[4], Akira Fukuda[1], Yasutaka Michiura[5], Keita Sakemi[5], and Michihiro Matsumoto[5]
[1]Graduate School of Information Science and Electrical Engineering, Kyushu University, Fukuoka, Japan
[2]DOCOMO Systems Inc., Tokyo, Japan
[3]System LSI Research Center, Kyushu University, Fukuoka, Japan
[4]School of Software, Dalian University of Technology, Dalian, China
[5]Japan Manned Space Systems Corporation, Tsukuba, Ibaraki, Japan

Abstract— *In this study, we developed a method for converting SysML state machine diagrams into Promela models that can be verified using the SPIN model checking tool. The Promela code generated in our approach is a sequential verification model that simplifies the verification process when used in the early stages, and also prevents state explosion in the verification process. Thus, using the sequential verification model reduces the cost of the overall verification process. In this paper, we describe the rules used to convert the SysML state machine diagrams with parallel processes to a single sequential process in Promela.*

Keywords: State Machine Diagram, SPIN, SysML, Model Checking, Formal Method

1. Introduction

Software is embedded in various devices, machines, and equipment, including smartphones, automobiles, space equipment. Because this embedded software needs to be very reliable, at each stage of development, rigorous verifications are required.

Model checking [1] is a well-known verification technique for formally analyzing state transition systems. In model checking, a target system is modeled in a formal description language and the model is exhaustively explored to check whether desired properties of the system are satisfied. SPIN [2], NuSMV [3], and UPPAAL [4] are state-of-the-art model checkers. To use any of these tools, a state transition diagram of the target system is first modeled in the formal description languages corresponding to the desired model checker. Further, the properties to be checked are written in a formal specification language such as Linear Temporal Logic (LTL) or Computation Tree Logic (CTL).

In formal verification such as model checking, factors such as verification costs, time, memory size, human resource, are major issues. Therefore, during the early stage of verification processes, verification with a simple model to reduce the cost is desired. The quality of the verification model is dependent on the skill of the verification engineer, which is often a problem. Therefore, automatic generation of the verification model, i.e., making it independent of the engineer's skill, is desired. In addition, because analysis of the verification results often tends to be complex, a technique that simplifies this process is also necessary.

In this paper, we discuss the automatic generation of simple verification models from the state machine diagrams in SysML [5]. In particular, we focus on the verification model used in the SPIN model checker, and propose a translation method that converts SysML state machine diagrams into simple verification models for SPIN. Using our translation method, the behavior of a state machine diagram with parallel processes is translated into a simple verification model with a single process in Promela, the input language used by SPIN. We demonstrate the efficacy of our proposed method by applying it to verification of a simple system.

2. Related Work

Much research has been conducted on the application of formal verification techniques to formally analyze state machine diagrams. Bhaduri and Ramesh [6] carried out a comprehensive survey of studies that applied model checking to state machines, in which various model checkers including SPIN [2], SMV [7], and FDR [8] were used.

Latella et al. [9] translated state machines into models written in Promela and then verified them using SPIN. However, they only dealt with the basic components of state machine diagrams. Lilius and Paltor [10] proposed a tool called vUML for verification of UML models using SPIN, but presented no details of the rules used to translate the models into Promela.

3. Proposed Translation Method

In this section, we describe our proposed state machine diagram to verification model translation method. The verification model is written as a simple process in Promela. Each element of the model is lumped using a macro description, in order to easily recognize the correspondence between the elements of the original diagram.

In our verification model, the parallel processes in a state machine are combined into one sequential process in Promela. Although, our verification model cannot represent all the behaviors of the diagram with parallel processes, it is useful in the early verification stage because a simple model prevents state explosion and reduces the cost of verification.

3.1 Overview

The inputs to our translation method are the state machine diagram and information about its variables. Information about the serialized state machine is stored in an XMI [5] file, and information about its variables is stored in a CSV file. The translation rules shown below translate the components of the diagram into their corresponding description in Promela.

1) The state names in the diagram are translated into mtype values in Promela.
2) An mtype variable that represents the current state is provided for each region in the diagram.
3) The event names in the diagram are translated into mtype values.
4) An mtype variable that represents the current event occurrence is provided.
5) On the basis of the input variable information, the corresponding variables in Promela are provided for the variables in the statement or guard condition in the diagram.
6) The initial pseudo state is translated into an inline macro description.
7) The states, including composite states, are translated into three types of inline macros.
8) The outgoing transitions of a state are translated into one inline macro.
9) The regions are translated into two types of inline macros.
10) The final state in each region is translated into an inline macro.
11) The event occurrence model is described as an inline macro.
12) The behavior of the original state machine is described by one process in Promela.

In the following subsections, we explain our translation rules in detail along with translation examples. Each of the following examples are obtained by translating the corresponding element of the state machine diagram shown in Fig. 1.

3.2 Declaration of Values and Variables

Each state name and event name is translated into an mtype value in Promela (lines 1–4, 10). Each variable representing the current state in a region is declared an mtype variable (lines 6–9), and each variable representing current event occurrence is declared an mtype variable (line 11). Each variable in guard conditions and actions in the state machine diagram is translated into a variable in Promela using the variables information in the CSV file(lines 13–17).

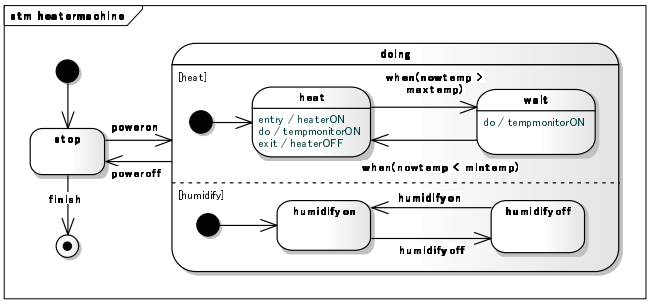


Fig. 1: State Machine Diagram for a Simple Air Conditioner

```
1  mtype = {top_init, top_stop, top_doing,
       top_final};
2
3  mtype = {doing_heat_init, doing_heat,
       doing_wait};
4  mtype = {doing_humidification_init,
       doing_humidifyon, doing_humidifyoff};
5
6  mtype topState = top_init;
7  mtype doing_heatState = doing_heat_init;
8  mtype doing_humidificationState =
       doing_humidification_init;
9
10 mtype = {NULL, finish, poweron, poweroff,
       when01, when02, humidifyon,
       humidifyoff};
11 mtype event = NULL;
12
13 int nowtemp = 25;
14 int maxtemp = 30;
15 int mintemp = 20;
16 bool heater = false;
17 bool tempmonitor = false;
```

3.3 Translation of Initial Pseudo States

Initial pseudo states are translated into inline macros that each has an inline macro call for the state entry behavior. The state is pointed to by the initial pseudo state.

```
1  inline T_init() {
2    S_stop_entry()
3  }
```

3.4 Translation of States

Each state is divided and translated into three inline macros: entry part (lines 1–6), do part (lines 8–11), exit part (lines 13–19). The entry part represents the entry behavior of the state, the do part represents the do behavior, and the exit part represents the exit behavior. In addition, when the state has regions, the entry part has inline macro calls for the entry behaviors into the regions (lines 4–5), the do part

has inline macro calls for the behavior in the regions (lines 9–10), and the exit part has inline macro calls for the exit behavior from the regions (lines 14–15).

These inline macro calls in each part are not parallel but sequential. Consequently, an execution sequence in our verification model is also sequential, and the correspondence between the execution sequence and the lines of the verification model code can be easily observed.

```
1  inline S_doing_entry() {
2    topState = top_doing;
3
4    T_doing_heat_init();
5    T_doing_humidification_init()
6  }
7
8  inline S_doing() {
9    R_doing_heat();
10   R_doing_humidification()
11 }
12
13 inline S_doing_exit() {
14   R_doing_heat_exit();
15   R_doing_humidification_exit();
16
17   doing_heatState = doing_heat_init;
18   doing_humidificationState =
         doing_humidification_init
19 }
```

3.5 Translation of Outgoing Transitions

Our transition rules translate all outgoing transitions of a state into an inline macro in a group. The internal transitions are dealt with in a manner similar to the outgoing transitions and they are translated into the same macro as outgoing transitions. In this macro, the transitions are described as conditional branches by the trigger event occurrences. The inline macro call of the exit behavior of the source state is placed before the actions of each transition. The inline macro call of the entry behavior of the target state is placed after the actions. These behaviors are obtained from the semantics of the state machine diagrams.

```
1  inline T_stop() {
2    if
3    :: (event == poweron) -> event = NULL;
4       S_stop_exit(); S_doing_entry()
5
6    :: (event == finish) -> event = NULL;
7       S_stop_exit(); S_final()
8
9    :: else -> skip
10   fi
11 }
```

3.6 Translation of Regions

Each region of the state machine diagram is translated into an inline macro that combines the inline macro calls of the states and transitions in the region (lines 1–9). The inline macro represents the behavior in the region. The exit behavior from the region is translated into a different inline macro (lines 11–19).

```
1  inline R_doing_heat() {
2    if
3    :: (doing_heatState == doing_heat) ->
4       S_doing_heat(); T_doing_heat()
5
6    :: (doing_heatState == doing_wait) ->
7       S_doing_wait(); T_doing_wait()
8    fi
9  }
10
11 inline R_doing_heat_exit() {
12   if
13   :: (doing_heatState == doing_heat) ->
14      S_doing_heat_exit()
15
16   :: (doing_heatState == doing_wait) ->
17      S_doing_wait_exit()
18   fi
19 }
```

3.7 Translation of Final State

The final state is translated into an inline macro that terminates the operations of the behavior in the region.

```
1  inline S_final() {
2    topState = top_final;
3    break
4  }
```

3.8 The Event Occurrence Model

In our verification model, we adopted the following event occurrence model. In the model, the events that can occur are the events that are triggers of the current states, including inner states.

```
1  inline eventOccur() {
2    event == NULL;
3
4    if
5    :: (topState == top_stop) ->
6       event = poweron
7    :: (topState == top_stop) ->
8       event = finish
9
10   :: (topState == top_doing) ->
11      event = poweroff
12
13   :: (topState == top_doing &&
        doing_heatState == doing_heat &&
        nowtemp > maxtemp) ->
14      event = when01
15
```

```
16      :: (topState == top_doing &&
            doing_heatState == doing_wait &&
            nowtemp < mintemp) ->
17         event = when02
18
19      :: (topState == top_doing &&
            doing_humidificationState ==
            doing_humidifyon) ->
20         event = humidifyon
21
22      :: (topState == top_doing &&
            doing_humidificationState ==
            doing_humidifyoff) ->
23         event = humidifyoff
24      fi
25  }
```

3.9 Process for State Machine Behavior

Only one process is present in our Promela verification model. This process represents the overall operation of the original state machine diagram. It starts from the inline macro call of the initial pseudo state and alternates the event occurrences and the state transitions for the events.

```
1  active proctype stm() {
2    T_init();
3    do
4    :: eventOccur();
5       R_top()
6    od
7  }
```

4. Case Study

In this section, we confirm the efficacy of our translation method for SPIN model checking.

We applied our translation method to the state machine diagram shown in Fig. 1, and generated the corresponding Promela code. Using the code as the input to the SPIN model checker, we conducted LTL verification. We uses the following LTL formula, which states that when the current state is the doing state, the stop state cannot be reached forever, as a verification property.

```
1    [](  (topState == top_doing) ->
2         [](topState != top_stop) )
```

The result of this verification case study is shown in Fig. 2. The figure shows that a transition sequence from the doing state to the stop state has been found. This is appropriate as a result for this verification case study, and shows that the generated model can be used for SPIN model checking.

The counterexample sequence for this case study is shown in Fig. 3. The figure shows that the complexity due to the execution order of the parallel behavior has been rectified—making it easy to observe the correspondence between the steps in counterexample and the code corresponding the verification model.

These results show that our proposed method is useful in the model checking process.

5. Conclusions

In this paper, we described our proposed translation method that converts state machine diagrams with parallel processes into simple SPIN verification models with a single Promela process. In our simple verification model, using inline Promela macros, we can easily recognize the correspondence between Promela codes and each component of the original diagram. Our translation rules convert states that have some regions into inline macros with sequential inline macro calls for the regions. The verification model alternates the event occurrences and state transitions for the event. Thus, a state transition sequence in our model is simple, and we can analyze the verification results more easily.

In future work, we plan to develop a translation method similar to the one proposed here, but which translates state machine diagrams into simple parallel verification models. We believe that using a one-step complex model, we can implement stepwise from simple verification to full verification, and then we can further reduce the verification cost. In addition, we plan to develop an automatic translation system based on our translation method. We are also planning to apply our method to various examples and refine the translation rules using feedback.

References

[1] E. M. Clarke, O. Grumberg, and D. Peled, *Model Checking*. The MIT Press, 1999.

[2] G. J. Holzmann, "The Model Checker SPIN," *IEEE Transactions on Software Engineering — Special issue on formal methods in software practice*, vol. 23, no. 5, pp. 279–295, May 1997.

[3] A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri, "NUSMV: A new Symbolic Model Verifier," in *Proc. of the Eleventh Conference on Computer-Aided Verification (CAV'99)*, ser. Lecture Notes in Computer Science, N. Halbwachs and D. Peled, Eds., no. 1633. Trento, Italy: Springer, July 1999, pp. 495–499.

[4] K. G. Larsen, P. Pettersson, and W. Yi, "Model-Checking for Real-Time Systems," in *Proc. of Fundamentals of Computation Theory*, ser. Lecture Notes in Computer Science, no. 965, Aug. 1995, pp. 62–88.

[5] OMG, "OMG Systems Modeling Language Version 1.3," June 2012. [Online]. Available: http://www.omg.org/spec/SysML/1.3/PDF

[6] P. Bhaduri and S. Ramesh, "Model Checking of Statechart Models: Survey and Research Directions," *CoRR*, vol. cs.SE/0407038, 2004.

[7] K. L. McMillan, "Symbolic Model Checking: An approach to the state explosion problem," Ph.D. dissertation, Pittsburgh, PA, USA, 1992.

[8] The Formal Systems website, "FDR2.91," http://www.fsel.com/, November 2010. [Online]. Available: http://www.fsel.com/

[9] D. Latella, I. Majzik, and M. Massink, "Automatic Verification of a Behavioural Subset of UML Statechart Diagrams Using the SPIN Model-checker," *Formal Asp. Comput.*, vol. 11, no. 6, pp. 637–664, 1999.

[10] J. Lilius and I. P. Paltor, "vUML: A tool for verifying UML models," in *Proc. of the 14th IEEE International Conference on Automated Software Engineering, ASE'99*. IEEE, 1999, pp. 255–258.

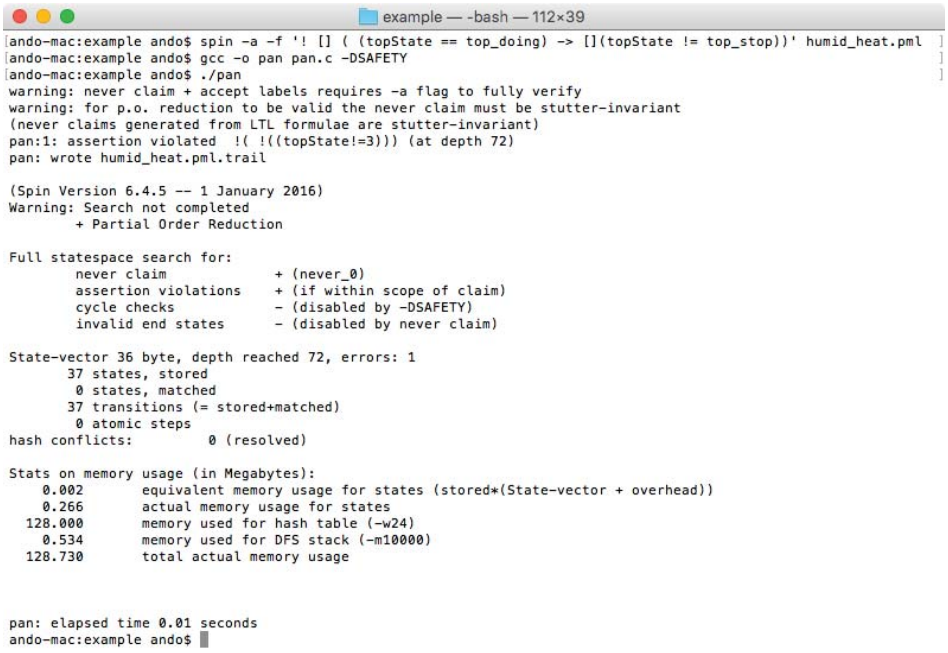


Fig. 2: Result of verification case study using the SPIN model checker

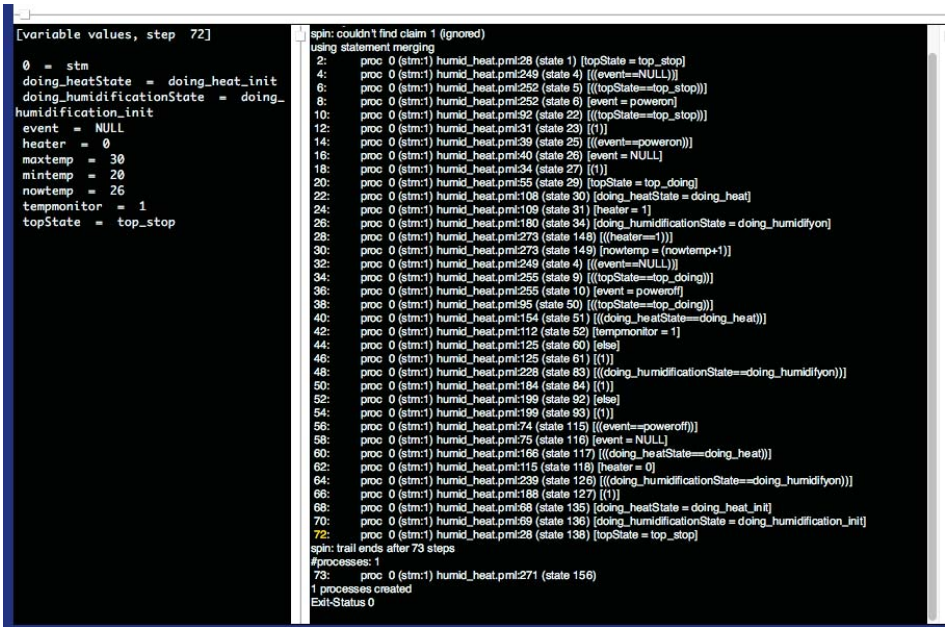


Fig. 3: Counterexample of a verification for the system using the SPIN model checker

SESSION

PROCESS AND PRODUCT LINE IMPROVEMENT + AGILE DEVELOPMENT AND MANAGEMENT

Chair(s)

TBA

ISBN: 1-60132-446-4, CSREA Press ©

ISBN: 1-60132-446-4, CSREA Press ©

Development of a System for Monitoring and Controlling Research Projects based on a Framework Integrating Traditional and Agile Methodologies

C. Gutiérrez[1], S. Díaz[1], J. De La Rosa[1], K. Gómez[1], C. Baron[2], I. Reyes[1], and M. Villanueva[1]
[1]Departamento de Ingenierías, Instituto Tecnológico de Toluca, Metepec, Edo. de México, México
[2]LAAS-CNRS, Université de Toulouse, CNRS, INSA, UPS, Toulouse, France.

Abstract – *The information systems used in the educational sector allow the automatization of tasks and processes, the reduction of time and effort, as well as the coordination and management of activities in an efficient manner. This paper presents the analysis, design and development of an information system for the monitoring and control of research projects in the educational sector, and which is based on a framework led by software engineering. An innovative aspect of this research consists of the integration of characteristics to the information system that contribute to the improvement of the education quality since these permit the supervision and control of several aspects. Its impact is reflected in the generation of evidence of the followed process as it complies with some indicators of the Quality Management System.*

Keywords: Information System, Software Engineering, Framework, Traditional and Agile Methodologies.

1 Introduction

Nowadays, software development has become more and more important in regards to quality, for this reason, software engineering seeks the integration of techniques, methodologies and models in its analysis, design and development processes to generate highly technological products [1], [2].

Since 1968, and as a consequence of the NATO conference where the concept of "software engineering" saw its first light, [3], the development of programs, started to be taken more seriously. In this sense, the definition of requisites, the organization of development and the generation of documentation of a product are but a few of the aspects that demand more interest by software engineering, thus, an application of a series of regulations, common language and a working discipline is necessary, which allow knowing the real state of the conception process in each stage of the process.

Currently, educational systems seek the reduction of the work load, as well as of the overload of information, as computers carry out more and more daily activities such as the management of email, social interactions and entertainment. However, technological advancements do not go at the same pace as the way people interact with machines as almost every system requires an explicit initialization as well as a monitoring of every event, for this reason, intelligent systems can make these changes. Such systems seek not only the improvement of the educational service, but also the automatization of administrative tasks that allow using the information easily.

Information Systems require certain techniques and characteristics to be successfully developed and implemented. Software Engineering allows identifying the characteristics certain information system must have, so it offers methods and techniques to maintain, develop, produce and ensure quality software [4], [5] and [6].

Mexican education systems set as a regulation that the culmination of higher studies is reached when the requirements of the institution are met, and with the conclusion of a research project or dissertation.

The number of students enrolling in a higher studies institution grows every year; it is frequent to notice a correlation between the level of studies and the salary offered by employers, this relation can be observed in the statistics obtained by the education institutions that have a follow-up process with their graduates, and they relate it to their salaries.

When considering that a research project or dissertation is a requirement to obtain a professional degree in higher studies, this research project began with the analysis of the number of students who do not conclude with this graduation process. It was observed that an important aspect lies in the lack of monitoring, orientation and continuity by the professors in the projects developed by students. Under this, a group of researchers [7] and [8], developed an Innovative Model, which under a multidisciplinary and collaborative platform, provides a methodological and procedural guide that facilitates the tasks of the teachers and provides students with tools, techniques and methodologies to generate products and their final documents.

In this way, with the model, every semester between 120 and 240 students are attended to, and these students generate between 40 and 70 projects. Some of them opt for obtaining their degree by means of these projects since at the end of their studies they generate a product or a system, as well as their final document, which is provided to their administrative department to continue with their graduation process.

However, despite the efficacy of the model, it is complex to oversee the large number of projects presented and therefore, a centralization and automatization of the

model for the monitoring of the projects, allows a greater number of graduated students.

In order to solve the mentioned problem, the main objective of this research project was centered in the creation of an information system, in the educational sector, for the monitoring and control of research projects using WEB technologies.

In this paper it is described how the analysis, design and development of the system was carried out, following a Framework that integrates Agile and Traditional Methodologies defined by the software engineering where aspects for risk management, requirements compliance and versions control are considered.

One of the main contributions of this research project is the fact that the system integrates characteristics that contribute to the improvement of the education quality by allowing the monitoring and control of the necessary elements for the development of a product, and up to the integration of teams to provide the project a follow-up. Its impact is reflected in the generation of evidences, in the followed process to comply with quality standards as established by the international norm ISO 9001:2015 [9] and certification organisms such as the CACEI [10] and CONAIC [11].

Another contribution of this project, on a social scale, was focused to create the interest of the student and academic bodies for adhering themselves to the processes, methods and techniques established for the monitoring and control of research projects and to collaborate in the validation of a process that can develop and strengthen the skills, attitudes and values of both faculty and student bodies.

2 Related software programs and projects

There are different projects related to the proposal described in this paper, these are divided into three topics; 1) Obtention of a degree through projects; 2) WEB systems for the obtention of a degree and 3) Software for the obtention of a degree.

2.1 Obtention of a degree through projects

Different Universities in the country offer the option of obtaining a degree through the presentation of research projects, this modality is convenient for the students to obtain their degree and to perform better in the academic area. Broadly speaking, the process is as follows:

1. A research project where at the end a product, system or software is generated, along with its final document.
2. The aforementioned research report is given to the Institution (from where the student graduated from).
3. The document or research report is revised by an assessment committee which examines in detail both the research and the product.
4. The committee provides feedback.
5. The student makes the necessary changes.
6. The processes is repeated from 3-5 until the committee issues its approval.

In the paragraphs below are listed a series of universities where the above mentioned modality is offered.

Universidad Autónoma de Guadalajara (Autonomous University of Guadalajara) [12]. This modality is offered as "Prototypes Project", where a project is created which will be later used for the obtention of a degree. In the UAG's main webpage the general characteristics the document must comply with for it to be approved are shown. However, it does not allow monitoring or controlling the projects.

Universidad de Sonora [13]. (Sonora University). In this institution, the Division of Economic and Administrative Sciences offers the option of presenting a research project to graduate. In its main web page, the members of the academia register their research project, initializing with this step the procedure. The page shows the registration process, however there is lacking a module where one can upload the documentation and track the complete process.

2.2 WEB Systems for the obtention of a degree

There is also software that provides support in the graduation process, and these were developed in WEB environments.

Universidad Autónoma de México [14]. (Autonomous University of the State of Mexico). The system developed at this University, specifically in the Faculty of Higher Studies Acatlán, shows a section identified as *Carrera* (*Career*), which integrates the different degree courses offered by this Faculty. Syllabii from 1978 to 2015 can be found there, and also an identification number, which corresponds to the student's identification number. It is a complex system which has been in operation since 2012, according to the information found in the page. However, this system does not allow the monitoring and control of the projects developed by students and assessors.

Universidad Autónoma de México - Facultad de Ingeniería [15]. (Autonomous University of the State of Mexico - Faculty of Engineering). The Faculty has a WEB system supported by the institutional service of the Faculty of Engineering, this system integrates an interface with a menu which provides important information about the process to obtain a degree such as the guide for graduation, academic orientation on graduation, amongst others. The system is one of the most complete systems in the WEB, however, it does not include aspects for monitoring and controlling the projects.

Instituto Tecnológico de Villahermosa [16]. (Villahermosa Technological Institute). This institution has a WEB system under development, it offers an option for consultation of the process by providing a control number; a disadvantage of this product is that it does not provide a level of security. In its main page are shown courses, formats, exam modules, graduation options and pre-registration, an appointment system with user and password.

2.3 Software for the obtention of a degree

There are also information systems, of a commercial nature, that provide support, from the enrollment up to the obtention of the students' degree. Among these there is GES (Gestión Educativa Software - Software Education Management) [17], developed by HT Mexico, Services and Consultancy. It provides support for the generation of class schedules, groups by time, schedule by classroom; it controls the enrollment and graduation process, social service or community service, professional practicum and residency; it also helps with the configuration of the different graduation options, the administration of the graduation of work for the graduation, the allocation of assessor and of the board of examiners, the organization for the professional exam, the designation of the result, the printing of the professional title and of the different certificates and minutes necessary.

Another commercial software analyzed was the Bit Academic Manager [18]; it consists of a system of academic and administrative control for the academic world. It is capable of carrying out the academic control for the enrollment, evaluation and the issuance of different documents for the students. Also, it allows professors to assess and grade online on an everyday basis for a better control.

Lastly, software Click [19], is a tool created by Grupo Index, and it consists of an academic control system. It has a wide range of functions for the administration of the academic world and a social service and professional practicum module. It even has functions for the control for graduated students. A disadvantage, however, is that it lacks tools to support the graduation process.

After analyzing and studying different information systems, it was determined that none of these products covers the need of being a support tool for graduation, a tool that provides monitoring and control to the students' research advancement.

2.4 Proposal

We decided to create a program that is capable of the automatization of the products development process and administration of the research projects. With this objective in mind, in consensus with the team, some characteristics of the previously described systems were considered; among these are the generation of schedules, users' access, generation of performance graphs, administration of content by the final user and that it would need to have the function of a multiplatform system.

Apart from considering the previous characteristics, the team decided to add a plus to this information system by integrating new functionalities and capabilities for the correct supervision of the procedures. In the Frame Work in section Three this process is detailed.

3 A Framework as a guide for the development of the system

When the activity of the software engineering is expressed as a multidisciplinary study and allows the interaction of different areas, it is possible to generate the development of complete and functional systems following a model that describes their operation and behavior. The development of the Sistema de Registro de Proyectos (Projects Registry System) was based on the Framework [7], [8]; which follows the recommendations of the software engineering, using agile and traditional methodologies (See Figure 1).

The framework integrates different tools that allow the coupling of two processes: the development and evolution of the software products and the management of projects, under a context that allows collaborative and organized work and, in a short time frame, results that are concrete and approved by the client are generated and moreover, these provide an added value.

In the first three stages of the Framework a waterfall model was used [5], [6], useful for the definition of systems requirements, the analysis and part of the design. After an incremental-evolutionary model was used for the development by system increments. Both models belong to traditional methodologies. Also, some recommendations defined by the agile methodologies were followed for the organization of the personnel involved in the project, the allotment of resources and the management of activities. In this work SCRUM and Kanban [20] were used.

3.1 Analysis and Design

3.1.1 Information Search

The first activity was the creation of a multidisciplinary work team of six people who were asked to deliver periodically small parts of the system, as well as the search of current technologies for WEB development, databases servers, programming languages, and software that would allow the use of a server. The group carried out the research of Education Institutions in Mexico in order to study other systems similar to the one developed in this project; as well as the research projects registry, control and monitoring processes followed by other institutions. The collaborators were assigned different roles.

The Waterfall Model was of use in the analysis, selection and optimization of the information, SCRUM in the allotment of activities and organization of the team.

3.1.2 Analysis, Selection and Optimization of the Information

In this stage, the classification of the documents was carried out in order to determine which information was to be of use in the development of the product. Here, a feedback between the present stage and the previous one to determine whether there was hidden or unknown information at the

moment of the selection, and to become familiar with the environment.

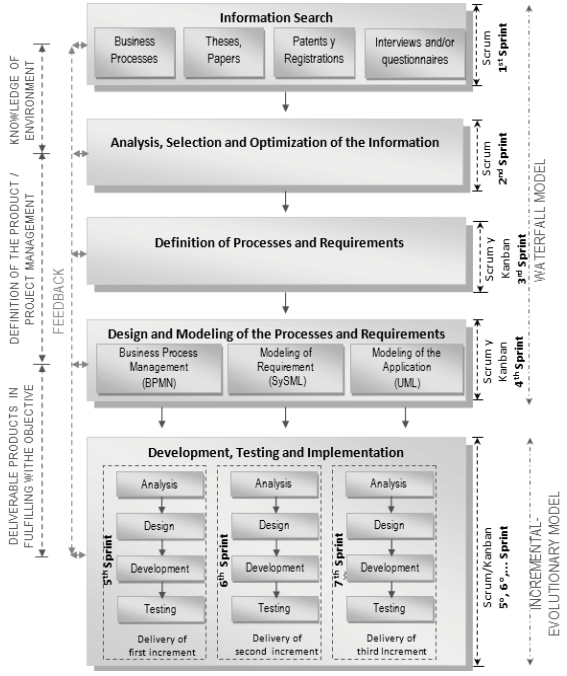


Figure 1. Framework.

The Waterfall Model continued to be used for the analysis, selection and optimization of the information and SCRUM for the allotment of activities and the organization of the team.

3.1.3 Definition of Processes and Requirements

This stage consisted on the definition of requirements in natural language in collaboration with the client. The requirements were analyzed and classified as functional and non-functional. Later, a document with all the identified requirements was drafted.

With the information obtained at this stage, a risk analysis was carried out in order to reduce the impact in the development of the system and achieve a more fluent and real development. Among the risks we can find:

Technology Risks. The computers which were originally destined to be the system server did not fulfill the necessary requirements to provide an adequate performance; the use of different operational systems for the development and execution of the program could create lags in the compatibility and partial execution of the program; the software used as a server and the database software, in its open code, were probably lacking all the necessary resources for housing the system.

Requirement Risks. The constant increment of requirements in each module or increment in the system, would create considerable modifications in the system.

Personnel Risks. The meetings to test the system proved to be difficult to schedule due the different activities of each member.

Risk Planning. Risk planning needed an analysis of each one of the risks defined and the proposal of a strategy that allow their management. In the monitoring and control system of the research project several strategies were identified, these can be seen in Table 1.

Table 1. Strategies for Risk Management.

Risk	Strategy
The computer equipment originally destined to be used as the server for the system did not comply with the necessary requirements to provide a good performance.	New equipment was considered, clearly one that had superior characteristics compared to the original one.
The use of the different operation systems for the development and the execution of the system included some drawbacks in terms of compatibility and partial execution of the system	An intermediary computer was used to test the system before uploading it to the server.
The daily meetings of the team to test the system proved to be difficult to organize.	Defining short and concise evening meetings with a moderator. Daily virtual 15-minute long meetings.
The constant growth in the requirement in each increment included considerable modifications in the system.	It was tried to obtain all the possible requirements to correctly design the system.

Selection of the software to be used. Based on the technologies analyzed in the search for information stage, and the listing of requirements, the software was designed

3.1.4 Design and Modeling of the Requirements

Once the requirements were clear and the risks analyzed, it was defined the way to system would be implemented by means of a General Diagram.

The general diagram of the *Sistema para el seguimiento y control de proyectos de investigación* (*System for the monitoring and control of research projects)* is shown in Figure 2. It comprises eight subsystems; from the information that manages and controls the server to the user's interface that integrates the functionality and behavior of the system.

3.1.5 Development, Testing and Implementation

It was at this stage that the physical prototype of the system was created, which approves the manipulation of the information and the registry of the information in need of evaluation, from both the process and the collaborative work, as well as from the software.

In order to achieve that, the eight subsystems were divided into three increments so a follow up of the correct process distribution of the evaluation was possible and that the client could be given functional versions of each increment. In the first increment, the base for the administration module was considered, as well as the installation of the required software, the creation of the databases and the access menu.

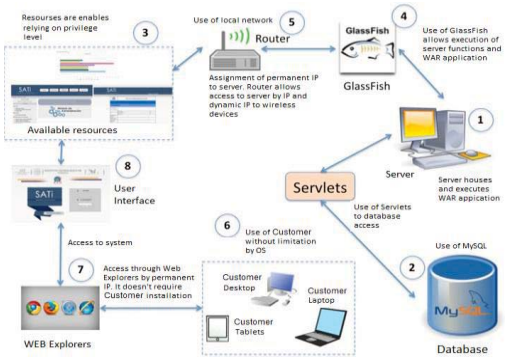


Figure 2. General Diagram of the *System*.

The second increment was mainly focused on the evaluation module, which complements the administration one and starts the management module. The third increment defined to include in the system the functions that allow the evaluation of the projects results, similarly, it includes the option that enables uploading files to the system for future reference. This phase can be seen in Figure 3.

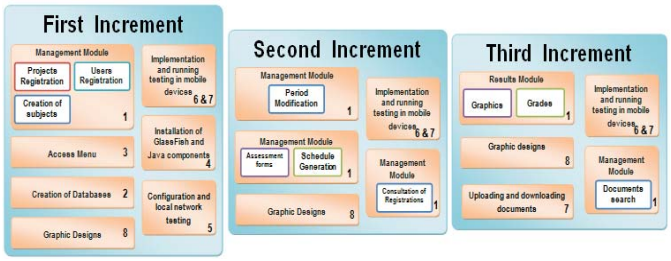


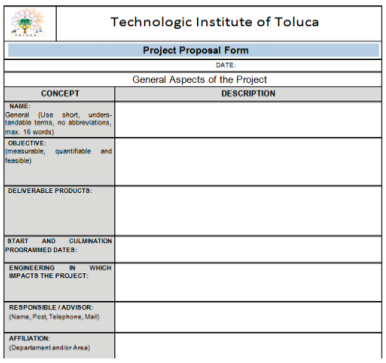
Figure 3. Diagram of the structure of the *Projects Registry System.*

Once the analysis of how the system would be implemented was done, the proposal with the defined elements was taken to the client. The access menu is shown in Figure 4.

Figure 4. Access Menu Design.

For the evaluation process several formats were created. As an example, the *format for projects proposal* designed for professors and assessors is shown in Figure 5. This format includes information on who is the person proposing the idea of the project, the commitments acquired for the completion of the project timewise and which academic activities will be carried out during a semester to achieve the objective. This assessment instrument complies with the function of creating a database of projects proposals, from which students can select the project that best adjust to their personal needs and knowledge.

Another particularity of the *system* consists of the graphic visualization of the results (Figure 6), to generate results graphs and control tables which promptly show the statistics and evaluations results using graphs, as well as the advancements and stages each of the projects currently is, from the beginning to the end. The information obtained can be accessed by project, team, group, assessor, subject, etc., depending anyone's needs.



Technologic Institute of Toluca	
Project Proposal Form	
DATE:	
General Aspects of the Project	
CONCEPT	**DESCRIPTION**
NAME: General (Use short, understandable terms, no abbreviations, max. 16 words)	
OBJECTIVE: (measurable, quantifiable and feasible)	
DELIVERABLE PRODUCTS:	
START AND CULMINATION PROGRAMMED DATES:	
ENGINEERING IN WHICH IMPACTS THE PROJECT:	
RESPONSIBLE / ADVISOR: (Name, Post, Telephone, Mail)	
AFFILIATION: (Departament and/or Area)	

Figure 5. Proposal of the initial questionnaire.

At this stage different methodologies were used. In each Subject it was established that in every project the process would begin following the recommendations of the incremental-evolutionary life cycle model, which establishes the definition a set of tasks grouped in small repetitive stages (or increments), that begin with the analysis and end with the instauration and approval of a product.

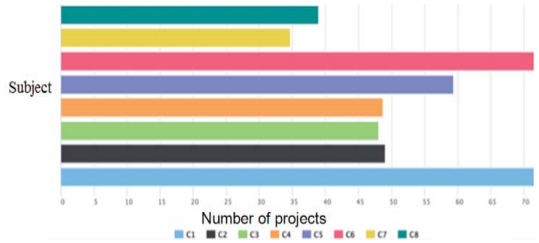


Figure 6. Graphics proposed in the design.

In each of the increments it was established that the tests and the necessary documentation must be included, compulsorily. These tests and documentation would allow each team to comply with all the objectives proposed from the beginning of their projects (as established in the questionnaire in Figure 5). Later, in each project the client and the users had the responsibility of validating each liberated product (or increment) in the established period, similarly, they were responsible of providing the team with feedback to verify and validate that the product complied with the quality characteristics previously set.

Besides the incremental-evolutionary model, the development of the app was controlled applying three more methodologies: XP (*eXtreme Programming*) for the development of the application increments, Kanbam in the definition, the assignment of roles and of the responsible team members of every activity and SCRUM for the organization of tasks and resources allotment.

4 Testing and Results

The research projects monitoring and control process has been automatized, as of today, in 80% in comparison to the conventional method. Registration of various projects and tests have been carried out, which allowed confirming the systems' functionality as well as its efficacy, compatibility and performance. The metrics used for comparing the before and after of the system implementation were the following: time to register and evaluate the project, time to generate a schedule, time for the projects search, time required for storing the registrations, the use of the processor to identify the response time, simultaneous connections and compatibility between devises of different platforms. Next are shown some of the results obtained.

Time to generate schedules. The time taken for the generation of an assessment schedule has been reduced; previously it would take from 1 to 3 hours to generate a schedule using the conventional method, now it is possible to generate a schedule in between 1 and 5 seconds (see Figure 7), this is 99% faster.

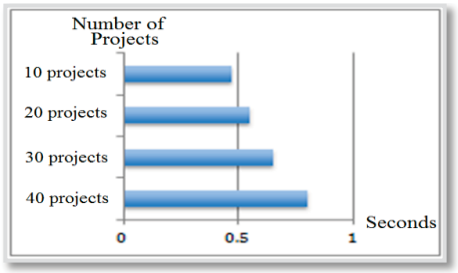


Fig 7. Time to generate ratings indicated, in seconds.

Time for the search of projects. The search of projects is one of the tasks that has been improved in the automatization of the process, now it is possible to consult a project more easily and faster. With the conventional method it was necessary to look up the project manually, which takes between 1 and 5 minutes, with the automatized method only between 1 and 15 seconds, which is about 90% faster. To search for a project, if the user so wishes, filters by subject can be used, and the results will give several results and search by reading one by one, or the search engine included could also be used. Figure 8 shows the result.

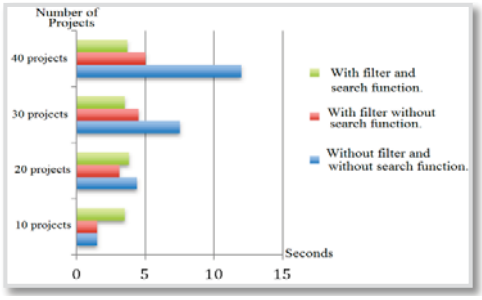


Fig. 8. Test time for searching projects, in seconds.

As it can be observed in Figure 8, the time increases depending on the number of projects since the user must read the title of each of the projects until the needed one is found. The function of the filter along with the search option reduces considerably the time, however, its real effect can be observed when the number of projects registered is larger than 30.

Time required to store registrations. With the conventional method it was necessary to have access to a physical space where to keep the projects files; with the automatized method these files are kept digitally, saving space and avoiding paper waste.

In this stage some tests related to the amount of storing memory required for the project were carried out. Figure 9 shows the amount of memory required, it was considered that each project has all the documents required for its evaluation.

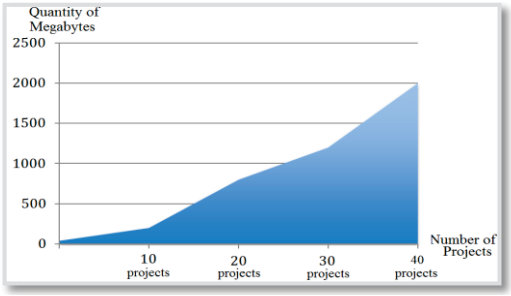


Fig. 9. Megabytes required to store projects

Use of the processor to identify response time. A very important test was the one that identifies the number of connections the server may receive before it overcharges the CPU and hence affect considerably affect the system's performance, which directly has an effect on the user's experience when using the system. It is important to mention that a connection is any device connected to the system. Figure 10 shows the CPU's performance resulting from the number of connections.

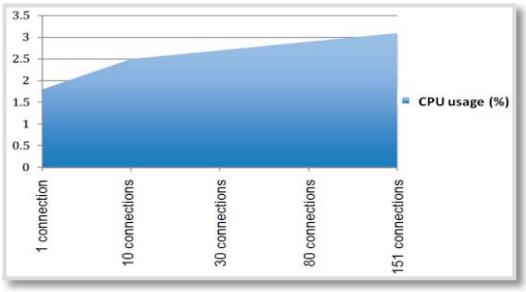


Fig. 10. CPU performance based on the connections in the database.

Compatibility between devices using different platforms. Different compatibility tests between iOS and Android running mobile devices were carried out. The objective was to verify that the resolution did not affect negatively the visibility or resolution; these devices access the server using Windows 8 only.

Another compatibility test carried out consisted of verifying the operation of the *system* in two operative systems.

For validation reasons, the server was installed in a computer running under Windows 8, and in another computer running

under Mac Os X 10.10. The results obtained during the compatibility tests in the different devices were successful.

Once the resolution was verified, the correct performance of each of the system's functions with the different devices used was checked. The functions consisted of the registry of projects, projects consultation, projects evaluation, generation of schedules, graphing, generation of grades, administration functions, for instance; adding a new subject or new user, the allotment of projects, changes authorization, saving projects, documents consultation previously uploaded in the system and uploading monitoring files. A survey to the client and the stakeholders was carried out in order to validate independently each increment and the results of the integration of everything in the system. Similarly, when the term concluded, approximately 400 surveys were done to professors and students whose main objective was an evaluation of the system. A report was crafted using the information from the surveys, the results were compared to the work model previously used (conventional model), resulting in a 100% improvement in terms of functionality, 90% in efficiency, 85% in monitoring and control of research projects, 87% in the documentation of projects and 92% in the counseling offered to students.

5 Conclusions

The result of this research shows that after having validated and ensured the good performance of the system for the monitoring and control of the research projects, following a framework aligned to the recommendations set by the software engineering, different competencies in the students are also developed, such as researching, reading, writing, improving oral and written production both in Spanish and in English. This has allowed to determine the trend to this activity in the last two semesters, highlighting characteristics such as the monitoring of the projects in a logic and constant manner, the obtention of products in less time compared to the conventional method, from this derived journal publications, the participation in different events and callings, and more importantly, the support provided to students in the increase of the graduation rates. These tools allow the development of complementary competencies for their working performance.

As a future project, on this same line, the corpus of projects will continue to increase until a large number of projects is achieved to continue studying and perpetuating the research work.

6 References

[1] Gutiérrez Estrada C. & Díaz Zagal S.: "Methodology to associate the Product Design and Project Management processes in a common platform", The 2010 IEEE International Conference on Information Reuse and Integration. Las Vegas, Nevada, USA, ISBN: 978-1-4244-2660-7, (eg.108–122), 2010.

[2] Blanchard B. S., WJ. Fabrycky. Systems Engineering and Analysis. 2nd Edition, Prentice-Hall, Inc., Englewood Cliffs, New Jersey. (1999).

[3] Software Engeneering: A Report on a Conference ponsored by the NATO Science Comité. Naut, P. y B. Randell. NATO, 1969.

[4] Boehm, B. (1998). A spiral model of software development and enhancement, Computer, 21(5), 61–72.

[5] Pressman R. "Software Engineering: A Practitioner's Approach". Mc Graw Hill. Hardcover-Part Four Managing Software Projects. 2005.

[6] Sommerville I. "Software Engineering" (9th Edition). Chapter 2 & Chapter 3. ED Pearson Addison-Wesley. United States Of America. 2011.

[7] Gutiérrez C. Méthodes et Outils de la Conception Système couplée à la Conduite de Projet. Thèse de Doctorat. LESIA-INSA. Toulouse France. 2007.

[8] Gutiérrez C., Díaz S., Reyes I., Baron C., Bartolo R., De La Rosa J., Villanueva M. "Modelado de los Procesos para el Seguimiento y Control de Proyectos de Investigación, en el Sector Educativo, utilizando redes de Petri". Revista de Sistemas y Gestión Educativa. Vol.2, No.5, p.p. 976-983. Revista trimestral ECORFAN. ISSN-2410-3977. CONACyT/RENIECYT. 2015.

[9] ISO 9001:2015. ISO 9001 Quality Management Systems. Last visited October 16th, 2015. Available from http://www.iso.org/iso/ iso9001_revision.

[10] CACEI. Consejo de Acreditación de la Enseñanza de la Ingeniería Superior, A.C. Last visited 21st September, 2015. Available from http://cacei.org.mx/

[11] CONAIC Consejo Nacional de Acreditación en Informática y Computación A. C. Last visited June 4th, 2015. Available from http://www.conaic.net/

[12] Universidad de Autónoma de Guadalajara (2015). Características generales de un proyecto de prototipos. Last visited June 4th, 2015. Available from http://crecea.uag.mx/opciones /prototipo.htm

[13] Universidad de Sonora (2015). Procedimiento para el registro de proyectos de investigación. Last visited June 4th, 2015. Available from: http://www.dcea.uson.mx/? page_id=336

[14] Universidad Nacional Autónoma de México (2015). Gradus. Last visited June 4th, 2015. Available from: http://sistemas.acatlan.unam.mx/titulos/ accesolicenciatura.aspx

[15] Universidad Nacional Autónoma de México (2015). Sistema de Titulación. Last visited June 4th, 2015. Available from: http://titulacion.ingenieria.unam.mx

[16] Instituto Tecnológico de Villahermosa (2015). Sistema de control de trámites de titulación. Last visited June 4th, 2015. Available from http://cc.itvillahermosa.edu.mx /sys/estpro/scott2/.

[17] EscolarHighTech. (2015). GES educativo – Software de control escolar. Last visited June 4th, 2015. Available from http://www.escolar hitech.com.mx/geseducativo.php

[18] Bit Technologies (2015). Bit Academic Manager - Software de control escolar y administrativo para escuelas. Last visited June 4th, 2015. Available from: http://www.bittech.mx/productos/bit-academic-manager-soft ware-de-control-escolar-y-administrativo-para-escuelas.

[19] Grupo Inndex (2015). Click-Escolar –Módulos del sistema. Last visited June 4th, 2015. Available from: http://www.grupoinndex.com/ControlEscolarModulosSistema.html

[20] Boehm, B. & Turner, R. (2005). Management challenges to implementing agile processes in traditional development organizations. IEEE Software, 22(5), 30–39.

New EVM Constructs to Support Agile Development Projects: Human Performance and EVMS

James A. Crowder and Valerie Bernard
Executive Training Centers

Abstract – *The Earned Value Measurement System (EVMS) has become a mainstay in Commercial and Government groups to measure progress and success of a project. EVMS is espoused to be an effective (albeit subjective) measure, but it does not play well with agile development efforts, due to its requirement of static schedules and work plans [10]. Here we introduce a new paradigm for EVMS that will accommodate and be affective in measuring progress and problems within agile development efforts. Included is a discussion of human performance improvement technology to address disconnects between classical EVMS and Agile Development approaches.*

Keywords: Earned Value, Agile Development, EVMS, Performance Improvement.

1. Introduction

The government instituted the formal practice of Earned Value in the 1960s as a methodology for program/project management in terms of scope, cost and schedule. Earned Value promises to provide accurate measurements of program/project performance as well as identifying problems; a crucial component of program/project management [15]. The basic 11 precepts or elements of the Earned Value Management system are:

1. Define Authorized Work Elements
2. Identify Program Organizational Structure
3. Integrate the Work Breakdown Structure (WBS) and Organizational Breakdown Structure (OBS).
4. Schedule the Work
5. Identify Products and Milestones
6. Set Time Phased Budget
7. Record Direct Costs
8. Determine Variances
9. Sum Data and Variances
10. Manage Action Plans
11. Incorporate Changes

By the late 1980s and early 1990s, the EVMS became a mainstay tool for managing, measuring, and executing programs/projects among the Department of Defense (DoD) and their respective Defense Contractors, Department of Energy (DoE), and NASA [6]. Since then many large commercial companies have adopted EVMS as well, like Boeing Commercial Airplane Division [12]. There are many Earned Value COTS software packages available for classical Earned Value. Some of the most popular ones are Microsoft Project©, Open Plan©, and Deltek wInsight©. These products are geared toward helping to plan, measure, analyze, and execute the classical waterfall development methodology [1]. Figure 1 illustrates this process, which includes measurement and analysis of earned value metrics.

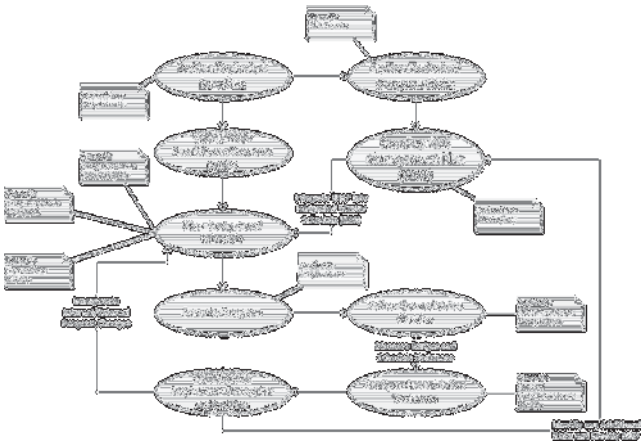


Figure 1. Classical Waterfall Execution

Contrasting Figure 1, Figure 2 illustrates the changes associated with creating a similar execution plan for agile development programs/projects. As you can see from Figure 2, the flow is quite different, and includes the recursive Sprint development process that continually refines the requirements as the Sprints progress [14]. The emphasis at the end of each Sprint is on working software that integrates together at the end of each Sprint. Customer input is sought and the Sprint plans and capability plans adjusted, based on the Sprint perspectives, integration and test, along with customer input. Included in the agile project execution process is the ability for the Sprint teams to self-organize for each Sprint; team members taking on different roles across the Sprints based on their capabilities and expertise [9].

2. Assessing EVMS & Agile Development

Earned Value and the Earned Value Management System (EVMS) provides cost and schedule performance metrics that, if handled carefully and honestly, can be useful in helping the program/project manager track the progress and

get early indications of problems. Basically, Earned Value measures whether you have earned the right to spend that much money, and whether you have earned the right to spend that much schedule [59]. Some of the metrics that Earned Value uses the measure program/project progress are:

1. **BCWS**: *Budgeted Cost of Work Scheduled*. This represents the "planned" value of the work scheduled for a given time period.
2. **BCWP**: *Budgeted Cost of Work Performed*. This represents the cost (from the original budget) of the work that was actually performed during a given time period.
3. **ACWP**: *Actual Cost of Work Performed*. This represents the actual collected costs over a given time period for the work that was actually performed. This may or may not represent the amount of work that was supposed to be accomplished during a given time period.
4. **BAC**: *Budget at Completion*. This is the total cost of the program/project at completion, or the BCWS at the end of the program/project.
5. **EAC**: *Estimate at Completion*. This is the ACWP to date, plus the estimate to complete the remaining work.
6. **CV**: *Cost Variance*. CV = BCWP – ACWP, or, the Budget cost of the work actually performed during a given time period (what they work should have cost), minus what the actual costs were for the work performed during the same time period.
7. **SV**: *Schedule Variance*. SV = BCWP – BCWS. Since both BCWP and BCWS represent the same time period, a negative SV means there is still work left to do that was not accomplished during the time period, which will take more time (i.e., schedule) to work off the remaining tasks. While it is possible for SV to be positive, which means there was more progress during the time period than was scheduled, this is the stuff of Earned Value folk lore [2].
8. **VAC**: *Variance at Completion*. VAC = BAC – EAC. This represents the complete variance for the program/project at the conclusion. Again, the goal is to have VAC as close to zero as possible, for if VAC is large positive, then the program/project was grossly over-budgeted, while a VAC that is large negative indicates a grossly under-budgeted program/project. Both are hazardous because it calls into question the company's budgeting practices.

If you read through metrics 1-8, they seem like reasonable measures of a program/project. However, the issues with classical Earned Value is that the entire program/project must be planned out in detail, often down to 2-4 week tasks, and detailed budgets put in place for the program/project schedule. Any variances from this budget or schedule are considered problems (variances) and variance reports must be written and explained; in short, in classical Earned Value change is ***bad*** and uncertainty is ***worse*** [3]. In fact, the concepts of classical Earned Value can be broken down in to seven major precepts:

1. Plan all work to completion.
2. Break down the work scope into finite pieces assigned to responsible persons for control of technical, cost, and schedule objectives.
3. Integrate work scope, cost, and schedule objectives into a performance baseline to measure progress against. Control changes to the baseline.
4. Use actual costs incurred and recorded in accomplishing the work performed.
5. Objectively assess accomplishments at the work performance level.
6. Analyze variances from the plan, forecast impacts, and prepare an EAC based on current performance.
7. Use Earned Value metrics to assess management processes.

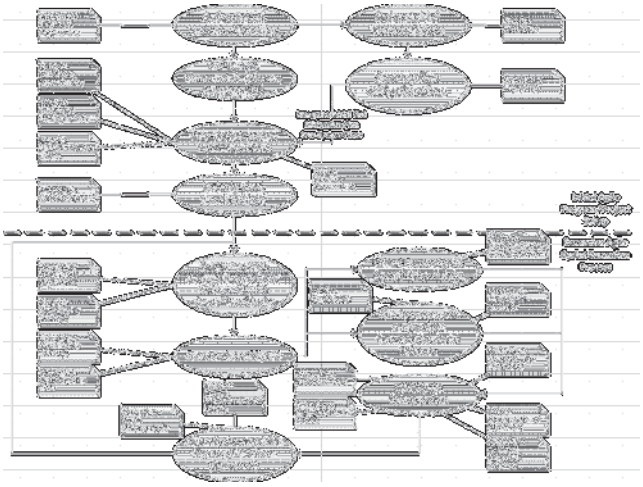


Figure 2. Agile Program Execution

In short, plan every detail of the project, including the work to be performed at every small increment, and create a detailed and complete schedule and budget across the entire project. Manage change carefully, for change is the *hobgoblin* of Earned Value. In classical development efforts, working software is delivered at major milestones.

3. Disconnects between Classical EVMS and Agile Development

Now let us bounce the precepts of classical Earned Value against the precepts of agile software development to see if there may be some issues.

1. The emphasis is on early and continuously working software deliveries at the end of each of the Sprints.

2. Constant customer interaction and collaboration that includes welcoming changes to requirements. This allows the customer to adapt to changing environment and user needs to create products and services that are considered viable by the end users.
3. Business development, management, customers and developers MUST work together throughout the project.
4. Sprints teams should be staffed with motivated individuals who are trained in both agile development and *agile team dynamics*.
5. Management needs to create an effective team environment and support the teams by being a facilitator and trusting the teams to develop the required software.
6. The most efficient and effective method of cooperation and collaboration within an agile development team is face-to-face conversation – even if it is over a Video Teleconference (VTC).
7. Working software and team/software ***entropy*** are the primary metrics.
8. Agile development processes promote sustainable development.
9. Continuous attention to technical excellence and good design enhances agility and promotes healthy cost and schedule metrics.
10. Simplicity is essential in agile development – work for work sake has no place in agile programs/projects.
11. The best architectures, requirements, and designs emerge from well-trained, self-organizing teams.
12. Teams must reflect at regular intervals (Sprint introspectives) on how to become more affective. The team must then tune/adjust its behavior accordingly.

The notion of detailed planning of every task in the program/project across the entire schedule and striving to control and drive down changes is completely antithetic to the precepts of agile software development. In agile development, change is welcomed throughout the project. The entire reason agile development was created was to deal with the reality that requirements and necessary capabilities change over time, especially for a project that spans years. In today's environment where technology, customer needs, geo-political, and cultural needs change rapidly, the need for embracing agile will only increase over time. The successful companies are those that not only embrace the mechanics of agile development, but are those that understand the need for management and developers with the non-technical skills (soft people skills) necessary to empower and facilitate efficient and motivated agile development Sprint teams. One of the most important things to understand in today's environments, is that it is possible to come in completely on budget and on schedule and yet the project fail because the program/project development did not adapt to changing customer needs. If no one wants the product once it's completed, it was not a success. Likewise, if the program/project comes in on schedule and on budget and meets customer needs, but your developers never want to work on a program/project with that manager ever again, the program/project failed. Table 1 below illustrates classical EVMS verses the concepts for Agile EVMS.

There are very many factors that can derail agile development teams and lead them to failure; the most prevalent are those revolving around a lack of management commitment and training in how to manage an agile program/project (i.e., how to be an agile manager). Even more issues arrive when managers must embrace a new paradigm of how to measure agile programs/projects, or how to use Agile Earned Value [26].

Table 1 – Classical vs. Agile EVMS

Classical EVMS	Concepts for Agile EVMS
Plan all work to completion of the program/project	Create capabilities backlog and loosely plan across Sprints
Break down work scope into finite pieces assigned to responsible person(s) who control technical, cost, and schedule objectives.	Work is assigned to Sprint teams who control technical content of the Sprint backlogs.
Integrate work scope, cost, and schedule into a performance/program baseline. Control changes to baseline	Create program/project backlog burndown plan, assigning capabilities across Sprints and teams. Results of demonstratable software at the end of each Sprint defines how requirements/capabilities change across the program/project.
Use actual costs incurred and recorded in accomplishing the work performed.	Agile teams assess performance characteristics of the teams and tune the teams needs and performance to improve over time, throughout the entire agile development cycle.
Objectively assess accomplishments at the work performance level	Assess accomplishment by continuously integrated working software.
Analyze variances from the plan, forecast impacts, and prepare EAC, based on current performance	Analyze, based on working software, new or changes in requirements for future Sprints. Assess efficiency and effectiveness, which includes volitility of teams and software, based on entropy measures.
Use Earned Value metrics to assess management process	Use Agile Earned Value metrics to assess effectiveness of both management and Sprint teams.

What follows is a discussion of the factors that can most easily derail an agile development project, from an Agile EVMS perspective:

1. **Lack of accountability**: This pertains both to the members of the agile development teams and the agile manager. Many managers may feel like they have nothing to do, given the autonomy and control that the agile team need to have over the development efforts. In this case, the manager may feel like they are no longer accountable for the project, and therefore will not facilitate the agile teams, becoming apathetic toward the entire process. In this case, the program/project has very little chance of being successful. If the teams are not chosen well, some team members may feel like they are not individually accountable and that it's the teams' responsibility, not theirs, to make sure things work well. Individual accountability to the teams is crucial to the overall success of agile development.
2. **Lack of commitment**: to Agile (holding on to classical EVMS): The manager that insists on using classical waterfall development Earned Value and management techniques on an agile development effort will not only

be unsuccessful, but the manager will be very frustrated throughout the entire effort. However, this requires commitment from upper management to provide the proper management training on agile projects.

3. **Poorly trained teams**: Just being efficient at writing software and being adaptable to changes doesn't mean agile teams are successful. Agile development Sprint teams need to be trained in how to collaborate effectively, how to deal with generational, cultural, and other differences that can cause mistrust among team members. Understanding the teams' personalities can go a long way toward the teams self-organizing in a way that allows the team go be effective across multiple Sprints and multiple programs/projects.
4. **Poor documentation**: Many developers feel that agile gives them the freedom to not worry about documentation; that documentation gets in the way of their freedom to self-organize and adapt. However, the right amount of documentation is essential in order for the team members and teams to understand the end goals, and to understand what each other is currently developing, how it fits into the Sprint, and how the Sprints will integrate together to form working software at the end of each Sprint.
5. **Using unproven collaboration/automation tools**: As we have discussed, providing productivity tools to the teams is necessary to keep the individual developers and the Sprint teams running at peak efficiency and can promote collaboration. However, introducing new tools into the teams during a development effort may completely disrupt the rhythm of the agile development process while each team member comes up to speed on the tools and how to use them effectively. In addition, if it turns out the tool is not appropriate for the teams, additional efficiencies will be lost when teams and individuals try to re-adopt previous tools.
6. **Inaccurate data**: It is vitally important that the Agile Manager gather accurate data concerning the productivity and effectivity of the teams across Sprints. Retrospectives are difficult if the teams are not provided accurate information.
7. **Manager holding everything at their level** (failure to communicate issues to the teams): While inaccurate data causes incorrect decisions to be made among the teams and between the teams, the lack of information is more devastating to effective agile development efforts. There must be complete transparency between the Agile Manager and the teams, the Agile Manager and individual developers, and between Sprint teams. The adaptivity the agile development process promises in only achievable if there is effective communications all throughout the program/project.

4. Human Performance Technology

The International Society for Performance Improvement defines human performance technology as "a systematic multi-disciplinary approach that stresses rigorous analysis of present and desired levels of performance, identifies causes of performance gaps, offers a wide range of interventions, guides the change management process, and evaluates the results." In order to solve costly business challenges, human performance technology utilizes an approach with proven models and methodologies which results in interventions to address the business challenges. As with the Earned Value Management System process, before deciding which techniques are appropriate, the process of human performance technology begins with analysis to discover multiple intervention opportunities. Pershing [17] states, "Human performance technology is the study and ethical practice of improving productivity in organizations by designing and developing effective interventions that are results-oriented, comprehensive, and systemic" (p. 6). The International Society for Performance Improvement [17] suggested improving the output of a company contained three components: cause analysis, performance analysis, and interventions. Stolovitch [16] defines human performance as "a field of endeavor that seeks to bring about changes to a system in such a way that the system is improved in terms of the achievements its values." These definitions make clear that Human Performance Technology aims at improving human organization results and, therefore, can provide a framework when it comes to closing the gap in performance.

4.1 Four Principles of Performance Technology

The International Society for Performance Improvement [18] suggested that human performance technology provides a guide for "systematically identifying and removing barriers to individual and organizational performance." Utilizing human performance technology framework as a starting point for EVMS and Agile Development Projects requires an understanding of the four principles performance improvement consultants recognize as a valuable guide and can be expressed as RSVP:

1. **R**--Focus on results
2. **S**—Take a system viewpoint:
3. **V**—Add value:
4. **P**—Establish partnerships:

Results from a performance consultant view is typically expressed in valuable, measureable results. Taking a system viewpoint considers the entire performance system during analysis. In order to take a system viewpoint, short and long term change, resource limitations and competition should be considered. By producing results which impact individuals and organizations in a positive way, value is added. Partnerships are a critical component to performance and bridging the gaps already identified between EVMS and Agile Development. The key is everyone working toward the same goal horizontally across the organization. The challenge remains within organizations who approach

performance from a vertical viewpoint in separate functional areas. Human Performance Improvement consultants understand results are the most critical component of the value they bring when they look at the performance system.

4.2 Gilbert's Behavioral Engineering Model

O'Donohue and Ferguson [19] suggested from Skinner's work, "Behavior is best influenced by rewarding acts that most closely approach the desired behavior." Thomas Gilbert was a student of B. F. Skinner. During his research in the 1960's and 1970's, Thomas Gilbert was interested in further understanding human behavior [20]. Thomas Gilbert [21] suggested, "For any given accomplishment, deficiency in performance always has as its immediate cause a deficiency in behavior repertory (P), or in the environment that supports the repertory (E), or in both. But its immediate cause will be found in a deficiency of the management system (M)" [21]. The Behavior Engineering Model (BEM) developed by Gilbert [21] provides organizations a way to identify factors that contribute to improved performance. Presented in Gilbert's book, Human Competence: Engineering Worthy Performance, [21], the Behavior Engineering Model provides a way to engineer and to troubleshoot performance for both the individual and the organization by looking at individual factors and environmental supports that either increase or decrease performance. Figure 3 illustrates this.

Gilbert's [21] BEM focuses on six key factors that are clustered in two groups: data, resources, incentives (grouped into Environmental Support), and knowledge, capacity, and motives (grouped into Person's Behavior Repertory) [22]. Gilbert [21] suggested a person's repertory of behavior (P) are individual characteristics of a person that they bring to their jobs.

	Information	Instrumentation	Motivation
Environmental Supports	*Data* 1. Relevant and frequent feedback about the adequacy of performance 2. Descriptions of what is expected of performance 3. Clear and relevant guides to adequate performance	*Resources* 1. Tools and materials of work designed scientifically to match human factors	*Incentives* 1. Adequate financial incentives made contingent upon performance 2. Non-monetary incentives made available 3. Career-development opportunities
Person's Repertory of Behavior	*Knowledge* 1. Systematically designed training that matches the requirements of exemplary performance 2. Placement	*Capacity* 1. Flexible scheduling of performance to match peak capacity 2. Prosthesis 3. Physical shaping 4. Adaptation 5. Selection	*Motives* 1. Assessment of people's motives to work 2. Recruitment of people to match the realities of the situation

Figure 3. Gilbert's Behavioral Engineering Model.

4.3 Behavioral Engineering and EVMS

To reach maximum performance within agile development teams (i.e., maximize EVMS), Gilbert's environmental factors must be provided [21]. The question we must ask initially when thinking there is a training issue are related to the organizational factors in Gilbert' BEM Model: information, resources, and incentives. While training programs may address the issue, many questions need answered first:

1. Are we clear regarding our expectations for our teams?
2. Are the processes clearly defined?
3. Do we have job-aids in place?
4. Does the environment support the work the employees are required to do?
5. What do we need to change?

Chevalier [23] goes into greater detail when it comes to leveraging results from the information, resources, and incentives factors (see Figure 4).

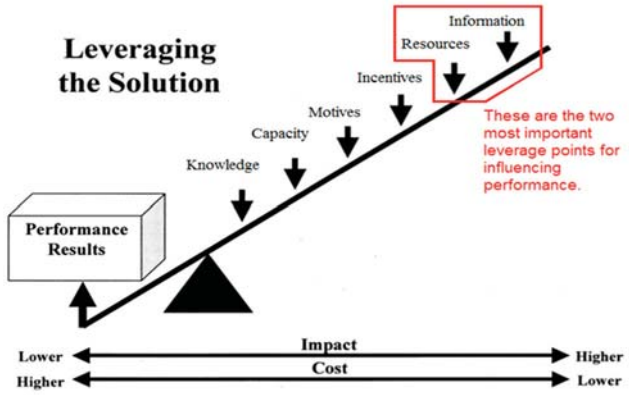


Figure 4. Behavioral Engineering Model

Human task performance requires highly effective systems, and EVMS and Agile Development is highly complex and must take into account human performance. In most cases; however, human performance is not a result of a characteristic or behavior flaw but more a consideration of a poorly designed system. As Rummler & Brache suggested [24], "if you pit a good performer against a bad system, the system will win almost every time." Behind every request for training and development, there is most generally always a larger human performance issue.

5. New Agile EVMS Metrics

In his paper on Assessing Agility [8], Lappo asserts that classical metrics are not much use for assessing agility. He goes on to explain that the use of metrics like Software Lines of Code (SLOC), function points, or quality metrics are not an effective measure of agile development, and assessments made should be made in terms of how the software, as well as the software development process, are effective in meeting the needs of the program/project, the customer, the end Users, and the overall companies business goals and visions.

This does not mean that these metrics are not useful

throughout the program/project to help with the overall agile development efforts. For instance, complexity measures are useful in determining which capabilities from the backlog are scheduled in a given Sprint, and are useful in determining how to "swap out" one set of capabilities for another when it is determined that a set of capabilities must be moved forward or moved out within the overall Sprint development schedule. However, complexity measures tell very little about the maintainability of the code; how easy is it to adapt the code for other purposes (i.e., how agile is the code). Some very complex code may be written very easy to understand and structured in such a way that it fits easily into the agile development style. At the same time some very simple code can be written in such a convoluted way that it is almost impossible to understand, modify, or maintain. Lappo's view [8] is that these low-level measures don't measure or provide insight into higher-level effectiveness and efficiency measures of the overall program/project's agile process.

Agile Earned Value metrics must take into account the entire agile development life-cycle, which includes assessments of the software, the program/project agile process, the environment that has been created for the developers and development (Sprint) teams, as well as assessment of the tools utilized in the agile development process. In order to effectively measure agile development in terms of Earned Value one must take into all of these factors, for each of them drives cost, schedule, and quality across the entire agile development program/project. Assessing software in terms of complexity may not provide a high-level view of overall program effectiveness, but according to Abran [2], it is an essential characteristic of the agile software process and product and should be measured. According to Cambell [5], capturing and utilizing context in such measurements is essential to capture the overall measure of complexity. Software complexity, combined with context, allows the Agile Manager to measure the computational, structural, functional, and representational complexity of the software throughout the agile development lifecycle. Abran [2] explains that measuring computational complexity (CC) provides classical Earned Value measurements of CV and SV, as it quantifies the time and resources required to write and test the software. This may be measured in terms of algorithmic efficiency of the software, coupled with the efficiency measure of each Sprint. Looking at the integrated, working software at the end of each Sprint from high-level dynamic event traces that are required to achieve the functional requirements of the system allows measurement of the functional complexity (FC). Representational complexity (RC) is measured from systems architecture (DoDAF[1]) perspective, looking at the graphical and textual notations for representations of the System Model (SV-1), System Interactions (SV-3), and System Behaviors (SV-4). Based on a measure from zero to one, the overall Sprint Complexity Factor (SCF) for a given team for a given Sprint is:

$$SCF = CC * FC * RC$$

and the overall agile cost and schedule metrics, Agile Cost Variance (ACV), and Agile Schedule Variance (ASV) become:

$$ACV = CV * SCF$$
$$ASV = SV * SCF$$

The overall Agile Effectiveness (AE) of a given Sprint for ***n*** number of Sprint teams is:

$$AE = \sum_{i=1}^{n} ACV_i * ASV_i$$

The Cumulative Agile Earned Value (CAEV) effectiveness measure, across ***m*** number of Sprints, then becomes:

$$CAEV = \sum_{j=1}^{m} \sum_{i=1}^{n} ACV_{i,j} * ASV_{i,j}$$

6. Entropy as an EVMS Measure for Agile Development

While some changes are embraced by the agile design methodology, it is important to measure those phenomena that drive uncertainty into agile development and are indicators of impending problems within the overall development rhythms of the agile program/project. We will discuss two of these in the next section:

1. **Volatility in Sprint team membership**: As was discussed earlier, it is important to keep the Sprint teams as stable as possible across the development program/project in order to keep a stable and sustainable development rhythm.
2. **Volatility or velocity of increase/decrease of Sprint software defects**: As the Sprint teams work together, get use to each other, understand each other's strengths and expertise, and as they gain experience writing software for this project, one would expect the number of defects across each Sprint to decrease. One way to measure this is with Entropy or the measure of change across Sprints.

Entropy is a concept in information theory proposed by Shannon [13], and generalized by Rényi [4]. Entropy is used in information theory to provide a quantitative measure of uncertainty in systems random variables. A simple way to describe the use of Entropy is to say that the more uncertainty there is in a given system, the more potential

[1] Department of Defense Architecture Framework

there is for volatility within the system. This is exactly the case we have with agile development programs/projects. However, the uncertainty here is not the uncertainty of requirements change, but the uncertainty of increase in Entropy of certain factors that drive the efficiency of agile development teams [7]. In particular we are talking about the uncertainty of teams (moving people between teams or bringing new people into teams) and the uncertainty that can be measured in the software defect volatility.

Earlier, we discussed the problems associated with changing out Sprint team members during the agile development program/project. This volatility of team members disrupts the agile development process and introduces uncertainty (or entropy) into the development efforts. In order to adequately measure the effectiveness and productivity of agile development, the Entropy of Team Volatility (ETV) must be a factor in the Agile Earned Value metrics. We will let ***X*** be a random variable that describes the probability of a change in team members across one or more agile teams, where the probability of team member volatility increases with the number of people in each team and increases with the number of teams. As the team size and the number of teams increases the probability of a change of one or more personnel increases also. We will assign an exponential random variable to the probability that there will be personnel changes, given a number of teams and number of personnel/team. Also, since removing a team member means adding a new team member, and changing out personnel from teams means moving at least two people (or an even number of personnel), for ***n*** number of changes there are *2n* people changed. Therefore the uncertainty (or entropy) equation becomes:

$$p(X) = \lambda e^{-\lambda X}, \text{ where } X = \ln\left(\sum_{i=1}^{n(teams)} size(team_i)\right)$$

6.1 Volatility of Software Defects

Volatility of software defects is a measure of whether the software defects are decreasing over time, given that over time, as the developers become more familiar with the overall system being developed and how all the services play together, and become more comfortable with the teams and team environment. For a given set of capabilities within each successive Sprint, if the complexity between the Sprints is normalized, one would expect the software defects to be decreasing. An increase in the normalized software defects over successive Sprints is increasing, this indicates volatility or Entropy in the development process and must be measured and remedies determined and put into place across the Sprint teams [25].

The Software Defect Entropy is determined and measured, based on the first Sprint, setting the complexity factor for the first Sprint equal to 1. Then the complexity of each successive Sprint is measured against the first Sprint and a complexity factor determined. Based on the software defects Sprint 1, the Software Defect Factor for Sprint ***i*** is:

$$SDF_1 = \#\, defects_1$$
$$SDF_i = \#\, defects_i * normalized\ \ complexity\ factor_i, i > 1$$

If $SDF_{i+1} > SDF_i$ it indicates Entropy has been introduced into the software development process and the causes must be determined and adjudicated in order to get the agile development effort back on track. The total Software Defect Entropy (SDE) across the agile development project is then measured, where we compute the change in normalized defects/Sprint team across each Sprint, or:

$$SDE = \sum_{i=1}^{m}\sum_{j=1}^{n} SDF_{i,j},$$

where *m= #Sprints* and *n = #teams*.

7. Discussion

Adjusting Earned Value metrics for agile development will be a long paradigm shifting exercise for managers and may take time to get the Agile Manager use to different types of metrics and measures than they have been used to. The emphasis with agile development needs to be on measuring the agile process, and results (working code), not antiquated measures like SLOC. Only when we embrace measures that are effective for agile development will the Agile Manager be able to truly understand the dynamics and issues with their agile development programs/projects. Having dealt with most of the issues surrounding the management of agile development, we move on to a subject that has been getting much more visibility in the last few years, and that is the subject of inclusiveness and diversity as part of the overall team dynamics for agile development.

References

1. Abba, W. 2000. How Earned Value Got to Prime Time: A Short Look Back and a Glance Ahead. PMI College of Performance Management (www.pmi-cpm.org).
2. Abran, A., Ormandjieva, O., and Abu Talib, M. 2001. Information-Theory-Based Functional Complexity Measures and Function Size with COSMIC-FFP. Université du Québec à Montréal.
3. Alleman, G. 2012. Herding Cats: Issues with Deploying Earned Value Management. http://zo-d.com/blog/archives/project-management-on-the-web/-pm-web-001-glen-b-allemans-herding-cats.html.
4. Beck, C, and Friedrich, A. 1993. Thermodynamics of Chaotic Systems: an Introduction. Cambridge University Press. ISBN 0521433673.

5. Campbell, J., Trapnell, P., Heine, S., Katz, E., Lavallee, L., and Lehman, D. 1996. Self-concept clarity: Measurement, personality correlates, and cultural boundaries. Journal of Personality and Social Psychology, 70, 141–156.
6. Defense Systems Management College. 1997. Earned Value Management Textbook, Chapter 2. Defense Systems Management College, EVM Dept., 9820 Belvoir Road, Fort Belvoir, VA 22060-5565.
7. Harrison, W. 2000. An Entropy-Based Measure of Software Complexity. IEEE Transactions on Software Engineering, Vol. 18(11).
8. ISO/IEC 19761. 2003. Software Engineering – COSMIC-FFP-A Functional Size Measurement Method. In International Organization of Standardization – ISO, Geneva, Switzerland.
9. Kurian, T. 2006. Agility Metrics: A Quantitative, Fuzzy-Based Approach for Measuring Agility of a Software Process. ISAM-Proceedings for the International Conference on Agile Manufacturing'06 (ICAM-2006), Norfolk, VA.
10. Marshall, Robert. 2007. The Contribution of Earned Value Management to Project Success of Contracted Efforts. Journal of Contract Management, pp. 21-331.
11. Pisano, N. 1999. Technical Performance Measurement, Earned Value, and Risk Management: An Integrated Diagnostic Tool for Program Management. Defense Acquisition University Acquisition Research Symposium.
12. Schulze, E. 2010. "How Earned Value Management is Limited". Retrieved 2013-04-04.
13. Shannon, C. 1969. The Mathematical Theory of Communication. University of Illinois Press, Urbana, Chicago.
14. Sulaiman, T. 2007. "Agile EVM -- Earned Value Management The Agile Way". Agile Journal.
15. Sumara, J. and Goodpasture, J. 1997. Earned Value -- The Next Generation -- A Practical Application for Commercial Projects. Retrieved 2006-10-26.
16. Stolovitch, H. 1982. Performance Technology: An Introduction. Performance and Instruction, 21(3), pp. 16-19.
17. Pershing, J. 2006. Handbook of Human Performance Technology. International Society for Performance Improvement. ISBN 978-0787965303.
18. Burkett, H. 2010. Applying an International Focus to Performance Improvement Opportunities. Performance Improvement, Vol. 49(7).
19. O'Donohue, W. and Ferguson, K. 2001. The Psychology of B. F. Skinner. Sage Publictions, Thousand Oaks, CA. ISBN 0-7619-1758-6.
20. Gilbert, T. 2015. eLearning – The Training Magazine Network. http://www.elearninglearning.com/thomas-gilbert/
21. Gilbert, T. 1978. Human Competence: Engineering Worthy Performance. International Society of Performance Improvement, ISBN 978-0787996154.
22. Gupta, A., Govindarajan, V., and Malhotra, A. 1999. Feedback-Seeking Behaviour Within Multinational Corporations. Strategic Management Journal, Vol. 20(3), pp. 205-222.
23. Chevalier, R. 2004. Human Performance Technology Revisited. International Society for Performance Improvement. ISBN 978-1890289188.
24. Rummler, G. and Brache, A. 1990. Improving Performance: How to Manage the White Space on the Organizational Chart. Jossey-Bass Publications, San Francisco, CA.
25. Crowder, J. and Friess, S. 2015. Agile Project Management: Managing for Success. Springer International Publishing Switzerland. ISBN 978-3-319-09017-7.
26. Crowder, J. and Friess, S. 2014. Systems Engineering Agile Design Methodologies. Publishing Switzerland. ISBN 978-1-4614-6663-5.

Configurable Method Model of Agile Methods - for Creating Project-Specific Methods

Daya Gupta[1], **Rinky Dwivedi**[2]
[1]Department of Computer Engineering, Delhi Technological University, Delhi, India
[2]Department of Computer Science, Maharaja Surajmal Institute of Technology, Delhi, India

Abstract - *The research focuses on Agile Method configuration process that supports an Essentiality attribute for the agile methods. Since, these methods adhere to a set of practices it's hard to produce a generic model for the purpose. It was noted, that software development community has adopted method configuration to form project-specific method for agile methodology but have not treated the '**notion of essentiality'** in a method. The consequence of this is that - The relationship between the original method and configured method is not fully explored. Thus, the extent to which a method can be configured, remain unanswered. This demands a full investigation into what can be configured into which method. The agile values defined in the agile manifesto are seen to define 'essentialities' in these methods. Further to configure a method for an agile project, each project is considered individually. The project characteristics provide support to the method engineer for deciding the '**variability in the methods'**.*

Keywords: Agile Methodology, Scrum, Method Configuration

1 Introduction

Now a day's software companies are extensively using agile methods. Miller and Lee describes the characteristic of agile software process as – "modularity, iterative with short cycles, time-bound, adaptive with possible new risks, incremental process approach, people-oriented and collaborative working style" [1]. These characteristics ensure the fast delivery of software projects within given time-span. Beck introduced the Extreme Programming method -better known as XP [3,4]. This is widely acknowledged as the starting point of various agile software development approaches. There are also a number of other methods either invented or rediscovered that belong to the same family of methods. Scrum [20], Feature Driven Development [6], crystal methods [7] and DSDM [8], etc are some examples of these methods or methodologies. Further these methods have a well-defined structure that includes process, practices, roles and responsibilities.

- ***Process***-Description of phases in the product-life-cycle.
- ***Practices***-They are concrete activities and work products that a method defines to be used in the process.
- ***Roles and responsibilities***- Allocation of specific roles through which the software production in a development team is carried out.

No single agile method is directly applicable to a particular project [12, 21]. Industries like Intel Shannon, IBM, Nokia have been customizing or refining the agile method based on project in hand.. To ensure that situational method confine to the principle of agility, a Method Configuration process[5] is needed where these light-weight methods can be configured by adapting an existing agile method or extends it by adding new practice or combine practises of two methods. In our earlier previous researches , we had presented a fuzzy rules to evaluate the suitability of practices of agile methods in order to configure project specific methods [9, 11]. This paper presents an **Agile Method Configuration Process,** to form situation specific method.

The novel contribution in the paper is to develop agile methods as agile configurable models (see section 2). Similar to traditional configurable models, agile configurable models also supports an *Essentiality* attribute; this essentiality attribute can take two values either *common* or *variable*. The agile values defined in agile manifesto along with practical and theoretical experience of various developers and academicians forms the basis for defining the commonality and variability in these methods. The organisational characteristics are used to select the most appropriate agile method [9]. The method is further configured to form project-specific methods [10]. The research offers to use project characteristics for deciding the inclusion of variable constituents in the configured method (section 3). The process is illustrated with the help of a case study (section 4).

The next section will define the essentialities in agile methods and presents the configurable model of various agile methods.

2 Essentialities in agile methods

A method has two aspects- **product and process**. The product aspect provides features for the product development whereas; process aspect is the route that needs to be followed to ensure the efficiency of the product development. The literature survey on various agile methods reveals that there exist many significant operational differences between the

process aspects of these methods. Thus it is difficult to produce a generic model of an agile method configuration process with sufficient granularity to be useful for the purpose. This moved the research, to the practices or the product aspect of these methods.

The agile practices are centred on the ***Agile values*** defined in [2]. To preserve agility, all popular agile methods found in literature, satisfies these agile values. So the present research considers these agile values as the basis for defining essentialities in the agile methods. In the next sub-section the paper presents agile values defined in agile manifesto and the corresponding method practices.

2.1 Mapping between agile values and agile method practices

The four core agile values defined in the manifesto are [2]:

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration.
- Responding to change.

The practices of agile methods are divided into four groups corresponding to the four core agile values. Table 1 adapts from [17] shows the mapping between the practices of popular agile methods with the agile values. This mapping shows the support of agile values by agile methods. The mapping between the agile values and the practices of agile methods **provide a guideline to define the *essentialities* in the method.** The guideline is presented below:

Guideline: *To satisfy agile values, at least one practice corresponding to an agile value must be considered as common to the method.*

Since the guideline is defined at a higher level it needs to be explored further to identify - **'commonality among the group of practices corresponding to an agile value'.** For the purpose, the practical and the theoretical experiences of various software developers and users are gathered and examined. The next sub-section presents the major outcomes of the research, used to decide the *essentialities* in method practices.

2.2 Determining the essentialities in agile methods

The widely accepted agile methods- Extreme Programming, Scrum and DSDM were introduced in the early and mid 1990's and have been found well documented. There exists a number of literature and experience support for them. Other methods that are also included in this research are FDD, crystal and ASD. However, less is known about their actual usage in real world but these methods have maintained their own interest and active research by user. Thus, they can be classified as "active" and are thus included in this research.

These methods have a well defined process and a set of practices that need to implement the process. To avert a repetition of arguments in the research and to present the effort contextually, the paper avoids exhibiting a review of the process, practices, roles and responsibilities of all the above methods. However, only the relevant points are briefly discussed and are presented in a nutshell. Interested readers are referred to [15] to get a detailed overview on the agile methods.

Extreme Programming (XP)

XP has evolved from the "problems caused by traditional development models" [3,4,14,17] to well documented on the key principles and practices used. The Beck, defined that key features of XP are - customer driven development, small teams, daily builds. The special features that makes it distinct from others is '*refactoring*'and code style. [12] found that developers at Intel Shannon formed a customized method, of XP. They took pair programming, testing, metaphor, collective ownership, refactoring, coding standards and simple design as the part of the customized method formed. Leaving behind planning game, small release, continuous integration, 40-hrs week and on-site customer. The customized method thus, formed behaves extremely well as compared to the original method. Similarly, in the literature another case study by [18] was found support the configuration of this method.

Thus, from the practical experiences and the available literature on XP, the essentiality of this method is defined as:

Common = {pair programming, testing, the planning game, metaphor, refactoring, coding style}. Variables = {collective ownership, on-site customer, short releases, continuous integration, simple design}.

The next method under consideration is scrum. The term 'scrum' originally derives from a game strategy of Rug-by where it denotes "getting an out-of-ball back into the game".

Scrum

Scrum focus on managing iterative development project having artifacts such as *Sprint*, *Scrum team* and *product backlog etc.* [20] identifies that scrum can be adopted for new project and suggests that practises *Sprint*, *product backlog* and daily Scrum Meeting are common to all projects

By personally interviewing, software developers in the HCL technologies currently working on the leading projects like banking and aviation. It was found that they consider practices like-*product backlog, sprint, sprint planning meeting and daily scrum meeting* as *common* to their process. [19] conducted a survey to verify the effectiveness of scrum for the development of mobile application. In their research they found that *sprints, product backlog and sprint backlog* as the most essential practice needed to be address during the development in this domain.

Thus, from the practical experiences and the available literature on Scrum, the essentiality of this method is defined as:

Common = {scrum teams, sprints, sprint planning meeting}. Variables = {Daily scrum meeting, sprint review, product backlog, sprint retrospective, scrum of scrums}. Similarly, the configurable models of other agile methods like FDD, ASD, DSDM and crystal can also be developed and depicted in Table 1.

2.3 Configurable model for agile methods

The agile values, and the researches, and practical experiences are examined to decide commonalities in the method practices corresponding to an agile value. Table 1 defines the commonality and variability in the methods. A 'C' corresponding to an agile practice indicates that essentiality=common for the practice and 'V' indicates essentiality=variable.

Now just as the traditional method configuration process yields a family of configured methods, so also agile method configuration process produces a family of methods. For example two configured model of XP are shown in Table 2.

3 Agile Method Configuration process

The proposed **Agile Method Configuration Process**, to form project specific method, is shown in Figure 1. Firstly the projects characteristics are gathered in the form of organisational requirements. These organisational requirements are then fed to Fuzzy Logic Controller to find the weight of the agile methods. Here the term 'weight' refers to the degree of applicability of the method for the specified set of requirements. The highly 'weighted methods' or 'most suitable methods' are retrieved from the method base. Thereafter the most suitable method is selected by method engineers amongst the retrived methods. Project characteristics provide guidelines to the method engineers for deciding the 'variability in the selected methods'. In this way the selected method is configured.

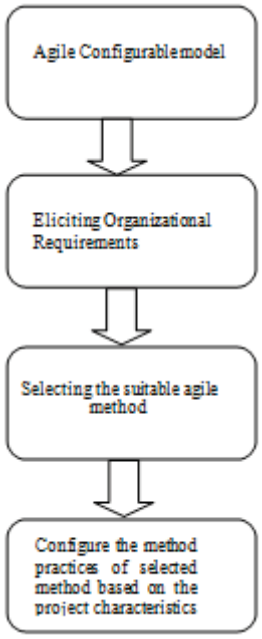


Figure 1. Agile Method Configuration Process

Table 1:- Commonality and variability in popular agile methods

Agile Values	XP		Scrum		FDD		ASD		DSDM		Crystal	
Individuals and Interactions over processes and tools	**Pair Programming** Collective Ownership On-Site Customer **The planning game**	C V V C	**Scrum Teams** Daily Scrum Meeting **Sprint Planning meeting**	C V C	**Domain Object Modelling.** Individual Class Ownership Feature Teams Inspection	C V V V	**Adaptive Management Model** Collaborative teams .JAD by independent agents Customer Focus Group reviews	C V V V	**Empowered Teams.** Active User Involvement	C V	**Holistic Diversity and Strategy** Parallelism and Flux User Viewings	C V V
Working Software over comprehensive documentation	**Testing** Short releases Continuous Integration	C V V	**Sprint** Sprint Review	C V	**Developing By Feature** Inspection Regular Builds Reporting/Visibility of results	C V V V	**Developing by Components** Software Inspection Project Post mortem	C V V	**Frequent Product Delivery** Iterative and Incremental development Integrated testing	C V V	**Monitoring of a progress** Revision and Review	C V
Customer Collaboration over contract negotiation	**The Planning Game** .On-Site Customer	C V	**Sprint planning meeting** .Product Backlog	C V	**Domain Object Modelling**	C	**Adaptive Management Model** JAD	C V	**Collaboration and Cooperation among stakeholders** Requirements are baseline at a high level	C V	**Staging** User Viewings	C V
Responding to change over following a plan	**Metaphor** Simple Design **Refactoring** **Coding standard**	C V C C	**Sprint Planning meeting** Sprint Review Sprint Retrospective Scrum of Scrums	C V V V	**Domain Object Modelling** .Configuration Management	C V	**Adaptive Cycle Planning** **Customer Focus group reviews**	C C	**Reversible Changes**	C	**Reflection workshops** Methodology Tuning	C V

Table 2: Instances of XP configured Method

	Configured Methods of XP
1.	All XP common concepts, with on-site customer and collective ownership.
2.	All XP common concepts, with simple design and continuous integration.

3.1 Selecting the suitable agile method

The method engineer is responsible for eliciting organisational requirements determined by organisational environment. Table 3 by [16] shows the set of Organisational requirements and the corresponding agile methods support that gives as an input to Fuzzy Logic Controller to find the membership metrics. Fuzzy rules are used to find the membership metrics of the methods under consideration. The purpose is to select the most suitable methods; FLC will assign membership to the methods depicting the degree of perfectness for the defined set of requirements. The details of fuzzy rules and organisational requirements are described in [10].

Table 3:- Organisational requirements and corresponding method support

Characteristics	***Values***	***Methods support***
Task extent	*Small*	*XP, SCRUM, FDD, DSDM, Crystal*
	Medium	*XP, SCRUM, FDD, Crystal*
	Large	*FDD, ASD*
	Complex	*ASD*
Group Size	*Less than 10*	*XP, SCRUM, Crystal*
	Multiple Teams	*SCRUM, DSDM, Crystal*
	No limits	*FDD*
Progress Approach	*Iterative*	*XP, SCRUM, FDD,ASD, DSDM, Crystal*
	Rapid Development	*XP, SCRUM,ASD, DSDM, Crystal*
	Distributed Development	*ASD*
Code Style	*Clean and Simple*	*XP*
	Not Specified	*SCRUM, FDD,ASD, DSDM, Crystal*
Expertise Environment	*Quick Feedback*	*XP*
	Not Specified	*SCRUM, FDD,ASD, DSDM, Crystal*
Physical Environment	*Co-located teams*	*XP, ASD, Crystal*
	Distributed teams	*XP, ASD*
	Not Specified	*SCRUM, FDD, DSDM*
Industry customs	*Collaborative and Cooperation*	*XP, DSDM*
	Not Specified	*SCRUM, FDD, ASD, Crystal*
Abstraction Mechanism	*Object-oriented*	*XP, SCRUM, FDD, ASD, DSDM, Crystal*
	Component-oriented	*ASD, DSDM*

3.2 Configuring the agile method

The significant concern regarding agile methods is that – they lack the factor of 'Discipline' [13]. These methods adhere to a set of practices rather than follow a common process for the development. *Thus it is difficult to produce a generic model of an agile process with sufficient granularity to be useful for the purpose.*

Further, to configure agile methods and to provide support for selecting *variables* in these methods. *It would prefer to consider each agile project individually rather than to provide a generic mapping between the practices and set of guidelines as is done in case of traditional methods.*

After analysing the project; the important project characteristics that have impact on agile practises for selected method are identified. These project characteristics provide support to the method engineer for deciding the 'variable constituents of configured method'. Following section present a case study.

4 Case study – Software Project for a Mobile company

To show the practical implementation of the proposed methodology, case studies are used as a research method. During the research, it was found that for mobile application domain agile development methodology is preferred over traditional methodology. The following case study shows – how an agile method is configured to form project-specific method for mobile application domain.

Case Study 1: A large software project developed for a mobile company to produce a usage analysis tool for analysing the customer's requirements in this domain and intelligently studies the areas for the development in this domain. It involves a huge and highly experienced team for its development which are further distributed into small teams. The project uses the complex technology for the implementation. The average duration of the project was 1 year. There is a need for documented requirements, to track the progress of the project and further to help during the testing phase.

The set of elicited organisational requirements is given in Table 4.

Table 4:- Set of elicited Organisation Requirements

	Characteristics	Values
R1	Task extent	Small
R2	Group size	Less than 10
R3	Development Style	Rapid Development
R4	Code Style	Clean and Simple
R5	Technology Environment	Quick Feedback
R6	Physical Environment	Distributed teams
R7	Business Culture	Collaborative and cooperation
R8	Abstraction Mechanism	Object-Oriented

The fuzzy logic controller will calculate the membership of agile methods, corresponding to the elicited organisational requirements. For the above set of elicited requirements *SCRUM has membership of 83% and the method XP has membership degree of 68%* and so on. After analysis the important project characteristics for deciding the key practises of selected agile method are shown in Table 5. In accordance, with these characteristics the weights are assigned to the method practices of Scrum shown in Table 6.

Table 5:- Identified project characteristics for case study 1.

Number	Requirements
R1	Large Software
R2	Complex Technology
R3	Experienced teams
R4	Distributed teams
R5	DocumentedRequirements
R6	Iterative Developments

Table 6:- Weighted practices of scrum for the case project 1.

Number	Practice	Weight
P1	Product Backlog	0.8
P2	Sprint Review	0.4
P3	Scrum teams	0.9
P4	Sprint	0.8
P5	Daily Scrum meeting	0.3
P6	Sprint planning meeting	0.6
P7	Sprint retrospective	0.0
P8	Scrum of Scrums	0.2

These weighted practices will provide a support system to the method engineer to select the *variables* in a method. The scrum process and configurable model of scrum (refer Table 1) are given below:-

The configurable model of scrum,
Common = {sprint, scrum planning meeting, scrum team}
Variable = {product backlog, sprint review, daily scrum meeting, sprint retrospective, scrum of scrums}

For the case project, the '*product backlog*' is found heavily weighted thus, among the set of variable it needs to be add in the configured method. The less weighted practices '*sprint review*' and '*daily scrum meeting*' and '*scrum of scrums*' can be tailored or modified for the purpose. However, '*sprint retrospective*' is removed from the configured method. Hence, the configured method formed for the current project is:

Configured Method Scrum for the case project: *All Scrum 'common concepts' with 'product backlog'* and modified '*sprint review*', '*daily scrum*' and '*scrum of scrums*'

5 Conclusions

In today's dynamic market environment producing high quality software rapidly and effectively is crucial. In order to allow fast and reliable development process, several agile methodologies have been designed and are now quite popular. Software developers find these methods as interesting and are concentrating more and more on these light-weight methods. Through their practical experience in the field it was found that agile processes may individually be incomplete to support the whole development process well, hence their processes require to be tailored to meet the requirements.

Therefore, a need arise to apply method engineering principles and practices to agile methods. As mentioned in the paper that these methods have a significant difference in their process thus, it is difficult to produce a generic model for them. They can only be adapted for the project-specific needs using configuration process.

The agile method configuration process finds the degree of veracity of these methods for the specified set of project characteristics and configures them to form project specific methods. The method configuration process is based on configurable model. This model illustrates the essential component of agile methods and is an attempt to show that "being agile" is a specific combination of practices only.

This revolutionary approach opens the paths to utilize the revolution brought by the concept of agility. The process supports to specify the requirements in laymen language and finds the suitable agile methods for the same with the practices that need to be followed. The aim is to deliver project specific agile method for the current organisation requirement.

6 References

[1] Miller, D. and Lee J. (2001). The people make the process: commitment to employees, decision making and performance. Journal of management (27), 163-189.

[2] Agile Manifesto (2001) Manifesto for Agile Software Development, [online] http://www.agilealliance.org/the-alliance/the-agile-manifesto/ (accessed 14 March 2005).

[3] Beck, K. (1999). Embracing change with extreme programming. IEEE Computer Society Press, Vol. 32, No. 10, pp.70–77.

[4] Beck, K. (1999). Extreme programming explained: Embrace change. Reading, Mass., Addison-Wesley.

[5] Cameron, J., (2002). Configurable development processes. Communications of the ACM, 45(3), 72–77.

[6] Coad, P., LeFebre, E. and DeLuca, J. (2000). Java Modeling in Color with UML: Enterprise Components and Process, Prentice Hall, Inc., Upper Saddle River, New Jersey.

[7] Cockburn, A. (2000). Writing effective use-cases. The crystal collection for software professionals. Addison-Wesley professionals.

[8] DSDM consoritium, (1997). Dynamic System Development Method, version 3. Ashford engineering, DSDM consortium.

[9] Dwivedi, R. and Gupta, D. (2015). The Agile Method Engineering: Applying fuzzy logic for evaluating and configuring agile methods in practice. In International Journal of Computer Aided and Engineering Technology. (In Press).

[10] Dwivedi, R. and Gupta, D. (2015). Applying machine learning for configuring agile methods. In International Journal of Software Engineering and its Application, 9(3), 29-40.

[11] Gupta, D. and Dwivedi, R. (2015). A frame work to support evaluation of project-in-hand and selection of software development method. In Journal of Applied and Theoretical Information Technology, 73(1), 137-148.

[12] Fitzgerald, B., Hartnett, G. and Conboy, K., (2006). Customizing agile methods to software practices at Intel Shannon, European Journal of Information Systems, 15(2), 197–210.

[13] Fuller A. and Croll P. (2004). Towards a generic model for agile process. In constructing the Infrastructure for the Knowledge Economy, Springer, (pp 179-185).

[14] Haungs, J. (2001). Pair programming on the C3 project. Computer 34(2): 118-119.

[15] Abrahamsson, P., Salo, O., Ronkainen, J. and Warsta, J. (2002). Agile Software Development Methods Review and Analysis. VIT Publications, Juhani Warsta, University of Oulu.

[16] Qumer, A. and Henderson-Sellers, B. (2008). A framework to support the evaluation, adoption and improvement of agile methods in practice. The Journal of Systems and Software, 81(11), 1899–1919.

[17] Qumer, A. and Henderson-Sellers, B. (2008). An evaluation of the degree of agility in six agile methods and its applicability for method engineering, Information and Software Technology, (50), 280–295.

[18] Rizwan, M. and Qureshi, J. (2012). Agile software development methodology for medium and large projects. IET Software, 6(4), 358–363.

[19] Scharff, C. and Verma, R. (2010). Scrum to Support Mobile Application Development Projects in a Just-in-Time Learning Context. In Proceedings of the ICSE Workshop on Cooperative and Human Aspects of Software Engineering. Cape Town, South Africa, (pp. 25–31).

[20] Schwaber, K. and Beedle, M. (2002). Agile Software Development with Scrum, Nouvelle editions.

[21] Vlaanderen, K., Jansen, S., Brinkkemper, S and Jaspers, E. (2011).The agile requirement refinery: applying SCRUM principles to Software product management. Information and Software Technology, 53(1), 58–70.

ISBN: 1-60132-446-4, CSREA Press ©

SESSION

TESTING, VERIFICATION, VALIDATION METHODS AND SECURITY RELATED ISSUES

Chair(s)

TBA

ISBN: 1-60132-446-4, CSREA Press ©

ISBN: 1-60132-446-4, CSREA Press ©

Security Evaluation Using Software Diversity Measurement: An Ecological Approach

Yong Wang
Dept. of Computer Science
Alcorn State University
Lorman, MS 39096, USA
email ywang@alcorn.edu

Qiang Duan
Information Science &Technology
The Pennsylvania State University
Abington, PA 19001, USA
email qduan@psu.edu

Dick Simmons
Dept. of Computer Science
Texas A&M University
College Station, Texas 77843, USA
email simmons@cse.tamu.edu

Abstract

Security evaluation for software ecosystems, which consist of various software systems interacting with each other through networks, is an important and challenging research problem. Previous study has shown the relationship between diversity and security of a software ecosystem; therefore, quantitative measurement for diversity level of software systems provides a useful tool for security evaluation. Inspired by the similarity between software ecosystems and ecological systems, in this paper we apply the Shannon-Wiener index, a typical method used in Biology for diversity measurement, in software systems to develop a quantitative method for measuring software system diversity. Using this method we evaluated the diversity levels of some important software systems in the Internet-based information infrastructure and discovered that most of the measured systems have a low diversity level, which implies potential weakness in these systems to resist security threats that may spread over the Internet. One exception we found is the operating systems for Internet servers, which currently have a fairly high diversity level. In addition, our analysis shows that the diversity level of Internet server OS has increased in the past two years, which is very encouraging for enhancing security of Internet-based computing systems.

Keywords

Software, diversity, security evaluation, ecological approach

Full/Regular Research Paper

1.INTRODUCTION

With rapid development of information and networking technologies, various software systems are interconnected through computer networks; thus forming a complex software ecosystem in which the performance of each individual software system is strongly influenced by various other systems. Compared to isolated software, such a networked software ecosystem faces more security challenges since security compromise happens at any individual system component may spread over a large part of the system quickly through networks. Examples for such challenges are software virus spread over the Internet that can infect a huge number of hosts in a short time periods and distributed deny of service attacks that may generate overwhelming traffic from various locations in the Internet to block a web server. What makes the situation even worse is that improvement in the software and networking systems, including faster CPU, larger memory, and high throughput for data transmission, also strengthens the impact that security compromises occur at individual software can make to the entire ecosystem. Also the emerging Cloud computing paradigm and network virtualization technologies enable a converged infrastructure for data processing and communications, which may significantly strengthen the influence among software systems interconnected through networks [5]. In order to face these new security challenges, we need new methods to obtain insights about the security of a networked software ecosystem.

Similar situations exist in ecological systems as in software systems. It has been long appreciated in Biology that monocultures are extremely vulnerable to pests and diseases. Management practices such as fertilizing and thinning help maintaining a high plant quality may also facilitate the rapid development of pest infestations, just like faster networks help software virus spread more quickly. On the other hand, researchers have found that multiple culture ecological systems can effectively prevent pests and diseases outbreak. Diversity is an important Source of robustness in biological systems. A stable ecological system contains many different species. If the diversity is lost, a few species become dominant, the ecological systems become susceptible to infestation, then pests and diseases outbreak.

Inspired by biology and ecological results, researchers in computer science recently started investigating the relationship between diversity of a software ecosystem and the security of the system. Geer pointed out in [6] that a monopoly environment is harmful for computer system security. Once a host in a monopoly computing environment is infected by worms and virus, the worms and virus can spread out rapidly. In [13], Geer *et al* explicitly studied the relationship of software monoculture and security threat and specifically presented the risks to cybersecurity post by dominance of Microsoft products. The authors used Window32/Blaster worm in 2003 as an example to show danger from monoculture and claimed that more diverse operating systems would have limited susceptible systems and thus reducing the worm infections. More works on enhancing information security by increasing software diversity have been

reported in the literature. Forrest et al. have contributed significantly to apply biological concepts to improving security in computer systems [7]. They stated that diversity can reduce the impact of security vulnerabilities and also proposed some general approaches to increasing diversity by avoiding unnecessary consistency. In [14] [18], the authors showed that diverse operating systems can statistically reduce software vulnerabilities and improve intrusion tolerance of the information system. Han and his coauthors studied effectiveness of software diversity for protecting system security and proposed an approach to choosing optimal combination of different operating systems.

The aforementioned works indicate that diversity of software in an information system, especially the diversity operating systems adopted by the hosts and servers in the system, has a significant impact on system security. Therefore, it is desirable to have a simple method to measure software (operating system) diversity for a software ecosystem. Such a measurement provides an indicator that can be used by the system designers and administrators to evaluate the current level of diversity; thus obtaining an insight about the tolerance that the system may have to security vulnerabilities and compromise.

In ecological systems, it is common to use Shannon-Wiener index as a measurement of community diversity and thus the stability of the ecosystem. The similarity between an ecological system and a software ecosystem inspires us to explore the application of Shannon-Wiener index to measure software diversity in this paper in order to obtain an indicator for system diversity and obtain some insights about system tolerance to security vulnerabilities.

Specifically we make the following contributions in this paper. We explore applicability of the Shannon-Wiener index method, which was originally for measuring diversity of ecological systems, to evaluate diversity of software systems in this paper. Then we apply this method to measure diversity of typical operating systems, especially for networking software ecosystems and uncover insights about how robust the studied systems are to resist security threats that are spread across the Internet.

In the rest of this paper, we will first give an introduction to the Shannon-Wiener method for measuring diversity in Section 2. Then in Section 3 we apply the Shannon-Wiener index to evaluate the diversity of operating systems for various software ecosystems and discuss the insights we obtain. We give conclusion remark and discuss possible future work directions in Section 4.

2. DIVERSITY MEASUREMENTS

In ecology, the most popular measurement of species diversity is based on information theory. The objective of information theory is to measure the amount of order (or disorder) contained in the system. Four types of diversity related data may be collected in a community: 1) the number of species, 2) the number of individuals in each species, 3) the places occupied by individuals of each species, and 4) the places occupied by individuals as separate individuals. In most community, only data in type 1 and 2 are obtained.

$$H' = -\sum_{i=1}^{s}(p_i)(\ln p_i) \qquad (1)$$

where H' = information content of sample (bits/individual)

= index of species diversity

S = Number of species

Pi = Proportion of total samples belonging to the *i*-th species.

Information content is a measure of the degree of uncertainty; therefore the greater is the value H', the stronger the uncertainty is.

The Shannon-Wiener index H' increases with the number of species in the community. In theory, it can reach a very large value. In practice, for biological communities, H' does not exceed 5.0 (Washington 1984). The theoretical maximum value is log(S), and minimum value (when N>>S) is log [N/(n-S)] (Fager 1972).

The true diversity is:

$$D(s) = \exp^{H(s)} = \exp^{-\sum_{i=1}^{s}(pi)(\ln pi)} \qquad (2)$$

$$D(S) = \exp^{H(s)} = \exp^{-\sum_{i=1}^{s}(pi)(\ln pi)} = \sum_{i=1}^{4}(1/pi^{pi}) \qquad (3)$$

The D value is equivalent species number.

The D value is equivalent species number.

Let us look at an example:

Table 1. Species composition in community

Species	1	2	3	4
Community A	025	0.75	0	0
Community B	0.1	0.2	0.3	0.4
Community C	0.25	0.25	0.25	0.25

Following the equation 2, we get then community A

$$H(A) = -\sum_{i=1}^{4} p_1 \ln(p_1) = -(1/4)\ln(1/4) - (3/4)\ln(3/4)$$
$$= 0.34657+0.21576 = 0.56233,$$

$D(A) = \exp^{0.5623} = 1.7548$

For data in Community C ,

$H(C) = -\sum_{i=1}^{4} (1/4) \ln (1/4) = -\ln (1/4) = \ln(4)$,

$D(C) = \exp^{\ln(4)} = 4$

We have the value of 4 esn (effective number of species), which is the maximum number with 4 species.

The community C is much less diverse than community A.

3. DIVERSITY MEASUREMENT FOR DIFFERENT OPERATING SYSTEMS

In this section we apply Shannon-Wiener index to measure diversity of operating systems in a few types of typical information systems.

3.1 Desktop and Laptop Operating Systems

Table 2 gives the percentages of typical operating systems used for desktop and laptop computers. Based on data in Table 2 we can calculate the diversity index as

Table 2. Market shares of different operating systems for desktop and laptop computers (July, 2014) [10]

Operating systems	Market share
Linux	1.68%
Windows Vista	3.05%
Mac OS X 10.9	4.12%
Windows 8	5.92%
Windows 8.1	6.56%
Windows XP	24.82%
Window 7	51.22%
Other	2.65%

$H = -\sum_{i=1}^{8} p_1 \ln(p_1) = -0.0168*\ln(0.0168)-$

$0.0305*\ln(0.0305)-0.0412*\ln(0.0412)-$

$0.0592*\ln(0.0592)-0.0656*\ln(0.0656)$

$-0.2482*\ln(0.2482)-0.5122*\ln(0.5122)-$

$0.0265*\ln(0.0265)$

$= 0.0687+0.1064+ 0.1314+ 0.1673 + 0.1787+0.3459+0.3427+0.0962 = 1.4373$

$D = 4.2093$

The diversity index is 4.2093, which is much lower than the optimal value of 8.0. We can see that in this software ecosystem, Windows are the dominating operating systems, which causes a low diversity level thus presenting a higher risk for security.

3.2 Web Client Operating Systems

Table 3 gives the market share information about different types of operating systems for Web clients. The table shows that there are a variety of operating systems in this area, but currently Windows 7 is holding the largest market share, followed by Windows XP. Based on this table we can calculate the diversity index as

$H = -\sum_{i=1}^{6} p_1 \ln(p_1) = -0.5928*\ln(0.5928)-0.1853*\ln(0.1853)-0.1727*\ln(0.1727)-0.0286*\ln(0.0286)-0.015\ln(0.015)-0.00479*\ln(0.00479) = 1.116$

$D = e^{1.116} = 3.0526$

In the web client market, Microsoft Windows is the major operating system, followed by Linux and Apple operating systems. The effective diversity index value is 3.0526, which is much lower than the optimal value of 6.0 (basically just half of the idea level). The obtained result implies that the Microsoft dominating market share in this area may potentially cause a software ecosystem that is fairly vulnerable to Internet-based security threats due to the low diversity. Therefore, from a security perspective, an organization might want to consider adopting more diverse operating system in an enterprise network; although this may implies more cost in system management.

Table 3. Web client operating systems [10]

Operating systems	Percents
Microsoft Windows 7	59.28%
Linux Kernel based	18.53%
Apple	17.27%
Symbian, S40	2.86%
Other	1.5%
Blackberry	0.479

3.3 The Operating Systems for Global Tablet Computers.

Tablet computers become a significant OS market share category starting with Apple's iOS based iPad. There have been 170 millions iPad sold as of October 2013 with 132 million iPads in 2012 and 2013 combined. There are 174 million Android and 5 million Microsoft based tablet devices at same time period. Table 4 gives data about market shares of different tablet computers, based on which we can calculate the diversity index as

$H = - \sum_{i=1}^{6} p_1 \ln(p_1) = -(0.7181)*\ln(0.7181)-0.2518*\ln(0.2518)-0.0253*\ln(0.0253)-0.0023*\ln(0.0023)-0.0014*\ln(0.0014)- 0.001*\ln(0.001) = 0.791$

$D = e^{0.791} = 2.2056$

The effective index value is 2.2056, which is much less than the optimal value 6.0. This is a very low diversity index value, which is almost just one-third of the idea level. In this ecosystem, Apple iOS is the dominating one with 71.81% market shares. Such a low diversity level caused by the Apple dominating market share may cause potential threat to security of the a ecosystem comprising networked tablet devices, similar to the issue caused by Microsoft dominating role in Web client operating systems.

Table 4. Global tablet computer usage (July 2014) [11]

Operating systems	Percents
Apple iOS	71.81%
Android	25.18%
Linux	2.53%
Blackberry	0.23%
Win RT (Microsoft)	0.14%
Other	0.1%

3.4. Supercomputer Operating Systems

Table 5 shows the percentages of different operating systems used by supercomputers in Top 500 project as of July 2014. Based on the data, diversity index can be obtained as

$H(C) = - \sum_{i=1}^{5} p_1 \ln(p_1)$

$= -0.964*\ln(0.964)-0.022*\ln(0.022)-0.008*\ln(0.008)-0.004*\ln(0.004) -0.002*\ln(0.002) = 0.19247$

$D = e^{0.19247} = 1.2122.$

Table 5. Operating systems for supercomputers in Top 500 projects (July 2014) [10]

Operating systems	Percentage
Linux	96.4%
Unix	2.2%
Mixed unix and Linus	0.8%
Microsoft Windows	0.4%
BSD based	0.2%

We can see that Linux is the operating system used on most of the supercomputers, probably because they are realized as clusters of Linux servers. Such a dominating operating system share leads to a very low diversity index, approximately 1.2 out of 5.0, which indicates that the supercomputers consisting of almost homogeneous servers might have fair weak resistance to some security threats such as virus and worms spreading through networks.

3.4 Mobile Device Operating Systems

Operating systems for mobile devices, such as smart phones, now form a significant part of the networked software ecosystem. Table 6 gives percentage shares of various typical mobile device operating systems. According to the data shown in this table, the diversity index is

$H = - \sum_{i=1}^{8} p_1 \ln(p_1)$

$=-0.4462*\ln(0.4462)-0.4419*\ln(0.4419)-0.0419*\ln(0.0419)+0.0257*\ln(0.0257)-0.0249*\ln(0.0249) -0.0121\ln(0.0121)-0.0064*\ln(0.0064)- 0.0009*\ln(0.0009)$

$= 1.13195$

$D = e^{1.13195} = 3.1017$

Table 6. Mobile operating systems

Operating systems	Percentage
Android	44.62%
iOS	44.19%
Java ME	4.19%
Symbian	2.57%
Windows Phone	2.49%
Blackberry	1.21%
Kindle	0.64%
Other	0.09%

The diversity value is 3.1 out of 8.0, which is still fairly low. This implies that although there are various types of mobile devices, Android and iOS are apparently the most popular OS used in most of them (more than 88% market share); thus still causing low diversity.

3.5 Internet Server Operating Systems

Internet servers, including Web servers, mail servers, and DNS servers, play a crucial role in Internet service provisioning. Internet servers are the main target of almost all kinds of security attacks, which are often lunched remotely through the network. Therefore diversity of Internet servers, which can reflect the tolerance of Internet servers to the Internet security attacks, must be carefully examined. The percentage shares of different operating systems for Internet

servers are listed in Table 7, based on which we can computing the diversity index as

$H = -\sum_{i=1}^{4} p_1 \ln(p_1) = -0.386*\ln(0.386)-0.01*\ln(0.01)-0.2777*\ln(0.2777)-0.326*\ln(0.326)$

$= 0.3674+0.0461 + 0.3558+0.3654 = 1.1347$

$D = = e^{1.1347} = 3.1102$

Table 7 Internet server operating systems (February, 2014)

Operating systems	Percentages
Linux	38.6%
BSD	1.0%
Other Unix	27.77%
Microsoft Windows	32.6%

Based on the data given in Table 7 and the obtain diversity index, we can see that Internet server operating systems have a fairly high diversity level. The effective number of species (D values) is more than 3.1 out of 4. This indicates that Internet operating systems, compared to operating systems for mobile phone, supercomputers, tablet devices, and regular desktop/laptop computers, have a higher diversity level; thus are more robust to resist security attacks spread through networks. This is an encouraging result because Internet servers play the most critical role in Internet service provisioning; therefore their security level has the most significant impact on the entire Internet-based software ecosystems. A relatively high level of diversity (3.1 out of the ideal 4.0) indicates that the current Internet server system potentially has fairly strong resistance to security threats that may be spread through networks.

4. SOFTWARE DIVERSITY IN OPERATING SYSTEMS EVOLUTION CROSS TIME

With the rapid development in mobile communications and Cloud computing technologies, using mobile devices for accessing Internet servers is becoming the most popular approaches for consumers to access and utilize Internet services. Therefore, security of the operating systems for mobile devices and Internet servers is particularly important and evaluating diversity of these operating systems is interesting to us. Both mobile computing devices and Internet servers form very dynamic areas where the market shares of different operating systems have been changing in the past few years. Therefore, we want to investigate the diversity of mobile OS and Internet servers evolve over time recently in order to obtain an insight of the trend of diversity of these two very important systems.

Table 8 gives the market shares of Internet server operating systems in November 2012 and February 2014. We calculated the diversity index value based on 2012 data and obtained that

$H(C) = -\sum_{i=1}^{5} p_1 \ln(p_1)$

$= -0.66*\ln(0.66)-0.06*\ln(0.06)-0.28*\ln(0.28)$

$= 0.27424 + 0.1688+ 0.3564 = 0.7987$

$D= e^{0.7987} = 2.2227$

Table 8. Internet operating systems market shares

Operating systems	Percent (November, 2012)	Percent (February, 2014)
Linux	66%	38.6%
BSD	6%	1.0%
Other Unix versions	0	27.77%
Microsoft windows	28%	32.6%

Comparing the above obtained D index value with the diversity index value obtained in Subsection 3.5, we are glad to see that from November 2012 to February 2014 the diversity index of Internet server operating systems increased from 2.2 to 3.1, which indicates that this area is becoming more diverse. We also calculated normalized diversity index values, which is the ratio of D index over the optimal value, for November 2012 and February 2014 data, and the results are plotted in Figure 1.

The above analysis uncovered that the diversity of Internet server operating systems, which is one of the most critical components in the entire Internet-based software ecosystem, has risen from 2.2/3.0 to 3.1/4.0 in the past 14 months. Such increment in diversity level of Internet server operating system brings in positive influence on enhancing Internet security.

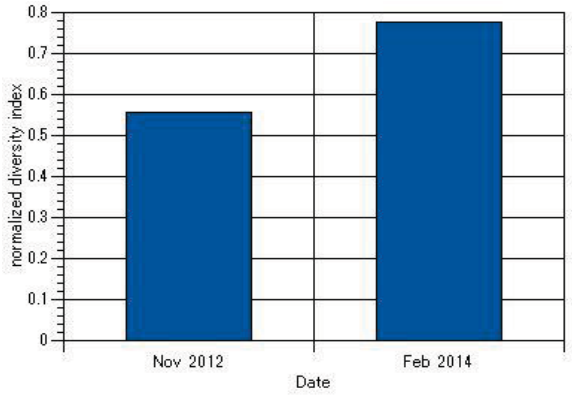


Figure 1 Normalized diversity index change over time for Internet server operating systems

Table 9. Mobile device OS market share in USA [15]

Year and quarter	IOS	Android	RIM*	WP**
2014 Q1	35.9%	57.6%	0.7%	5.3%
2013 Q1	43.7%	49.3%	0.9%	5.6%
2012 Q1	44.6%	47.9%	2.6%	3.7%

*RIM: Windows phone

**Blackberry operating systems.

The market shares of different mobile operating systems for the past three years are given in Table 9. Therefore, we can calculate the diversity index for the past three years as follows.

For the first quarter of 2012,

$H(C) = - \sum^{5}_{i=1} p_1 \ln(p_1)$

$=-0.446*\ln(0.446)-0.479*\ln(0.479)-0.026*\ln(0.026)-0.037*\ln(0.037) =0.9296$

$D= e^{0.9296} = 2.5335$

For the first quarter of 2013,

$H(C) = - \sum^{5}_{i=1} p_1 \ln(p_1)$

$=-0.437*\ln(0.437)-0.493*\ln(0.493)-0.009*\ln(0.009)-0.056*\ln(0.056) =0.9141$

$D= e^{0.9141} = 2.4945$

For the first quarter of 2014,

$H(C) = - \sum^{5}_{i=1} p_1 \ln(p_1)$

$=-0.359*\ln(0.359)-0.570*\ln(0.570)-0.007*\ln(0.007)-0.053*\ln(0.053) = 0.8759$

$D= e^{0.8759} = 2.4010$

The above obtained data show that although market shares of different mobile device operating systems have changed, the overall diversity index value for this area has slighted dropped from 2.53 to 2.40 out of an optimal value 4.0. This implies that the diversity of this ecosystem have been staying at a relatively low level and even become a little less diverse in the past two years. In order to show the trend more clearly, we calculated normalized diversity index values and the obtained results are plotted in Figure 2.

The obtained analysis result regarding lack of diversity of mobile device operating system and the trend that this area is even becoming less diverse in the past two years bring in a serious concern about the vulnerability of mobile devices to security threats, such as malwares and virus, that are spreading through the Internet, due to the low diversity of such software ecosystems. Since mobile computing is becoming the main model for regular customers for access Internet to consume computing services, potential security weakness caused by lack of diversity deserves close attention. In addition, mobile devices are typically operated by users who may not have sufficient knowledge and skills for configuring their devices with highly secure setting; therefore it is relatively easier for hacker to penetrate security protection on mobile devices, which makes the security threats more serious.

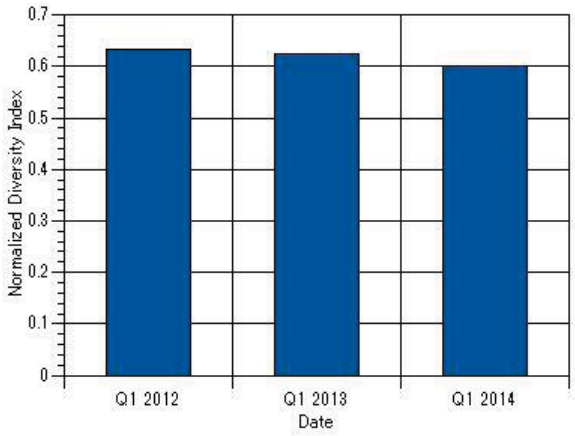


Figure 2 Normalized diversity index change over time for mobile device operating systems

5. CONCLUDING REMARKS

Due to the direct relationship between the diversity of a networked software ecosystem and its tolerance resistance to Internet security threats, measuring diversity level of software system becomes an important research topic for evaluating system security. Inspired by the similarity between networked software ecosystem and ecological environment, in this paper we explored the application of the Shannon-Wiener diversity index to give a quantitative measurement for diversity of software systems.

We analyzed operating systems of various software systems, including desktop/laptop computers, tablet computers, supercomputers, mobile devices, and Internet servers and calculated the diversity index values for these systems. The obtained results indicate that the operating systems used in most of the current software systems, including desktop/laptop computers, tablet computers, supercomputer, and mobile devices, all have fairly low diversity level. This implies that these software systems have relatively weak resistance and low tolerance to Internet security attacks. On the other hand, our analysis showed that the diversity index value for the operating systems of Internet servers are fairly high (3.1 out of 4), indicating a quite diverse ecosystem in

this field. We also studied how the diversity levels of OS for mobile devices and Internet servers evolved over time in the past three years. We found that diversity level of mobile operating systems has slightly dropped, but the Internet server operating system field has become more diverse since 2012.

The increasing diversity level of Internet server OS, which is one of the key software components in the entire Internet software ecosystem, shows a positive sign in terms of Internet security. More diverse Internet server software has stronger resistance that prevents security compromise from quickly spreading over the Internet; thus may mitigate some security vulnerabilities. On the other hand, the relatively low diversity index values that we found in our paper for other software systems urgently call for actions to enhance diversity level of these systems. It is true that managing more diverse software systems could imply extra costs in maintenance and support. There are some options for increasing the diversity of the off-the-shelf components of software systems without introducing too much extra cost. Software components reuse may provide less exposure to vulnerabilities and is beneficial for software system security. Other possible techniques for increasing software system diversity include space layout randomization and N-variants. We believe that enhancing software system diversity and evaluating its effect on improving system security is an interesting and challenging research topic that deserves thorough study.

REFERENCES

[1] Elena Gbriela Barrantes, David H. Ackley, Trek S. Palmer, Darko Stefanovic, and Dino Dai Zovi. "Randomized instruction set emulation to disrupt binary code injection attacks," In Proceeding of 10th ACM conference on Computer and communication security, pages 281-289. ACM, 2003.

[2] Sandeep Bhatkar, Daniel C DuVarney, and Ron Sekar. "Address obfuscation: An efficient approach to combat a broad range of memory error exploits" In Proceedings of th 12th USENIX security symposium, volume 120, Washington, DC. 2003.

[3] "Desktop Operating Systems market Share," http://www.netmarketshare.com/operating-system-market-share.aspx?qprid=10&qpcustomd=0

[4] Qiang Duan, Yuhong Yan, and Athonasios Vasilakos, "A Survey on Service-Oriented Network Virtualization toward Convergence of Networking and Cloud Computing," IEEE Transactions on Network and Service Management, 9(4): 373-392 (2012)

[5] E. W. Fager 1972. "Diversity: a sampling study". American Naturalist 106:293-310

[6] Daniel Geer. Monopoly considered harmful. Security and Privacy. IEEE 1(6):14-17, 2003.

[7] Stephanie Forrest, Anil Somayaji, and David H. Ackley. "Building diverse computer systems," In operating systems, 1997, the sixth Workshop on Hot Topics, Page 67-72, IEEE 1997.

[8] Charles J. Krebs, Ecological Methodology, 2nd edition, Addison, Wesley, Longman, 1999.

[9] E. C. Pielou 1966. "The measurement of diversity in different types of biological collections," Journal of Theoretical Biology 13:131-144.

[10] H. G. Washington 1984. Diversity, biotixc and simularity indices: a review with special relevance to acquatic ecosystems. Water research 18:653-694.

[11] Wikipedia, July, 2014 "Usage share of operating systems,"http://en.wikipedia.org/wiki/Usage_share_of_operating_systems

[12] Jun Xu, Zbigniew Kalbarczyk, and Ravishankar K Iyer, "Transparent runtime randomization for security, " In Reliable Distributed Systems, 2003, Proceedings, 22nd International Symposium on, page 260-269, IEEE, 2003.

[13] Daniel Geer, Rebecca Bace, Peter Gutmann, Perry Metzeger, Charles P Pfleeger, John S. Quarterman and Bruce Schneier "Cybersecurity: The cost of monopoly," Computer and Communications Industry Association (CCIA), 2003.

[14] Miguel Garcia, Alysson Bessani, Ilir Gashi, Nuno Neves, and Rafael Obelhearo. "Analysis of operating systems diversity for intrusion tolerance," Software: Practice and Experience, 2013.

[15] Jin Han, DebinGao, and Robert H Deng. "On the effectiveness of software diversity: A systematic study on real-world vulnerabilities," In Detection of Intrusions and Malware, and Vulnerability Assessment, pages 127-146. Springer, 2009.

[16] Mobile Operating Systems. http://en.wikipedia.org/wiki/Mobile_operating_system.

[17] Jane Jorgensen, Philippe Rossignol, Masami Takikawa, Daniel Upper. "Cyber ecology: Looking to ecology for insightas into information assurance." http://www.pitt.edu/~dtipper/3957/Biology2.pdf, IEEE, 2001

[18] Julio Hernandez-Castro, and Jeremy Rossman, Measuring software diversity, with Applications to Security, 2013.

[19] Christian Korner, "Functional plant ecology of high mountain ecosystem," http://www.springer.com, 2008

A Survey of Testing Context-aware Software: Challenges and Resolution

Songhui Yue [1], Songqing Yue[2], and Randy Smith [1]
[1]Department of Computer Science, University of Alabama, Tuscaloosa, AL, USA
[2]Department of Mathematics and Computer Science, University of Central Missouri, Warrensburg, MO, USA

Abstract
Testing is an essential method to ensure the quality of software. Research of testing context-aware software is gaining in importance with the rapid development of context-aware software and the increasing needs to ensure their quality. Context-aware abilities bring new challenges to testing context-aware software. This paper investigates this from the perspective of four categories of challenges: context data, adequacy criteria, adaptation and testing execution. We also describe approaches current researchers are using to solve these challenges. Our contributions in this paper include the analysis of the relationships between the identified challenges and an ontology diagram that depicts these challenges and relationships, which may benefit the exploration of future research in related areas.

Keywords: Context-aware, Testing, Quality, Challenges, Resolution

1 Introduction

Nowadays, our electronic devices become more powerful in both computing and obtaining information from the environment. Many new devices employ a multi-core processor, and with the technological advances in networked computing environments, new computing paradigms such as cloud computing have been proposed and adopted [8]. Consumers with mobile devices can access data from a "Cloud" at any time in a fast speed wherever network connection is available. Particularly, a modern smart phone can be equipped with as many as fourteen sensors [9], such as proximity sensor, ambient light sensor, accelerometer, magnetometer, and gyroscopic sensor. As a result, a large variety of information could be used as context to enrich the functionality of software applications. The extra abilities of modern devices could be used by applications to process more information for benefits of users, and this advantage makes context-aware become more and more popular in ubiquitous computing area.

A variety of context-aware applications have already been developed, such as location-aware systems, hospital information-aware systems, office-aware applications, and home-aware applications [4][5][6]. These applications are deployed on different platforms, such as mobile applications, web-based applications [10] and embedded applications. Plenty of concepts and components were introduced for facilitating the development of context-aware software, such as context, context-aware middleware, and adaptation rule. They provide software with context-aware abilities and meantime bring new challenges to testing, thus should be considered thoroughly. We will discuss these concepts in detail in section 2.

The following sections are organized as follows: Section 2 introduces some key concepts as the background for understanding our study. Section 3 describes the four categories of challenges we identify from our survey and various approaches to solving them. Section 4 analyzes the relationship between the areas inspired by the challenges and Section 5 serves as the conclusion.

2 Background

This section provides detailed explanation of important concepts that serve as the basis for understanding testing context-aware software.

2.1 Context

The context definitions given by researchers are slightly different from each other because of their different understanding or application of the term. Schilit and Theimer [14] first introduced "context-aware" in their work and defined context as location, identities of nearby people and objects and changes to those objects (1994). Brown [15] defined context as a combination of elements of the user's environment that the computer knows about (1996). Dey et al. [16] defined context as the user information and user's changing location, the changing objects in the environment, and the familiarity with the environment (1998).

Based on all the prior attempts to define context, Dey & Abowd (2000) [17] provided a comprehensive definition of context which is used by most of the current related studies as "*any information that can be used to characterize the situation of entities (i.e. whether a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves. Context is typically the location, identity and state of people, groups and computational and physical objects.*"

In a context-aware application, context data can be retrieved with the assistance of hardware or software. For location based context-aware software, context information

contains discrete data to mark the locations, which are usually derived from the hardware level [19]. Sensors are widely utilized to capture changing contextual data and then pass them to the software [18]. Context data may also be generated from the software level. For instance, contextual information can be collected from other applications running in the same or related devices [18].

2.2 Context-aware Middleware

Context-aware middleware is widely used for facilitating development and execution of context-aware software [13]. Middleware refers to software systems, which provide an abstraction and mechanisms between network operating system layer and applications layer [20] [21]. Researchers have developed various middleware systems for building and rapidly prototyping context-aware services [22] [23]. As the work in [24] suggests, typical middleware architecture for developing context-aware software contains two key components: context manager and adaptation manager. Context manager captures and manages context from surroundings, and pushes the context changes to adaptation manager. Adaptation manager is responsible for reasoning on the impact of context changes and then choosing proper reactions for applications behaviors.

2.3 Context-aware Adaptation

Context-aware adaptation refers to the ability of computing systems to adapt their behaviors or structures to highly dynamic environments without explicit intervention from users, with the ultimate aim of improving the user experience of these computing systems [35]. Context can be used by software through triggering the context adaptation rules. Adaptation rules, which are usually maintained, evaluated and applied by adaptation manager of a context-aware system, define a significant portion of an application's behavior [13]. We can use an example of a car system to illustrate how an adaptation rule works. Suppose a car installed with an autonomous-driving system (ADS) needs to change lanes. The adaptation rules in ADS need to assure that the car can take this action only if the current context is safe for changing lanes. There should be some additional rules to define what is safe in a real driving environment, which ADS can use to check the safety. If ADS knows the context is safe, it will choose a way to react according to some other rules: changing to left lane or changing to right lane, and in what speed.

2.4 Boundary testing

Boundary testing is an important traditional testing technique, which can also be applied to testing context-aware software. With boundary value testing, test cases are designed to take extremes of input domain. The extremes include values of maximum, minimum, inside/outside boundaries, typical values, error values, and etc. New challenges emerge when boundary testing is used in testing context-aware software, which may require extra attention.

3 Challenges in Testing Context-aware Software

Context-aware capacity imposes many new challenges in developing and testing applications that support context-awareness. After investigating the state of the art in this area, we have identified four main categories of challenges in testing context-aware software: context source, adequacy criteria, adaptation and testing execution. In this section, we provide detailed description for challenges in each category.

3.1 Context

Wang et al. [7] argue that the added capabilities of context-awareness introduce a distinct input space. Since context changes can affect software behavior at any point during the execution, context as testing data should be well studied and selected. However, context data retrieved from sensors usually have such characteristics as being *inaccurate, inconsistent*, and *continuous* which may increase the difficulty in selecting testing data. In this subsection, we mainly discuss the features of inaccuracy and inconsistency in context data and briefly introduce how continuous context may affect boundary testing.

3.1.1 Context Inaccuracy

Sensor data can be inaccurate [25]. Such data should be well studied before using for testing. Traditional testing methods usually use accurate values as test cases. However, for testing context-aware applications, especially those obtaining data directly from sensors, it is reasonable for testing engineers to question the reliability of the data.

Vaninha et al. [25] illustrate the relationships between the context sources (sensors or software) and defect patterns. They show that context sources are closely related to faults of several types: incompleteness, inconsistency, sensor noise, slow sensing, granularity mismatch, problematic rule logic, and overlapping sensors. Each fault type is caused by one ore more failures in context sources, such a Camera, GPS, or WiFi. Table 1 (borrowed from [25]) shows the relationship between context sources and fault types, e.g., ambiguity, as one form of incomplete, may be caused by errors in the context source of RFID/NFC, QR-CODE or Clock/Alarm.

The problem of inaccuracy in context data can cause a high-level defect called context inconsistency, which may relate to multiple context sources or is a defect in interpretation from context [25].

3.1.2 Context Inconsistency

Context inconsistency occurs when there is at least one contradiction in a computation task's context [27]. It can be caused by sensor errors or sensor data inaccuracy [11] [12] [25] [26]. Asynchronous updating of context information can also cause the same problem [13]. As a result of the possible

Table 1. Context-Sources in Combination with Defect Patterns [25]

Context-Source	Incomplete			Sensor Noise				Slow Sensing		Overlapping Sensors	
	Unavailability	Not Interpretable	Ambiguity	Incorrectness	False Reading	Instability	Unreliability	Out-of-Dateness	Wrong Interpretation	Concurrent Values	Unpredictable
Accelerometer	X			X	X			X	X		
Wi-Fi	X						X	X	X		X
Camera	X	X						X	X		
RFID/NFC	X	X	X	X	X						
QR-Code	X	X	X	X	X						
GPS	X			X	X		X	X	X		
Light Sensor	X			X	X			X	X		
Clock/Alarm	X		X					X	X		
Calendar	X			X	X			X	X		
Gyroscope	X							X	X		

inconsistency of context, the application logic that rely on the context can lead to wrong behaviors or execution errors.

We can illustrate context contradiction using the WiFi access point (WAP) application where WAP can be used to detect the location of a device connected to it [11]. Suppose in a location identification service, WAP installed in each room of a building is supposed to detect the location of a person who is wearing a smart device. The smart device has a unique identification for each person. Context inconsistency may happen in the following situation: if WAP S1 installed in room R1 detects person P and claims that P is in R1 now and meanwhile WAP S2 embedded in room R2 detects the same person P and claims P is in R2. This type of inconsistency can happen in the following scenarios: rooms R1 and R2 are near each other or they are in the same coordinates of nearby floor.

Context-aware applications can get raw data from a single sensor or several sensors, and they can also get synthesized context data from middleware [29], which collects data from sensors as well. Raw data from a single sensor have great opportunities to exhibit inconsistency problems, however, data from a middleware, which does not apply consistency checking, may also experience inconsistency problems.

Chang et al. [27] try to solve the inconsistency problem using a framework for realizing dynamic context consistency management. Based on a semantic matching and inconsistency-triggering model, the framework can detect inconsistency problems. The framework also applies inconsistency resolution with proactive actions to context sources.

3.1.3 Continuous Context

Continuous context is used in many context-aware applications [29]. Challenges may arise when applying boundary testing in a continuous context. A straightforward way of modeling continuous context is to directly convert it into discrete one by dividing it into different time windows [29] [32]. Hidasi et al. [32] demonstrate that much information will be lost if such modeling approach is used. This missing information can be the boundary values, which will greatly affect the effectiveness of testing with the technique of boundary value analysis. To build better models for continuous context, Hidasis et al. propose fuzzy modeling approaches. The fuzzy modeling method advocates that context-state is not only associated with the interval it belongs to, but is also influenced by its relative location in the interval and neighboring intervals. Thus, a better understanding of the event or context-state with respect to a specific interval can be achieved, in which way the information loss of boundary values can be complemented.

For context data collection, Chen et al. [31] define *snapshot* as the union of all sensing values at a particular timestamp. The act of collecting multiple continuous snapshots is called continuous data collection (CDC) [30]. Their work focuses on challenges of network capacity, while Nath's [29] work concentrates on reducing sensing cost using a middleware approach when continuous context sensing is required.

3.2 Adequacy Criteria

Testing adequacy criterion is usually defined as a rule or a collection of rules a test set should satisfy [36]. To measure how well a program is examined by a test suite, usually one or more criteria are used. A variety of testing adequacy criteria have been developed for traditional testing while only a few are suitable for testing context-aware software. According to the work of Lu et al. [28], there are three kinds of obstacles that hinder the effective application of standard data flow testing criteria to testing Context-aware Middleware-Centric (CM-Centric) software, namely,

1) Context-aware faults: faults in the triggering logics in the middleware;

2) *Environmental interplay*: environmental updates may happen anytime, and test set should be updated in time accordingly;

3) *Context-aware control flow*: it is difficult to enumerate every control flow trace of context changing for some situations.

Recent research is using special approaches to generate testing criteria for context-aware software [26] [28]. For instance, Lu et al. [28] have applied a data flow method to generate adequacy criteria for testing middleware-centric context-aware programs. Different from traditional variables, a context variable can be defined and updated via either an assignment or an environmental update. Therefore, a new definition of "*definition (DEF) of variables*" and "*usages (USES) of variables*" are given, as well as "*update-use occurrences of variables*", which refers to an occurrences of a context definition due to sensing of environmental contexts and a context use. Imitating the conventional def-use (DU) associations, the paper provides definitions of def-use associations for CM-Centric programs, as well as a definition for the pairwise DU associations. Using the defined data flow associations, they generate novel test adequacy criteria to measure the quality of a test set for a CM-centric program.

3.3 Adaptation

Adaptation is the core process of using context for computing in context-aware software. In this subsection, we introduce testing challenges of context-aware software in *adaptation* activities. We explain the challenges in two perspectives: Erroneous adaptation rules and continuous adaptation.

Adaptation rules can be erroneous. Realizing that adaptation rules play an important portion in middleware based context-aware applications, the work of Sama et al. [13] is focused on fault detection in adaptation rules. In their approach, detection is driven by the requirement that the rules and its finite state machine satisfy the following properties: *Determinism*, *State Liveness, Rule Liveness, Stability, Reachability*. For example, determinism requires that for each state of the finite state machine and each possible assignment of values to the context variables in that state, the assignment of the value can only trigger at most one rule.

Continuous adaptation makes it hard to identify which adaptation rule have caused the faults, so it is difficult to set up an effective test oracle [33]. Xu et al. [33] suggest that for context-aware applications, the adaptation to the environmental changes may contain defects when the complexity of modeling all environmental changes is beyond a developer's ability. Such defects can cause failures to the adaptation and result in application crash or freezing. More importantly, they argue that tracking an obvious failure of the system back to the root cause in adaptation is generally difficult [33]. The reasons are as follows. Firstly, a failure is usually a consequence of multiply adaptations, and it is difficult to set up an effective test oracle. Secondly, when a failure happens, it is hard to collect all the context data because some of the data are from outside sensors. Thirdly, it is hard to repeat an observed failure. In their work, they propose a novel approach, called ADAM (adaptation modeling), to assist identifying defects in the context-aware adaptation.

3.4 Testing Execution

Testing execution refers to the process of executing a test plan, in which all the challenges mentioned in above categories should be considered. It not only needs to consider making test plans to resolve aforementioned challenges, but also to realize them by creating novel tools or mechanisms. In this subsection, we discuss the challenges of generating context for testing and introduce an open topic that new mechanisms are necessary for facilitating testing execution.

3.4.1 Context Testing Data Generation

Context can be complex and plenty of work has concentrated on context testing data generation. Two approaches can be used to provide context test information: real world testing and simulator testing. Real world testing means to evaluate an application in real devices with multiple sensors and network conditions. Repeated real-world testing can be expensive in time and effort, sometimes even infeasible when context and environment are complex, e.g. aerospace. However, real-world testing is still highly recommended before the acceptance or commercialization of an application.

Simulator testing can be an alternative when real-world testing is expensive or unpractical, and it is a frequently used approach [2] [3] [18] [34]. Designers need a set of models and tools that aim to achieve the objective of "design for reality". In real world, as we have discussed, context derived from sensors can be inaccurate, inconsistent, and continuous. Besides, sensor reading and network connections may strongly depend on the providers of sensors and networks. Thus it is very challengeable to build a well-equipped simulator. Eleanor et al. [18] propose a testing platform for the user-centered design and evaluation of context-aware services by using a 3D virtual reality simulation to show the environment to users and generate the simulated environment's context. They recognize that to simulate the sensors is very difficult.

3.4.2 Adoption of New Mechanism

Some new mechanisms have been adopted to facilitate context-aware software testing. Griebe et al. [1] use model transformation approach on context-enriched design-time system models to generate platform specific and technology specific test cases. For fulfilling testing criteria, Wang et al. [7] use a component of *Context Interleave Generator* to form potential context interleaving that may be of value a context-coverage criterion requires.

When an observed failure happens, repeating it is a common method to track to its original defect. Collecting all the runtime information can help to achieve this purpose. However, when data is from outside sensors, the task can be difficult [33]. Asynchronous updating of context information can also lead to inconsistencies between external states and internal states. To our best knowledge, these problems have not been thoroughly discussed and new methods for resolving them needs to be explored.

4 Relationships among Challenges

In this section we give our analysis of the relationship among the four identified categories of challenges. As shown in Figure 1, an ontology diagram is built to illustrate these challenges and their relationships.

There are two outstanding features in context testing data, data defects and being continuous, which greatly affect the generation and usage of testing data. Testing criteria are used to evaluate how well software can be tested. The criteria can be used to direct testing data generation and usage, and are also related to adaptation and testing execution. Adaptation can be erroneous and continuous. It should consider context-testing data because continuous context can affect the adaptation as discussed in section 3. Testing execution should not only consider all the challenges from aforementioned categories, but it also needs to consider new mechanisms for implementation of testing plans, e. g., collecting run time data.

5 Conclusion and Future Work

In this paper, we study the challenges of testing context-aware software, divide them into four categories and present the solutions current researchers use to overcome those challenges. After analyzing the relationship among the challenges of the four categories, we developed an ontology diagram to represent the challenges and their relationships. As far as we know, there is no automatic testing framework that considers all of the above challenges. We are currently building such a framework as an execution platform to ease the difficulty of testing context aware software. We will concentrate on addressing the challenges mentioned in the category of testing execution. Since context plays an important part in assuring the quality of context-aware software, we also plan to collect data from context-aware software testing processes and try to find the fault patterns that lead to system error or failure with respect to data inconsistency and adaptation.

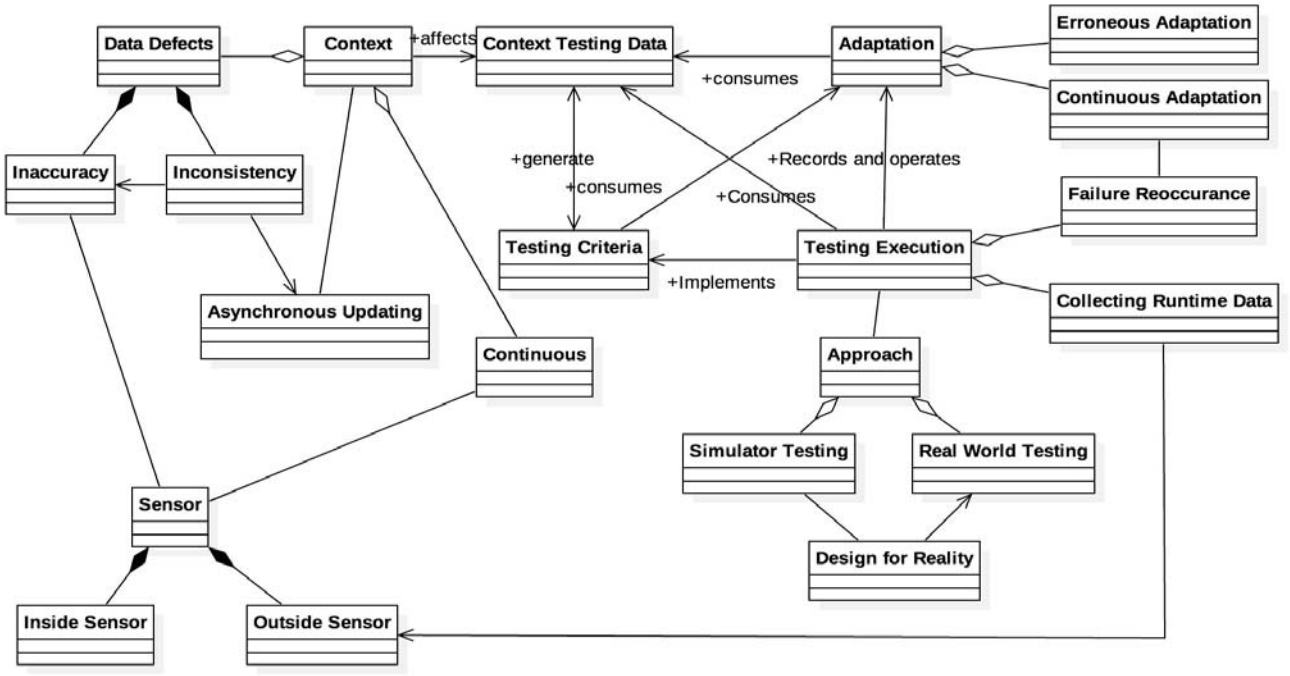


Figure 1: The ontology of identified testing challenges and their relationships

References

[1] Tobias Griebe, Volker Gruhn. “A model-based approach to test automation for context-aware mobile applications”. In Proceedings of the 29th Annual ACM Symposium on Applied Computing (SAC '14). ACM, New York, NY, USA, 420-427. 2014

[2] Vaninha Vieira, Konstantin Holl, and Michael Hassel. “A context simulator as testing support for mobile apps”. In Proceedings of the 30th Annual ACM Symposium on Applied Computing (SAC '15). ACM, New York, NY, USA, 535-541. 2015

[3] Minsu Jang, Jaehong Kim, Joo-Chan Sohn. "Simulation framework for testing context-aware ubiquitous applications" ICACT 2005. The 7th International Conference on Advanced Communication Technology, vol.2, no., pp.1337-1340, 0-0 0. 2005

[4] Hao Yan and Ted Selker. “Context-aware office assistant”. In Proceedings of the 5th international conference on Intelligent user interfaces (IUI '00). ACM, New York, NY, USA, 276-279. 2000

[5] Sven Meyer and Andry Rakotonirainy. “A survey of research on context-aware homes”. In Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003 - Volume 21 (ACSW Frontiers '03), Chris Johnson, Paul Montague, and Chris Steketee (Eds.), Vol. 21. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 159-168. 2003

[6] Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg. “A survey on context-aware systems”. Int. J. Ad Hoc Ubiquitous Comput. 2, 4 (June 2007), 263-277. 2007

[7] Zhimin Wang, Sebastian Elbaum, David Rosenblum. "Automated Generation of Context-Aware Tests" ICSE 2007. 29th International Conference on Software Engineering, vol., no., pp.406,415, 20-26. 2007

[8] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal. "Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities" HPCC '08. 10th IEEE International Conference on High Performance Computing and Communications, vol., no., pp.5-13, 25-27. 2008

[9] https://blogs.synopsys.com/configurablethoughts/2012/05/sensing-your-world/

[10] Stefano Ceri, Florian Daniel, Maristella Matera, and Federico M. Facca. “Model-driven development of context-aware Web applications”. ACM Trans. Internet Technol. 7, 1, Article 2 . 2007

[11] Dik Lun Lee, Qiuxia Chen. “A model-based WiFi localization method”. In Proceedings of the 2nd international conference on Scalable information systems (InfoScale '07). ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium, Article 40 , 7 pages. 2007

[12] Jalal Mahmud, Jeffrey Nichols, and Clemens Drews. “Home Location Identification of Twitter Users”. ACM Trans. Intell. Syst. Technol. 5, 3, Article 47, 21 pages. 2014

[13] Michele Sama, David S. Rosenblum, Zhimin Wang, and Sebastian Elbaum. “Model-based fault detection in context-aware adaptive applications”. In Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering (SIGSOFT '08/FSE-16). ACM, New York, NY, USA, 261-271. 2008

[14] Bill N. Schilit, Marvin M. Theimer, “Disseminating Active Map Information to Mobile Hosts”. IEEE Network, 8(5) 22-32. 1994

[15] Brown, P.J. “The Stick-e Document: a Framework for Creating Context-Aware Applications”. Electronic Publishing ’96 259-272. 1996

[16] Dey, A.K., Abowd, G.D., Wood, A. “CyberDesk: A Framework for Providing Self-Integrating Context-Aware Services”. Knowledge-Based Systems, 11 3-13. 1999

[17] Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, Pete Steggles. “Towards a Better Understanding of Context and Context-Awareness”. HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing. Publisher: Springer-Verlag. September 1999

[18] Eleanor O'Neill, David Lewis, Kris McGlinn, and Simon Dobson. “Rapid user-centred evaluation for context-aware systems”. In Proceedings of the 13th international conference on Interactive systems: Design, specification, and verification (DSVIS'06), Gavin Doherty and Ann Blandford (Eds.). Springer-Verlag, Berlin, Heidelberg, 220-233. 2006

[19] Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg. “A survey on context-aware systems”. Int. J. Ad Hoc Ubiquitous Comput. 2, 4, 263-277. June 2007

[20] Licia Capra, Wolfgang Emmerich, Cecilia Mascolo. "CARISMA: context-aware reflective middleware system for mobile applications". IEEE Transactions on Software Engineering, vol.29, no.10, pp.929,945, Oct. 2003

[21] Kristian Ellebæk Kjær. “A survey of context-aware middleware”. In Proceedings of the 25th conference on IASTED International Multi-Conference: Software Engineering (SE'07), W. Hasselbring (Ed.). ACTA Press, Anaheim, CA, USA, 148-155. 2007

[22] Tao Gu, Hung Keng Pung, Da Qing Zhang, “A service-oriented middleware for building context-aware services”, Journal of Network and Computer Applications, Volume 28, Issue 1, Pages 1-18, ISSN 1084-8045. January 2005

[23] Qin, Weijun; Shi, Yuanchun; Suo, Yue, “Ontology-based context-aware middleware for smart spaces”. Tsinghua Science and Technology , vol.12, no.6, pp.707,713, Dec. 2007

[24] Di Zheng; Hang Yan; Jun Wang, “Research of the Middleware Based Quality Management for Context-Aware Pervasive Applications”. 2011 International Conference on Computer and Management (CAMAN), vol., no., pp.1,4, 19-21. May 2011

[25] Vaninha Vieira, Konstantin Holl, and Michael Hassel. “A context simulator as testing support for mobile apps”. In Proceedings of the 30th Annual ACM Symposium on

Applied Computing (SAC '15). ACM, New York, NY, USA, 535-541. 2015

[26] Heng Lu, Chan W.K., Tse T.H.. "Testing pervasive software in the presence of context inconsistency resolution services". ICSE '08. ACM/IEEE 30th International Conference on Software Engineering, vol., no., pp.61,70, 10-18 May 2008

[27] Chang Xu and S. C. Cheung. "Inconsistency detection and resolution for contextaware middleware support". In Proceedings of the 10th European software engineering conference held jointly with 13th ACM SIGSOFT international symposium on Foundations of software engineering (ESEC/FSE-13). ACM, New York, NY, USA, 336-345. 2005

[28] Heng Lu, W. K. Chan, T. H. Tse. "Testing context-aware middleware-centric programs: a data flow approach and an RFID-based experimentation". SIGSOFT '06/FSE-14: Proceedings of the 14th ACM SIGSOFT international symposium on Foundations of software engineering. November 2006

[29] Suman Nath. "ACE: exploiting correlation for energy-efficient and continuous context sensing". In Proceedings of the 10th international conference on Mobile systems, applications, and services (MobiSys '12). ACM, New York, NY, USA, 29-42. 2012

[30] Shouling Ji, Jing (Selena) He, A. Selcuk Uluagac, Raheem Beyah, and Yingshu Li. "Cell-based snapshot and continuous data collection in wireless sensor networks". *ACM Trans. Sen. Netw.* 9, 4, Article 47 (July 2013), 29 pages. 2013

[31] Siyuan Chen, Shaojie Tang, Minsu Huang, Yu Wang. "Capacity of Data Collection in Arbitrary Wireless Sensor Networks" in *INFOCOM, 2010 Proceedings IEEE* , vol., no., pp.1-5, 14-19. March 2010

[32] Balázs Hidasi and Domonkos Tikk. "Approximate modeling of continuous context in factorization algorithms". In *Proceedings of the 4th Workshop on Context-Awareness in Retrieval and Recommendation* (CARR '14). ACM, New York, NY, USA, 3-9. 2014

[33] Chang Xu, S.C. Cheung, Xiaoxing Ma, Chun Cao, Jian Lu. "Adam: Identifying defects in context-aware adaptation". Journal of Systems and Software, Volume 85, Issue 12, Pages 2812-2828, ISSN 0164-1212. December 2012

[34] Stefan Taranu and Jens Tiemann. "General method for testing context aware applications". In Proceedings of the 6th international workshop on Managing ubiquitous communications and services (MUCS '09). ACM, New York, NY, USA, 3-8. 2009

[35] Edwin J.Y. Wei, Alvin T.S. Chan. "CAMPUS: A middleware for automated context-aware adaptation decision making at run time". Pervasive and Mobile Computing, Volume 9, Issue 1, Pages 35-56, ISSN 1574-1192. February 2013

[36] Paul Ammann and Jeff Offutt. "Introduction to software testing". Cambridge University Press. 2008

Resolving Cyclic Dependencies – Design for Testability

Yurii Boreisha, Oksana Myronovych

Department of Computer Science, Minnesota State University Moorhead, Moorhead, MN, USA

Department of Computer Science, North Dakota State University, Fargo, ND, USA

Abstract - *This paper is dedicated to the issues of resolving cyclic/circular dependencies in contemporary software systems. Interface-based and event-based approaches are the best practices to deal with dependency problems and provide for creating software that's easier to test. The main design patterns to support these practices are the dependency injection and observer/listener. It is demonstrated how the C# mechanism based on events and delegates can be used to implement the observer/listener design pattern to resolve cyclic/circular dependencies.*

Keywords: Dependencies and layering, design for testability, acyclic dependencies principle.

1 Introduction

To produce high-quality software, developers must strive to ensure that their code is maintainable and testable. The code also should be adaptive to change. Agile development, the leading trend in the system development, helps keep system development projects responsive to change [6, 10].

Agile developers follow these steps to create contemporary high quality software [3, 4, 8]:

- Detect problems by following agile practices.
- Diagnose problems by applying design principles.
- Solve problems by applying appropriate design patterns.

SOLID is the acronym for a set of principles, patterns and practices that, when implemented together, make code adaptive to change. The SOLID practices were introduced by Bob Martin almost 15 years ago:

S – Single responsibility principle: "a class should have only one reason to change".

O – Open/close principle: "software entities (classes, modules, functions, etc.) should be open for extension, but close for modification".

L – Liskov substitution principle: "subtypes must be substitutable for their base types".

I – Interface segregation principle: "clients should not be forced to depend upon methods that they don't use; interfaces belong to clients, not to hierarchies".

D – Dependency inversion principle: "higher-level modules should not depend on low-level modules; both should depend on abstractions. Abstractions should not depend upon details; details should depend upon abstractions".

Even taken in isolation, each of these principles are very useful. When used in collaboration, they give the code a structure that lends itself to change. SOLID patterns and practices are merely tools for you to use. Deciding when and where to apply any pattern or practice is a part of the art of software development.

In the context of software engineering, a broadly accepted definition for testability is "the ease of performing testing". Testing is the process of checking software to ensure that it behaves as expected, contains no errors, and satisfies its requirements. The ability to test software, and in particular to test software automatically, is an aspect of extraordinary importance because automated tests give us a mechanical way to figure out quickly and reliably whether certain features that worked at some point still work after we make some required changes. In addition, tests make it possible for us to calculate metrics and take the pulse of a project, as well. Testing is an important part of the change [1, 2, 9].

Testable software is inherently better from a design perspective. Design for testability (DfT) was adapted to software engineering and applied to test units of code through tailor-made programs. DfT defines three attributes that any unit of software must have to be easily testable: control, visibility, and simplicity. When you apply control, visibility, and simplicity to the software development process, you

end up with relatively small building blocks that interact only via contracted interfaces [5, 7, 11].

To keep the code adaptive to change and testable one must manage dependencies effectively. This applies at all levels of the software – from architectural dependencies between subsystems to implementation dependencies between individual methods.

When the dependency structure is incomprehensible, a change in one module can cause a direct side effect in another, seemingly unrelated module. There are established patterns that help you arrange your application in the short term so that it can adapt to changes in the long term. Layering is one of the most common architectural design patterns, and MVC, MVVM, Web API, etc., provide examples of widely used implementations of the layering.

The following design principles also target the proper dependency management:

- Coupling is a qualitative measure of how closely the classes in a design class diagram are linked. A simple way to think about coupling is as the number of navigation arrows on the design class diagram. Low coupling is usually better for a system than high coupling. In other words, fewer navigation visibility arrows indicate that a system is easier to understand and maintain.
- Cohesion refers to the consistency of the functions within a single class. Cohesion is a qualitative measure of the focus or unity of purpose within a single class. Classes with low cohesion have several negative effects. First, they are hard to maintain. Second, it is hard to reuse such classes. Finally, classes with low cohesion are usually difficult to understand, their functions are intertwined and their logic is complex. High cohesion is the most desirable.
- Indirection – is a design principle in which an intermediate class (or component) is placed between two classes (or system components) to decouple but still link them. Inserting an intermediate object allows any variations in one system to be isolated in that intermediate object. Indirection is also very useful for many corporate security systems (for example, firewalls and proxy servers).
- Acyclic dependencies principle states that the dependency graph of packages or components should have no cycles. This implies that the dependencies form a directed acyclic graph.

There is a strict relationship between coupling and testability. A class that can't be easily instantiated in a test has some serious coupling problems. If the problem of coupling between components is not properly addressed in the design, you end up testing components that interact with others, producing something that looks more like an integration test than a unit test. Integration test are still necessary, but they ideally should run on individual units of code (for example, classes) that already have been thoroughly tested in isolation. Integration tests are not run as often as unit tests because of their slow speed and higher setup costs.

By keeping coupling under control at the design phase we improve testability. Conversely, by pursuing testability, we keep coupling under control and end up with a better design for the software.

Interface-based and event-based approaches are the best practices to deal with dependency problems and provide for creating software that's easier to test. The idea of writing code against interfaces rather than implementations is widely accepted and applied. There is a number of options to accomplish this, for example, dependency injection (DI) and inversion of control (IoC) container [6].

This paper is dedicated to the issues of resolving cyclic/circular dependencies in contemporary software systems. We discuss these issues from the point of view of the event-based approach. It is demonstrated how the C# mechanism based on events and delegates can be used to implement the observer/listener design pattern to resolve cyclic/circular dependencies.

2 Dependencies and Layering

Layering is an architectural pattern that encourages you to think of software components as horizontal layers of functionality that build on each other to form a whole application. Components are layered, one on top of the other, and the dependency direction must always point downward. That is, the bottom layer of the application has no dependencies, and each layer upward depends on the layer immediately below it. At the top of the stack is the user interface. If the application is a service layer, the top layer will be the API that clients will use to interact with the system.

The primary reason for using layers (and tiers) is the Separation of Concerns (SoC). As an architect, you determine which layer talks to which layer, and you have testing, code inspection, and perhaps check-in policies to enforce these rules. However, even when two layers are expected to collaborate, you don't want them to be tightly coupled. In this regard the dependency inversion principle (DIP) helps a lot.

DIP is the formalization of a top-down approach to defining the behavior of any significant class

method. In using this top-down approach, we focus on the work flow that happens at the method level rather than focus on the implementation of its particular dependencies. At some point, though lower-level classes should be linked to the mainstream code. DIP suggest that this should happen via injection.

From the point of view of MS Visual Studio all classes and interfaces are contained in assemblies. When correctly organized, your assemblies will contain classes and interfaces that only pertain to a single group of related functionality [6].

Groups of two or more interrelated assemblies form components of the software system that is being developed. These components interact in a similarly well-defined and structured fashion. Components are not physical artifacts of deployment, like assembly dynamic-link libraries (DLLs), but are logical groupings of assemblies that share a similar theme.

In dependency management, components are no different from other programming constructs at lower levels. As with methods, classes, and assemblies, you can consider layers to be nodes in the dependency graphs. The same rules apply: keeping the dependency graph acyclic and ensuring a single responsibility.

There are several common layering patterns from which to choose for any project. The differentiating factor between the layering patterns is simply the number of layers used. The number of layers required correlates to the complexity of the solution; the complexity of the solution correlates to the complexity of the problem.

The difference between layers and tiers is the difference between the logical organization and physical deployment of code. Logically, you could separate an application in two or more layers, but physically deploy it into one tier.

With every tier you deploy to, you accept that you are crossing a network boundary, and with that comes a temporal cost: it is expensive to cross a processing boundary within the same machine, but it is much more expensive to cross a network boundary. However, deploying in tiers has a distinct advantage because it allows you to scale your application. If you deploy a web application that consists of a user interface layer, a logic layer, and a data access layer onto a single machine – thus a single tier – that machine now has a lot of work to do, and the number of users you can support will necessary be lower. Were you to split the application's deployment into two tiers – putting the database on one tier and the user interface and logic layers on another – you could actually scale the user interface layer both horizontally and vertically.

To scale vertically, you just increase the power of the computer by adding memory or processing units. This allows the single machine to achieve more by itself. However, you can also scale horizontally by adding completely new machines that perform exactly the same task. There would be multiple machines to host the web user interface code and a load balancer that would direct clients to the least busy machine at any point in time. Of course, this is not a panacea to supporting more concurrent users on a web application. This requires more care with data caching and user authentication, because each request made by a user could be handled by a different machine.

3 Navigation Visibility and Dependency Relationships

Navigation visibility refers to the ability of one object to interact with another object. When building navigation visibility we should transform the problem domain model into the corresponding design class diagram. Here are a few general guidelines:

- One-to-many associations that indicate a superior/subordinate relationship are usually navigated from superior to the subordinate.
- Mandatory associations, where objects in one class can't exist without objects of another class, are usually navigated from the independent class to the dependent class.
- When an object needs information from another object, a navigation arrow might be required, pointing either to the object itself or to its parent in a hierarchy.
- Navigation visibility arrows may be bidirectional, for example when an object might need to send a message to another object as well as the reverse.

To resolve the problem of bidirectional navigation visibility arrows one can apply the observer/listener design pattern. It is a design pattern in which an object, called the publisher, maintains a list of its dependents, called subscribers (observers, listeners), and notifies them automatically of any state changes, usually by calling one of their methods. It is mainly used to implement distributed event handling systems. This pattern is also a key part in the familiar MVC architectural pattern. The observer/listener pattern is implemented in numerous programming libraries and systems, including almost all GUI toolkits.

The observer/listener pattern can cause memory leaks, known as the lapsed listener problem, because in basic implementation it requires both explicit registration and explicit deregistration, as in the dispose pattern, because the publisher holds strong

references to the observers/listeners, keeping them alive.

Herewith we provide an implementation of the observer/listener pattern based on C# mechanism of events and delegates. This implementation is free from the lapsed listener problem.

Let's create class *Account* as it's shown in Figure 1, with the related class *AccountEventArgs* like in Figure 2. The *Subscribe*r class interface can look like in Figure 3. The related *Subscriber* classes should implement this interface as it's shown in Figure 4. The *Publisher* class may look like in Figure 5.

This implementation is used to create and agreement/contract between the *Publisher* and *Subscriber* classes by determining the signature of the corresponding event handler method. The code in Figure 6 shows the *Main* class where a publisher object and a number of subscriber objects are instantiated.

From now on all subscribers will be informed about the changes made by the *Publisher* to the object of class *Account* (they will listen for the related event, *Published*, to fire); bidirectional visibility arrows will disappear.

```
public class Account {
  public string AccountNumber { get; set; }
  public override string ToString() {
    return string.Format("Account {0}", AccountNumber);
  }
}
```

Figure 1: Class Account

```
public class AccountEventArgs : EventArgs {
  public Account MyAccount { get; set; }
}
```

Figure 2: Class AccountEventArgs

```
public interface ISubscriber {
  void OnPublished(object source, AccountEventArgs args);
}
```

Figure 3: Interface ISubscriber

```
public class Subscriber1 : ISubscriber {
  public void OnPublished(object source, AccountEventArgs args) {
    Console.WriteLine("Subscriber1 getting info about {0}", args.MyAccount);
  }
}

public class Subscriber2 : ISubscriber {
  public void OnPublished(object source, AccountEventArgs args) {
    Console.WriteLine("Subscriber2 getting info about {0}", args.MyAccount)
  }
}
```

Figure 4: Classes Subscriber1 and Subscriber2

```
public class Publisher {
  public void Publish(Account account) {
    Console.WriteLine("Working with: {0}", account);
    // To do: some stuff
    OnPublished(account);
  }
```

```
  public event EventHandler<AccountEventArgs> Published;

  protected virtual void OnPublished(Account account) {
    if (Published != null) // any subscribers?
          Published(this, new AccountEventArgs() { MyAccount = account });
  }
}
```

Figure 5: Class Publisher

```
class Program {
  static void Main(string[] args) {
    Account account = new Account() { AccountNumber = "12345" };
    Publisher publisher = new Publisher();

    ISubscriber sub1 = new Subscriber1();
    publisher.Published += sub1.OnPublished;

    ISubscriber sub2 = new Subscriber2();
    publisher.Published += sub2.OnPublished;

    publisher.Publish(account);
  }
}
```

Figure 6: Class Main

The observer/listener pattern is also used to implement the two-way data binding operations in Universal Windows Platform applications to resolve cyclical dependencies between view and domain layers. The data binding does not know when the data to which it is bound has been changed. The domain layer object needs to inform the data binding of any modifications by sending a *PropertyChanged* event to the view layer. This event is part of the interface named *INotifyPropertyChanged*, and all object that support two-way data binding should implement this interface (it is defined in the *System.ComponentModel* namespace) as in Figure 7.

The *OnPropertyChanged* method raises the *PropertyChanged* event. The *PropertyChangedEventArgs* parameter to the *PropertyChanged* event should specify the name of the property that has changed. This value is passed in as a parameter to the *OnPropertyChanged* method as it's shown in Figure 8 for the property *EmailAddress*.

```
public class Customer : INotifyPropertyChanged {
   …
   public event PropertyChangedEventHandler PropertyChanged;

       protected virtual void OnPropertyChanged(string propertyName) {
          if (PropertyChanged != null) {
                 PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
                 }
        }
}
```

Figure 7: Domain layer class Customer

```
private string _emailAddress;
public string EmailAddress {
   get { return this._emailAddress; }
        set {
           this._emailAddress = value;
                  this.OnPropertyChanged(nameof(EmailAddress));
        }
}
```

Figure 8: *OnPropertyChanged* method call

4 Dealing with Dependencies

UML design class diagram contains the final definition of each class in the final object-oriented software system. The primary source of information for this diagram is the problem domain model. The navigation visibility graph of the design class diagram should not contain cycles.

UML package diagram is a high-level diagram that allows designers to associate classes of related groups. The classes are placed inside the appropriate package based on the layer to which they belong. Dependency relationship is a relationship between packages, classes, or use cases in which a change in the independent item requires a change in the dependent item. Dependency direction must always point downward.

As we already mentioned above the interface-based and event-based approaches are the best practices to deal with dependency problems and provide for creating software that's easier to test. The main design patterns to support these practices are the dependency injection and observer/listener (described in the previous section).

UML diagrams free from cyclic/circular dependencies leverage the software testability. As far as testing is concerned, one could say that there are the following main types of dependencies: those that you want to resolve (like cyclic/circular ones), those that you want to ignore, and those with which you want to interact but in a controlled manner. The following technique is used for the latter case.

According to the concept of the testing in isolation, it is highly recommended to isolate the class being tested from all its dependencies. This can happen only if the class is designed in a loosely coupled manner. In an object-oriented scenario, for example, class A depends on class B when any of the following conditions are verified:

- Class A derives from class B (inheritance).
- Class A includes a member of class B (aggregation, composition).
- One of the methods of class A invokes a method of class B.
- One of the methods of class A receives or returns a parameter of class B.
- Class A depends on a class that in turn depends on class B.

To neutralize such dependencies we use test doubles. If the classes under the test support the DI, providing test doubles is relatively easy [5].

A test double is a class you write and add to the test project. This class implements a given interface or inherits from a given base class. After you have the instance, you inject it inside the object under the test by using the public interface of the object being tested.

There are two main types of test doubles: fakes and mocks. A fake object is a relatively simple clone of an object that offers the same interface as the original object but returns hardcoded or programmatically determined values. A mock object does all that a fake does, plus something more.

In a way, a mock is an object with its own personality that mimics the behavior and interface of another object. Essentially, a mock accommodates verification of the context of the method call. With a mock, one can verify that a method call happens with the right preconditions and in the correct order with respect to other methods in the class. For the level of flexibility you expect from a mock, you need an ad hoc mocking framework.

5 Conclusions

There is a strict relationship between coupling and testability. A class that can't be easily instantiated in a test has some serious coupling problems. If the problem of coupling between components is not properly addressed in the design, you end up testing components that interact with others, producing something that looks more like an integration test than a unit test. Integration test are still necessary, but they ideally should run on individual units of code (for example, classes) that already have been thoroughly tested in isolation.

Two major recent contributions to the software testing area are the definition of new frameworks for test execution (collectively referred as xUnit) and promote shorter cycles in the testing process, such as continuous integration (CI).

The basic idea behind CI is to commit, one or more times per day, all of the working copies of the software on which different developers or groups of developers are working. CI is related to the concept of automated test execution frameworks, in that a regression test suite should be automatically run against the code (ideally, prior to commit it) to help ensure that the codebase remains stable (i.e., no regression errors have been introduced) and continuing engineering efforts can be performed more rapidly. If some test fail, the developers responsible for the change must then correct the problems revealed by the tests. This approach is advantageous because it can reduce the amount of code rework that is needed in later phases of development, and speed up overall development time [9].

This paper is dedicated to the issues of resolving cyclic/circular dependencies in contemporary software systems. We discuss these issues from the point of view of the event-based approach. It is demonstrated how the C# mechanism based on events and delegates can be used to implement the observer/listener design pattern to resolve cyclic/circular dependencies.

6 References

[1] Boreisha, Y. and Myronovych, O. Genetic Algorithm and Mutation Analysis for Software Testing; Proceedings of the 2010 International Conference on Software Engineering Research and Practice, SERP2010, 247-252, July, 2010, Las Vegas, Nevada, USA.

[2] Boreisha, Y. and Myronovych, O. Modified Genetic Algorithm for Mutation-Based Testing; Proceedings of the 2009 International Conference on Software Engineering Research and Practice, SERP2009, 44-49, July, 2009, Las Vegas, Nevada, USA.

[3] Cooper, J.W. C# Design Patterns. Addison Wesley, 2003.

[4] Dasiewicz, P. Design Patterns and Object-Oriented Software Testing; IEEE CCECE/CCGEI, 904-907, May, 2005, Saskatoon.

[5] Esposito, D. Programming Microsoft ASP.NET MVC, 3rd Edition. Microsoft Press, 2014.

[6] Hall, M. G. Adaptive Code via C#. Agile Coding with Design Patterns and SOLID Principles. Microsoft Press, 2014.

[7] Khatri, S. and Chhillar, R. Improving the Testability of Object-Oriented Software During Testing and Debugging Processes, International Journal of Computer Applications, V 35, N 11, 24-35, 2011.

[8] Martin, R. and Martin, M. Agile Principles, Patterns, and Practices in C#. Prentice Hall, 2007.

[9] Orso, A. and Rothermel, G. Software Testing: A Research Travelogue (2000-2014); Proceedings of FOSE'14, 117-132, June, 2014, Hyderabad, India.

[10] Satzinger, J. et al. Systems Analysis and Design in a Changing World, 7th Edition. Cengage Learning, 2016.

[11] Suri, P. and Singhani, H. Object-Oriented Software Testability (OOST) Metrics Analysis; International Journal of Computer Applications, Technology and Research, V 4, N 5, 359-367, 2015.

Enhanced Fault Localization by Weighting Test Cases with Multiple Faults

Jaehee Lee [1,2], Jeongho Kim[2], Eunseok Lee [2]
[1] Software Engineering Laboratory, Software R&D Center, Samsung Electronics, Seoul, South Korea
[2] Dept. of Information and Communication Engineering, Sungkyunkwan University, Suwon, South Korea

Abstract - *Fault localization is known to be one of the most time-consuming and difficult tasks in the debugging process. Many fault localization techniques have been proposed to automate this step, but they have generally assumed the single-fault situation, which reduces the performance when multiple faults are encountered. This paper proposes new weighting techniques to improve the effectiveness of spectrum-based fault localization by using information extracted from failed test cases that were caused by multiple faults. We evaluate the performance of our technique with six different metrics using the Siemens test suite and space with 159 multiple-fault versions. Based on the experimental study, we observe that our technique outperforms the baselines, in terms of the average expense maximum, by 12 %.*

Keywords: fault localization, multiple fault, spectrum-based, weighting, classification

1 Introduction

In software engineering, debugging is necessary in order to maintain code quality. However, manually finding the locations of faults is one of the most costly and time-consuming tasks that developers must carry out [1]. In attempts to alleviate this problem, various automatic fault localization techniques have been proposed.

Spectrum-based fault localization (SFL) techniques extract program spectra, which are the execution profiles of program statements and information about whether tests pass or fail [2][3][11][18][22]. This information is used with a ranking metric to rank the program statements according to how likely they are to be buggy.

SFL techniques have been widely used by researchers and developers due to their simplicity, light mechanism, and accuracy. However, they mostly assume only single-fault situations. However, most real world programs contain more than one fault and many failures are caused by multiple faults. The assumption of a single fault degrades the effectiveness of the proposed techniques.

To alleviate this problem, in this paper, we propose a new weighting technique to improve the effectiveness of spectrum-based fault localization by using information extracted from failed test cases that were caused by multiple faults. Our approach is based on the assumption that, if we identify failed test cases executing multiple faults, each statement from these test cases will have a higher likelihood of being buggy than those from test cases executing only a single fault. The empirical results from this study indicate that our approach is promising. The main contributions of this paper are:

1. We propose an enhanced fault localization technique for multiple-fault environments by using weighting to improve the effectiveness of SFL; this is done by including information extracted from failed test cases that were caused by multiple faults.
2. The performance of our technique is verified experimentally. We perform experiments on various types of real programs [16], including the Siemens test suite and space with various metrics containing 159 multiple-fault versions; these are shipped with between two and eight faults.

The remainder of this paper is organized as follows. In Section II, we introduce the necessary background and related work on SFL and the previous research. In Section III, we describe our technique in detail. Section IV goes on to describe the design of our experiments. Section V reports the results and analysis. Threats to the validity of our techniques are discussed in Section VI. Finally, Section VII presents our future work and conclusions.

2 Background and Related work

2.1 Spectrum-based fault localization

Fault localization techniques intend to reduce the cost of debugging by automating the process that is used to find the location of faults in a program. Among them, spectrum-based fault localization (SFL) techniques have been widely used due to their simplicity (i.e., no modeling or complex computation) and relatively high effectiveness [18].

SFL techniques assign a suspicious score to each statement (branches, predicates, and functions can be used) in the program based on the number of passed and failed test cases in the test suite that executed the statement. The basic assumption of SFL is that, if there is failure in a certain

executed test case, a fault exists among the statements that were visited in the test during execution. However, we cannot expect to determine the exact fault location by using only the failed test case. Therefore, the passed test cases are also used to narrow down the fault location.

Table 1 describes some of the notations that are commonly used in the fault localization field. h_i contains binary information indicating whether the statement was visited or not.

Table 1 Relation between statement hit and test result

Notation	Value		Description	
	h_i	e_i	Statement hit	Test result
$a11$	1	1	O	Fail
$a10$	1	0	O	Pass
$a01$	0	1	X	Fail
$a00$	0	0	X	Pass

e_i contains binary information that describes the test result (pass or fail). If the test case (T_i), which is one of the test cases in the test pool, was executed during the runtime and the test result was fail, a certain statement (s_i) can be described as either $a11$ (this line was visited) or $a01$ (this line was not visited). In the same way, if the test result was pass, a certain statement (s_i) can be described as either $a10$ (this line was visited) or $a00$ (this line was not visited). Therefore, according to the test result, every statement will be counted with one of four types of notation ($a11$, $a10$, $a01$, or $a00$). Table 2 gives an example with five test cases; of these, the first four fail.

Table 2 Example of program spectra

	T1	T2	T3	T4	T5	$a00$	$a01$	$a10$	$a11$
Stmt1	1	1	1	0	1	1	1	0	3
Stmt2	1	1	0	0	0	0	2	1	2
Stmt3	1	0	1	1	1	1	1	0	3
Stmt4	0	1	0	1	1	1	2	0	2
Result	1	1	1	1	0				

A variety of ranking metrics have been proposed and studied, including Tarantula [2], Ochiai [3], Jaccard [21], AMPLE [20], Naish2 [22], GP13 [17], and Hybrid [11]. Each of these calculates suspicious fault ratios in a different way. We describe two representative ranking metrics below.

$$\text{Tarantula} = \frac{\frac{a11}{a11+a01}}{\frac{a11}{a11+a01}+\frac{a10}{a10+a00}} \quad (1)$$

J. A. Jones et al. developed Tarantula [2], which aims to show the suspiciousness of every statement. In addition, they conducted an experimental program based on the language C.

$$\text{Naish2} = a11 - \frac{a10}{a10+a00+1} \quad (2)$$

L. Naish et al. proposed an optimal risk evaluation formula [22] with respect to their model and performance measurement. To simulate a single-fault program, a model program and the average performance over all possible multisets of the execution paths was used for performance measure.

2.2 Effectiveness of SFL in the context of multiple faults

Many approaches have been proposed in an attempt to elucidate fault interactions and their repercussions. J. A. Jones et al. [2] reported that the effectiveness of the techniques declines for all faults as the number of faults increases. Debroy and Wong [4] explored the idea of fault interference by examining the Siemens test suite. N. DiGiuseppe et al. extended their research by classifying fault interactions into one of four types: independent, synergy, obfuscation, and multiple [6]. They also verified the total cost to resolve all faults as the number of faults increases [5].

2.3 Classifying failing test cases

In [8], J. A. Jones and colleagues investigated the use of failure clustering to remove "noise" caused by one fault inhibiting the localization of another. W. Hogerle et al. explored various alternative clustering algorithms to increase parallelism by using algorithms from integer linear programming [10]. However, as they mentioned, relying on the fact that each cluster focuses on a single fault does not seem realistic.

Yu et al. [9] proposed a technique that can be used to distinguish failing test cases that executed a single fault from those that executed multiple faults. To achieve the goal, their technique uses extracted information from a set of fault localization ranked lists, each of which is produced for a certain failing test and the distance between a failing test and the passing test that most resembles it. They mainly aimed to separate failing test cases that executed a single fault in order to apply an existing approach (SFL, automated fault repairing, failure clustering, etc.). Alternatively, our approach focuses on failed test cases executing multiple faults.

2.4 Weighting test cases in SFL

Naish et al. [12] proposed a weighting strategy for failed tests. Failed tests that cover few statements provide more information than other types of failed tests. Thus, they assumed that the weight of a failed test is inversely proportional to the number of statements exercised in the test. In [13], they also proposed an approach where the frequency execution count of each program statement, executed by a respective test, is used.

Bandyopadhyay et al. [14] extended the idea of the nearest neighbor model [18] to utilize the relative importance of different passing test cases for calculation of suspiciousness scores. They stated that the importance of a passing test case is proportional to its average proximity to the failing test cases.

Y. Li et al. [15] proposed a weight-based refinement for SFL techniques depending on the execution information and the test status. They treated test cases as being unequally important and improved the effectiveness by exploiting varying weights according to the distribution of the test cases.

However, all of these weighting techniques assume a single-fault situation; we expect that their performance will be degraded in a multiple-fault environment.

3 Proposed Approach

In this chapter, we explain the detailed mechanism of our suggested weighting approach. Our approach is based on the assumption that if we identify failed test cases executing multiple faults, each statement of these test cases will be more suspicious than those executing a single fault stochastically. Hence, we assign more weight to statements visited by multiple test cases.

3.1 Overall procedure

In Figure 1, we describe the overall procedure of our proposed approach; this is similar with the general SFL technique. First, the test suite and target subject program are inserted as the input data. Next, the spectrum data are extracted by executing test suites on the subject program. Then, two additional steps are executed. When classifying the test case, failed test cases are classified into one of two groups: single-fault-executing test cases and multiple-fault executing-test cases. After classifying the test cases, we grant more weight to multiple-fault-executing test cases. Finally, a ranked list of each statement, in descending order by the suspicious score, is calculated according to each ranking metric.

Any ranking metric can be used with our approach. In this paper, we use Tarantula, Ochiai, Jaccard, AMPLE, Naish2, and GP13, which are popular in the literature.

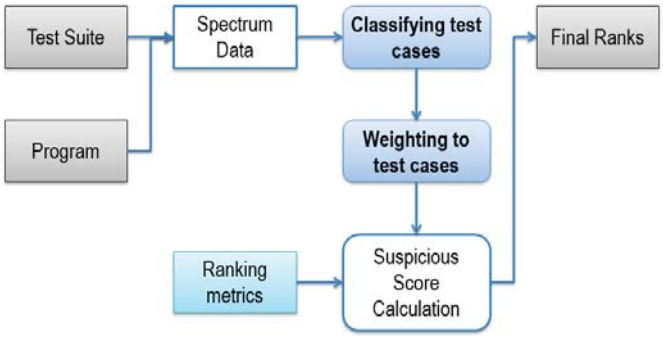


Figure 1 Overall procedure of proposed approach

3.2 Classifying test cases

Basically, we extend Yu's technique [9] to classify failed test cases as either single-fault-executing test cases or multiple-fault-executing cases. This is done by using the pattern of the spectrum to classify the test cases. We modified their technique by using the hamming distance to calculate the distance between the binary coverage information of the test cases. We checked the accuracy of our results to determine whether this can be used to classify failed test cases; our results were found to be similar to theirs, which indicates that the performance is sufficiently high.

3.3 Weighting test cases

After the multiple-fault-executing test cases were classified, proper weights were granted to each statement that was visited by the test. We used relative weights; for instance, we assigned a weight $\omega t = 1$ to $a11$ values in the case of single-fault test cases, as is done in the general SFL technique. Alternatively, we assigned a weight $\omega t = \alpha$ (>1) to $a11$ values in the case of multiple-fault test cases. The α value is introduced as a parameter to represent different weights to multiple test cases. We check the performance of each metric according to variations of the α value.

To calculate $a11$ for each statement, we take the sum of the weights of the failed test cases.

$$a^s 11 = \sum_{s,t=1} \omega t \frac{N}{W} \qquad (3)$$

ωt : the relative weight of each failed test case

N : the number of failed test cases

W : the sum of the relative weights ωt

As an example, with data from Table 2, the relative weights for test cases 1-4 are 1.6, 0.8, 0.8, and 0.8 if we assume that T1 is a multiple-fault-executing test case and α=2 (N=4

and W=5). Accordingly, the $a11$ values for statements 1-4 are changed to 3.2, 2.4, 3.2, and 1.6, respectively. The $a01$ values can be computed by using a similar weight sum, or we can simply use the total number of failed tests minus $a11$.

In Table 3, we describe the algorithm of our proposed approach:

- TC is the set of all test cases devised for the program being tested;
- TCn is a subset of TC;
- TP is the set of passed test cases;
- TF is the set of failed test cases;
- TFn is a subset of TF and contains one failed test case;

Table 3 Algorithm of proposed approach

```
1:  Run all test cases in TC
2:  while TF is not empty do
3:      for n=1 to N do
4:          TCn = TFn ∪ TP
5:              Classifying TFn to single or multiple faults
6:          Weighting to TFn according to classifying result
7:      end for
8:      Perform fault localization for each test cases in TF
9:      Re-run all test cases in TC
10: end while
```

In order to evaluate and compare the effectiveness of our proposed technique, we investigated the following research questions:

RQ1. How effective is our proposed weighting technique compared to existing (unweighted) approaches?

RQ2. Is there any dependency between the performance of the proposed weighting technique and different subject programs?

RQ3. Is there any dependency between the performance of the proposed weighting technique and different ranking metrics?

RQ4. Do different weighting parameters affect the result? What is the optimal threshold parameter?

4 Experimental Setup

This section presents the experimental setup for the empirical study.

4.1 Subject programs

Our empirical study involved eight subject programs, including the Siemens test suite and space from SIR (Software Infrastructure Repository) [16], along with their faulty versions and test cases. Table 4 provides more information about these subject programs and test cases. Note that, since the test suites for space are relatively large (13525), we opted to randomly select a smaller subset of 738 cases.

Table 4 Subject programs and test cases

Subject	Versions	LOC	Test cases	Description
print_tokens	7	536	4140	Lexical analyzer
print_tokens2	10	387	4115	Lexical anlayzer
replace	27	554	5540	Pattern replacement
schedule	4	425	2650	Priority scheduler
schedule2	9	766	2710	Priority scheduler
Tcas	41	173	1578	Altitude separation
tot_info	23	494	1052	Information measure
Space	38	6445	13525	Array definition language

4.2 Fault versions

We created 159 multiple-fault versions of the subject programs by taking different combinations of the available faults. We used one-fault version programs, where the fault was in a single line of source code, and discarded versions with runtime errors and those with no failed test cases. In each subject, for each occurrence of multiple faults, we generated up to the number of faulty versions using the SIR. Therefore, the exact number of multiple-fault versions for each subject is the same as the sum of the faulty versions shown in Table 4 (a total of 159 multiple-fault versions).

4.3 Evaluation metrics

According to the fault localization literature [8][18], fault localization evaluation metrics are defined as the percentage of the program that needs to be examined before reaching the first statement (when ranking metrics are used to order executable statements). As Equation (4) indicates, the range of possible values for the fault localization expense varies, and the effectiveness of the employed fault localization technique decreases as the expense value increases. This value is indicative of the time or effort that a developer spends while finding/using the ranks computed by the fault localization technique. This metric, which we refer to as the "expense", is computed by the following equation:

$$\text{Expense} = \frac{\text{rank of fault}}{\text{size of program}} * 100 \qquad (4)$$

5 Results and Analysis

In our experiments, we apply the weighted technique to all of the metrics mentioned in Section III and to all of the subject programs listed in Section IV. We also investigate its effectiveness in improving the performance of these SFL techniques on several subject programs. We present our experimental results in Figures 2, 3, 4, and 5. In each figure, the vertical axis (horizontal axis in Figure 2) represents the average expense required to examine the source code. All of the experiments were carried out on a Windows 7 machine with a 3.7 GHz Intel quad-core CPU and 8 GB of memory.

5.1 Effectiveness of the proposed weighting technique

Table 5 shows that our weighting technique achieves better or similar performance than existing unweighted techniques, with an average improvement of 7 %. These results are illustrated in Figure 2.

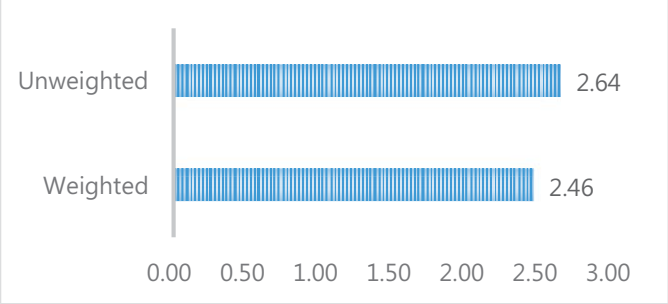


Figure 2 Total average expense (unweighted vs. weighted)

Table 5 Average expense for each subject program

Subject	Unweighted	Weighted	Improvement
print_tokens	2.52	**2.27**	9.92 %
print_tokens2	2.69	**2.47**	8.18 %
replace	3.40	**2.99**	12.06 %
schedule	2.70	**2.58**	4.44 %
schedule2	2.71	**2.61**	3.69 %
Tcas	3.33	**3.07**	7.81 %
tot_info	2.41	2.41	0.00 %
space	1.39	**1.26**	9.03 %
Total	2.64	2.46	7.02 %

5.2 Different subject programs

An additional consideration is made by looking into each subject program separately. Figure 3 depicts the average expense for each subject program. Our weighting technique outperforms almost all of the subject programs in terms of the average expense (with the exception of a single program). These improvements range between 3.69 % and 12.06 %. Our technique shows equivalent performance to the tot_info program. We speculate this range in performances comes from the characteristics of the program, which affect the results.

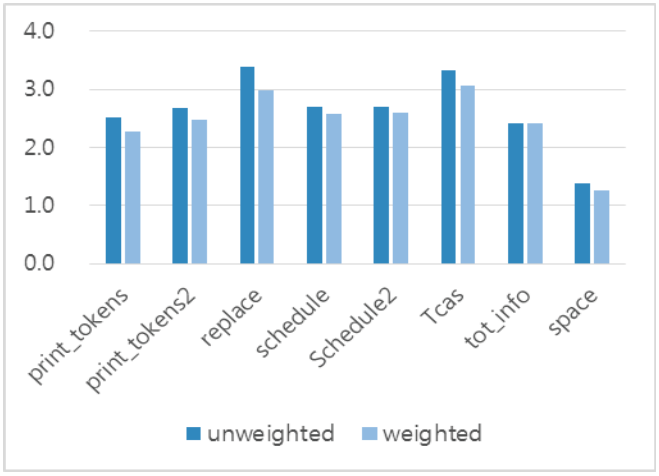


Figure 3 Average expense for each subject program

5.3 Different ranking metrics

Figure 4 visualizes the average expense for each ranking metric. Our weighting technique outperforms all of the metrics in terms of the average expense; this improvement ranges between 3.69 % and 16.30 %. We also speculate that this range in improvement comes from the characteristic of the various ranking metrics. Table 6 shows the results in greater detail.

Table 6 Average expense for each ranking metric

Subject	Unweighted	Weighted	Improvement
Tarantula	2.44	**2.35**	3.69 %
Ochiai	2.31	**2.22**	3.90 %
Jaccard	2.34	**2.24**	4.27 %
AMPLE	2.88	**2.78**	3.47 %
Naish2	1.84	**1.54**	16.30 %
GP13	1.79	**1.53**	14.53 %

5.4 Weighting parameter effect

To determine the best threshold parameterization, which is the same as finding the optimal weighting value of α, another experiment was conducted. Figure 5 shows the average

expense for different weighting values. This indicates that if the weight is too high the performance can be degraded (as opposed to reducing the expense). We speculate that the optimal α value varies according to characteristics of the subject program and the ranking metric. A study to obtain the optimum value of α for each program and metric will be conducted in the future. In this experiment, to generate preliminary experimental results, we typically fixed the weighting value at α = 2.25.

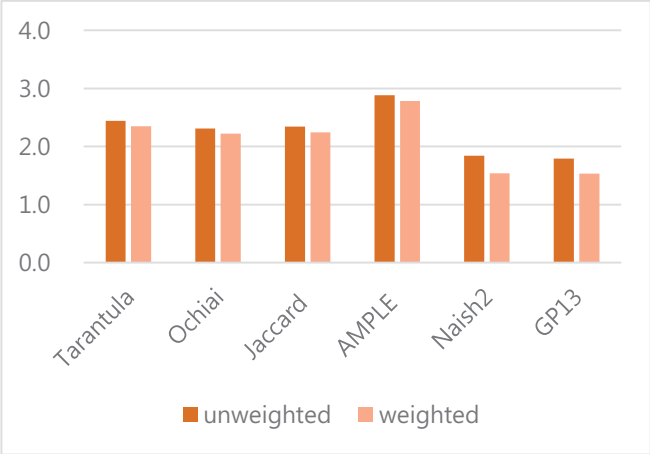


Figure 4 Average expense for each ranking metric

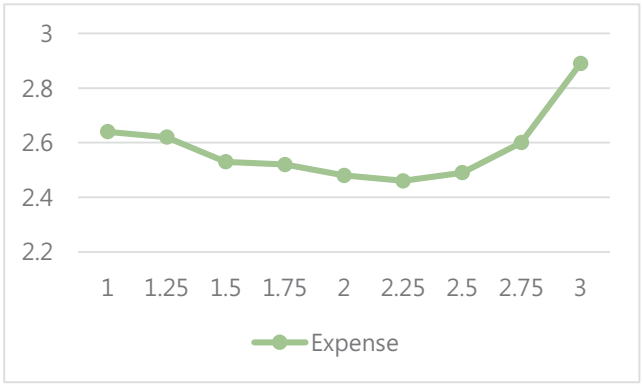


Figure 5 Average expense for different weighting values

6 Threats to Validity

There are a number of potential factors that could threaten the validity of our experiment. Here, we discuss these one by one.

We evaluated only eight small/medium-sized C programs in our experimental evaluation. However, all of these real-world programs are widely used and our technique was evaluated across 159 multiple-fault versions. Hence, we expect the results from our study to be sufficiently representative. Additional experiments with large-scale programs will be conducted in the future.

We verified our experiment with six ranking metrics. We expect other suitable fault localization algorithms to deliver similar results; this will also be studied in our future work.

Finally, as mentioned in Section III, we created multiple-fault versions of programs by randomly selecting a number of available faults. It is possible that these faults could be non-representative of real faults. However, due to a lack of fault data for multiple-fault research, many researchers have manipulated multiple-fault versions of programs by using mutation-based fault injection. We made multiple faults by combining existing available faults; this was done because we believe that it is difficult to minimize problems while artificially generating multiple faulty versions in order to simulate realistic faults by the mutant generating process [19].

7 Conclusions And Future Work

In this paper, we proposed an enhanced technique of fault localization for multiple-fault environments. This was done by using weighting to improve the effectiveness of SFL by incorporating information extracted from failed test cases caused by multiple faults. Additionally, we experimentally evaluated the performance of our technique and compared it to various types of real programs and metrics with multiple-fault versions. The results show that our proposed weighting technique can locate faults more precisely than existing unweighted methods. Furthermore, we investigated the dependency between the performance of the proposed weighting technique with different subject programs and ranking metrics.

In the future, we plan to conduct more empirical studies by using large-scale programs, ranking metrics, and multiple-fault versions. Additionally, as mentioned in Section V, we will investigate how the optimal α value changes according to the characteristics of the subject programs and metrics.

8 Acknowledgments

This research was supported by the Next Generation Information Computing Development Program through the National Research Foundation of Korea (NRF), and is funded by the Ministry of Education, Science and Technology (No. 2015045358) and the MISP (Ministry of Science, ICT & Future Planning), Korea, under the National Program for Excellence in Software that is supervised by the IITP (Institute for Information & Communications Technology Promotion).

9 References

[1] Vessey, Iris. "Expertise in debugging computer programs: A process analysis." International Journal of Man-Machine Studies 23.5 (1985)

[2] Jones, James A., and Mary Jean Harrold. "Empirical evaluation of the tarantula automatic fault-localization

technique." Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering. ACM, 2005.

[3] Abreu, Rui, Peter Zoeteweij, and Arjan JC Van Gemund. "On the accuracy of spectrum-based fault localization." Testing: Academic and Industrial Conference Practice and Research Techniques-MUTATION, 2007. TAICPART-MUTATION 2007. IEEE, 2007

[4] Debroy, Vidroha, and W. Eric Wong. "Insights on fault interference for programs with multiple bugs." Software Reliability Engineering, 2009. 20th International Symposium on. IEEE, 2009

[5] DiGiuseppe, Nicholas, and James A. Jones. "On the influence of multiple faults on coverage-based fault localization." Proceedings of the 2011 international symposium on software testing and analysis. ACM, 2011.

[6] DiGiuseppe, Nicholas, and James Jones. "Fault interaction and its repercussions." Software Maintenance (ICSM), 2011 27th IEEE International Conference on. IEEE, 2011.

[7] Abreu, Rui, Peter Zoeteweij, and Arjan JC Van Gemund. "Spectrum-based multiple fault localization." Automated Software Engineering, 2009. ASE'09. 24th IEEE/ACM International Conference on. IEEE, 2009.

[8] Jones, James A., James F. Bowring, and Mary Jean Harrold. "Debugging in parallel." Proceedings of the 2007 international symposium Fn Software testing and analysis. ACM, 2007

[9] Yu, Zhongxing, Chenggang Bai, and Kai-Yuan Cai. "Does the Failing Test Execute a Single or Multiple Faults? An Approach to Classifying Failing Tests.", ICSE 2015

[10] Hogerle, Wolfgang, Friedrich Steimann, and Marcus Frenkel. "More Debugging in Parallel." Software Reliability Engineering (ISSRE), IEEE 25th International Symposium on. IEEE, 2014.

[11] Jonghee Park, Jeongho Kim, and Eunseok Lee, "Experimental Evaluation of Hybrid Algorithm in Spectrum based Fault Localization", The 2014 International Conference on Software Engineering Research and Practice (SERP 2014), July. 2014

[12] Naish, Lee, Hua Jie Lee, and Kotagiri Ramamohanarao. "Spectral debugging with weights and incremental ranking." 2009 16th Asia-Pacific Software Engineering Conference. IEEE, 2009.4

[13] Lee, Hua Jie, Lee Naish, and Kotagiri Ramamohanarao. "Effective software bug localization using spectral frequency weighting function." Computer Software and Applications Conference (COMPSAC), 2010 IEEE 34th Annual. IEEE, 2010.

[14] Bandyopadhyay, Aritra. "Improving spectrum-based fault localization using proximity-based weighting of test cases." Automated Software Engineering (ASE), 2011 26th IEEE/ACM International Conference on. IEEE, 2011.

[15] Li, Yihan, Chao Liu, and Zi Yuan. "Exploiting Weights of Test Cases to Enhance Fault Localization (S)." SEKE. 2013.

[16] Do, Hyunsook, Sebastian Elbaum, and Gregg Rothermel. "Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact." Empirical Software Engineering 10.4 (2005): 405-435.

[17] Yoo, Shin. "Evolving human competitive spectra-based fault localisation techniques." Search Based Software Engineering. Springer Berlin Heidelberg, 2012. 244-258.

[18] Renieres, Manos, and Steven P. Reiss. "Fault localization with nearest neighbor queries." Automated Software Engineering, 2003. Proceedings. 18th IEEE International Conference on. IEEE, 2003.

[19] Offutt, A. Jefferson, et al. "An experimental determination of sufficient mutant operators." ACM Transactions on Software Engineering and Methodology (TOSEM) 5.2 (1996): 99-118.

[20] Rui Abreu, Peter Zoeteweij and Arjan J.C. van Gemund, An Evaluation of Similarity Coefficients for Software Fault Localization, PRDC 2006

[21] Mike Y. Chen, Emre Kıcıman, Eugene Fratkin, Armando Fox and Eric Brewer, Pinpoint: Problem Determination in Large, Dynamic Internet Services, DSN 2002

[22] Naish, Lee, Hua Jie Lee, and Kotagiri Ramamohanarao. "A model for spectra-based software diagnosis." ACM Transactions on software engineering and methodology (TOSEM) 20.3 (2011): 11.

[23] Xie, Xiaoyuan, et al. "A theoretical analysis of the risk evaluation formulas for spectrum-based fault localization." ACM Transactions on Software Engineering and Methodology (TOSEM) 22.4 (2013): 31.

SESSION

SOFTWARE ARCHITECTURES: INCLUDING HCI, ENTERPRISE SYSTEMS, SERVICE ORIENTED ARCHITECTURES, WEB-BASED APPLICATIONS

Chair(s)

TBA

ISBN: 1-60132-446-4, CSREA Press ©

ISBN: 1-60132-446-4, CSREA Press ©

A systematic approach to evaluate enterprise ontologies using testing techniques of software usability

Alessandro Viola Pizzoleto[1], **and Hilda Carvalho de Oliveira**[2]
[1]Department of Computer Science Federal University of São Carlos - UFSCar São Carlos, SP, Brazil
[2]Dep. of Statistics, Applied Mathematics and Computer Science
Univ Estadual Paulista - UNESP Rio Claro, SP, Brazil
alessandropizzoleto@gmail.com, hildarc@unesp.br

Abstract— *this paper introduces a systematic for evaluating the enterprise ontologies using software usability testing techniques. The objective is to support developers of enterprise ontologies to verify if the ontology that has been developed is easy and simple to use and navigate as well as if it eases the content addressed understanding. Inconsistencies in the content can also found. The systematic adapts software usability testing techniques with the users' participation in an organized and objective way. A case study is presented for the evaluation of an enterprise ontology considering the processes of the software quality model MPS-SW. The results contributed for improvements in the ontology, because it was possible to correct inconsistencies and problems in the organization and relationships of the content.*

Keywords: Enterprise ontology; Usability; Software processes; Ontology evaluation; MR-MPS-SW

1. Introduction

Researches about ontologies have been significant over the years, including many types and their applications. In general, ontologies represent knowledge between different systems and/or components, improving the collaborative use of such content. They are semantic models to capture and represent aspects of a real domain. This work was directed to a type of ontology known as Enterprise Ontology , describing concepts and relationships that exist in an enterprise domain.

An Enterprise Ontology is a concept that is defined for the Enterprise Project [1], from the inclusion of new concepts from project TOVE [1]. This project aims at adapt the methods of ontology modeling for business and change management, providing a common vocabulary in the organization. It facilitates the development of systems that handle the organization's knowledge, promoting integration between tools that manipulate knowledge related to ontology, through the databases sharing created from its ontological structure.

According to Uschold e King [1], in order to build an ontology enterprise there are four stages to follow: (1) identification of the proposal of the ontology to determine the formality level of the ontology description; (2) construction of ontology, capturing and coding knowledge, including integration with other ontologies; (3) ontology evaluation throughout the process; (4) formal documentation (definition of constants, predicates and axioms), reviewing the stages of identification the scope and formalization.

According to Blomqvist [2], an enterprise ontology can be built following five stages: (1) requirements analysis, considering scope and use cases; (2) iterative construction, with middle-out approach to covering the requirements specifications; (3) implementation with appropriate tool; (4) evaluation of the clarity, consistency and usability and (5) maintenance. For the author, the construction of an enterprise ontology can be manual or automatic.

In this context, this paper covers step (3) of [1] and step (4) of [2], proposing a systematic to evaluate the clarity and usability of an ontology with human participation.

For that, the concept of Rubin's evaluation tests [3] was considered, which could be applied to any product development cycle. According to the author, evaluation tests verify that the conceptual models have been implemented properly if the user can develop real tasks, identifying the specific deficiencies of usability. In this type of test, the user navigates between the screens following a specific sequence. The data collected by the evaluator must be analyzed, in order to generate improvements in the system specifications [4].

Thus, Section 2 presents a short literature review about evaluation of ontologies. Section 3 presents the systematic proposed to evaluate enterprise ontologies, composed to six steps for evaluation. A case study is presented in Section 4, considering a business ontology about the levels G and F from the software quality model MPS-SW. The conclusions about using the systematic approach are presented in Section 5.

2. Literatures reviews

In the last decade, many research related to the evaluation of ontologies have been found in the literature. Some authors proposed a systematic of generic evaluation [5] [6] [8], evaluation focusing on reuse [9] and others defined criteria and evaluation metrics [10] [8], standards-based methods

Table 1: Systematic Classification of Evaluation [1]

Level	Approach			
	(1) Golden Standard	(2) Application-based	(3) Data-driven	(4) Humans
(a) Lexical, vocabulary, concepts, data	X	X	X	X
(b) Hierarchy, taxonomy	X	X	X	X
(c) Other semantic relations	X	X	X	X
(d) Context, application		X		X
(e) Syntatic	X			X
(f) Structure, architecture, design				X

have arisen recently [11]. However, the same approach of usability tests proposed in this paper was not found.

Felix, Taofiki e Adetokunbo [12] proposed a spectral analysis of the graph generated by the ontology information. The authors changed the relationship of classes in a tree structure where nodes and edges are used to represent the concepts (classes, objects and entities) and the relationship between them. The structure and the structural dimension demonstrate the complexity and help rapid recovery concepts. This proposal requires an ontologiesâĂŹ documentation, to simplify the tests and which results prove the veracity of knowledge contained in the ontology.

Poveda-VilalÃşn [13] presents the "OOPS!" tool, with the purpose of detecting flaws in ontologies. This tool identifies failures caused by inexperienced developers in building ontologies. "OOPS!" may be used in any domain ontologies implemented in any language. This tool can sort the flaws in three levels of severity: critical, important and less relevant [13]. The tool is useful for validation process of the ontology. However, it analyzes and identifies only predefined faults in a pre-existing configuration. Currently, the tool has 50 preconfigured failures.

Ma et al. [14] propose an approach that examines the ontology in search of inconsistencies by using measurement of axioms and relevant subsets. For each type of inconsistency, a weight for the calculation of the impact value of each type is assigned. Some algorithms for the calculation of the impact value of each inconsistency type are proposed. The aim is to identify and classify the axioms and subassemblies more susceptible to give rise to inconsistencies in the ontology, as well as the concepts that must be modified [15].

Brank, Grobelnik e Mladenić [15] analyzed some types of systematic assessment and defined four evaluation approaches, as well as five levels of expertise about them, as shown in Table I. The differences of the levels adopted by the researchers were treated.

Levels from Table I are explained below, (a) to (d). (a) Lexical, vocabulary, concepts and data: the assessment verify the concepts included in the ontology and the vocabulary used to represent and identify these concepts. Documents related to the problem domain processed with the ontology are used to assist the evaluation; (b) Hierarchy and taxonomy: the relationships between the concepts are validated, especially if the relationship is one-to-many; (c) Other semantic relations: all relations between the concepts that are not "is-a" are evaluated separately; (d) Context, application: whereas an ontology can be part of a set of ontologies or part of a software, the tests should be run on each usage environment; (e) Syntactic level: manually created ontologies are validated, whereas the language used for description of the ontology and its documentation; (f) Structure, architecture, design: predefined criteria are required to analyze the ontology structure. This evaluation is usually made manually.

The approaches to evaluate ontologies presented in Table I are presented below from (1) to (4). (1) Golden standard: compares the syntax used in the ontology definition with the syntax of the language with in which it was written. (2) Application-based: this approach considers that there is an integrated application to ontology. Thus, an ontology can be classified as efficient or not, depending on the part that the application uses. The ontology is evaluated by the results of the application, requesting documentation with the possible results. (3) Data-Driven: compares the information contained in the ontology with existing data (usually a set of textual documents) in ontology domain. (4) Humans: the evaluation is performed by humans, evaluating whether the ontology introduces a predefined set of criteria, standards and requirements related to your domain.

Note that the approach (4) Humans is the only one that can cover all levels of evaluation of an ontology. The systematic proposal in Section III follows this direction, to allow the evaluation of a business ontology at all levels, from (a) to (f). That is because the evaluator has to decide which tests levels to cover and which profiles the participants will have.

3. Systematic proposal

According to Rubin [3], usability corresponds to the quality degree of the interface interaction with users. The quality is linked to the principles: a) ease learning; b) ease of task memorization in case of intermittent use; c) user productivity on tasks execution; d) prevention, targeting the reduction of errors from users; e) user satisfaction. Tests for evaluation of the degree of usability of a product are usually used with the participation of potential users.

Thus, this section presents the systematic proposal for running usability tests with users to evaluate enterprise ontologies. The systematic consists of six steps and presented in subsections from A to F.

Note that the ontology evaluation is generally performed through a software tool such as ProtÃl'gÃl', for instance. In this case, it is important to be clear to the evaluator and the participants that the tool is not being evaluated. If the participant fails to meet the tool used, the evaluator should grant some minutes to present the main tool features

to the end. This time is not considered in the tests. It is important not to lose focus in the definition of the tests: evaluate the structure, navigability, completeness and depth of knowledge contained in the ontology and as the user identifies information.

3.1 Definition of participants' profiles

Efficient tests require that all levels mentioned in Table I are evaluated. Thus, participants with good knowledge about the domain of ontology must be selected. However, it is important that participants with different knowledge levels and experience are also selected. It is recommended to classify the degree of knowledge such as beginner, intermediate, and expert. Roles can be associated with each class of knowledge degree. The amount of profiles and participants in each knowledge degree depends upon the depth you want to achieve with the test run. However, it is not recommended a greater number than fifteen participants nor less than five, for reasons of results and sustainability.

3.2 Preparation of documents to guide the application of usability testing

Seven documents should be made to guide the application of the tests in enterprises ontologies:

(1) Test plan: contains information regarding the purpose of the test, the statement of the problem to be tested, the methodology to guide the test run, the tasks and responsibilities of the evaluator, the metrics of evaluation and preparation of the environment for the participants;

(2) Roadmap of the evaluator: contains information to guide the evaluator or evaluation team, during the tests running:-the objectives; -the environment and the equipment; -the evaluation rules; - ontology functionality; -protocols and procedure; -forms used in the test;

(3) Orientation script for the participant: contains information that must be submitted for each participant before starting the tests: the test objectives and how it will be conducted. The intention is to help the participant on the difficulties encountered in operating the software tool of ontologies used, difficulties in understanding the information contained in the ontology, etc.;

(4) Task list: it is composed by different tasks to be performed by the participants, considering increasing levels of difficulty. A certain degree of difficulty must be associated with each task. The tasks must be defined with the participant's intention to cover all points of enterprise ontology are verified. It is recommended that the tasks be defined based on the ten heuristics of Nielsen for software systems [16]. This is an unusual use form of these heuristics. Nevertheless, they deal with important aspects to be evaluated in a software and that can be evaluated on business ontologies. The task list should be split into two documents: one to be delivered to the participant that has only the tasks; another for the evaluator, with the tasks and information on implementation and results to be obtained, to serve as a guide to the monitoring of the tasks implementation;

(5) Form for data collection: document used by the assessor for explanations relating to the execution of each task by the participant, including their attitudes during the test. This document can be implemented through a text field under each task from the task list; in this case, a copy of the task list with the fields for notes should be used for each participant;

(6) Evaluation questionnaire: contains questions to be answered by each participant at the end of the test. The goal is to evaluate the perception of the participants in relation to the difficulties they found, their expectations and suggestions for improvements in the ontology. Examples of information required from participant: level of ease to find information; difficulties found during navigation by ontology; positive and negative aspects regarding the use of ontology, as to the information contained in it and about the organization of information; ontology development failures;

(7) Terms of image use: document that must be signed by the participant to allow the use of his image, in order to confirm the test. The participant does not need to show his face.

3.3 Contact with participants and implementation of usability testing

Whereas the evaluator defined the participants profiles in step 1 (see subsection A), candidates must be contacted. This is a process that must provide for refusal of some candidates due to the lack of time. However, the evaluator should emphasize that the test should spend approximately one hour only and the schedule can be adjusted.

The evaluator can act in two ways for carrying out the tests: go to the location where is the end or invite the participant to come to a location prepared specifically for the test. The necessary infrastructure must be provided for both. However, in the second situation, the laboratory environment should be pleasing to the end.

It is very important to inform the participant that he is not being evaluated, but the ontology. He must be aware that the ontology must satisfy their needs otherwise it should be adjusted. The participant must also receive a short guidance on how to use the ontology presentation system.

Specific software tools for the execution and usability testing analyze are recommended. These tools allow recording all the participant's actions to performing the tasks. Some tools allow the evaluator to make notes about participant's observations in real time. It is important that the tool support the evaluator during the tabulation process the results.

It is interesting that the evaluator present a possible usage scenario for each task, as well as the participant understands the purpose of that task execution. The intention is that the participant imagines a real situation of your day-to-day life, motivating him to the task.

It should be noted that all participants must perform the same tests and the conduct of the evaluator must be the same in all tests. All supporting documents for the tests should have been adequately provided.

3.4 Analysis of usability testing

After the fourth stage of the tests, the data recorded by the support tool and documented notes must be analyzed. Overall, some examples of facts that can be verified: - Number of times the mouse was used for a participant to complete a task; - Sequence of actions for the task (may be different from that provided by the evaluator sequence); - Participant's behavior through their facial expressions and body during the execution of the task; - Important observations made by participants during the test; - Difficulties found by the participant to complete the task; - Unfinished tasks and the reasons for the participant not finish them.

It is important to consider the time used by each participant to perform each task. This indicator helps the developer to make adjustments in the ontology in order to reduce the time spent and the paths to facilitate obtaining the desired results.

The evaluator should compare the features of ontology that the participant used to complete the task with the resources provided. If the resources used were different, it is appropriate to examine whether the results were also satisfactory for the participant.

When there is an occurrence of several errors in the execution of a task, this can be a sign that the ontology is not self-explanatory, requiring adjustments. Generally, the occurrence of many errors is related to the lack of knowledge about the concepts represented in the ontology - which impairs its use.

Usability testing is also used to verify the level of acceptance of the ontology by the participants. The evaluation questionnaire filled out by participants support this type of evaluation. The observations and evaluator's notes help to identify gaps and necessary improvements that were not observed in other analyzes.

3.5 Preparation of the recommendations

Following the analysis step, it is important to develop a list of the ontology strengths as well as a guide of recommendations for improvement of weaknesses. This guide may contain inferred guidelines between the group of participants and the evaluator, together or separately. These guidelines must be clear and objective to ease the implementation.

The ten Nielsen heuristics can help to organize the recommendations. While this form of heuristics use is not usual, it helps to support the view for user needs to access to the ontology information [16]. All the participants signed the "terms of use".

4. Case study

This section introduces the systematic enforcement proposal in section III, whereas a business ontology on the levels G and F software quality model MPS-SW [17]. The version of the ontology considered for this work was the v. 1.1-not yet tested with users. The systematic application is presented in subsections A to E, where each section corresponds to a step, with the exception of the steps 3 and 4, which are in the same subsection.

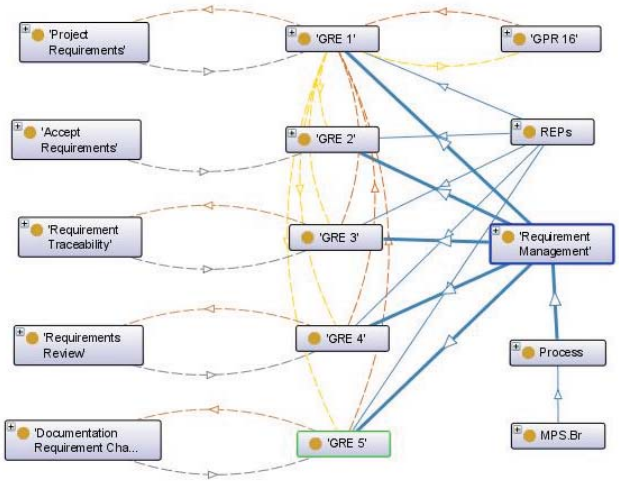


Fig. 1: Part of the ontology in OWL

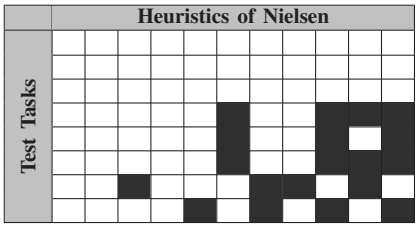
The MPS-SW model has well defined rules for people involved in the process, with the following responsibilities: (1) execution of the process; (2) monitoring of the evaluation process; (3) auditor ship (if the process was executed properly and meet the objectives); (4) validation of the processes (if the case meets the criteria of the company's internal policy).

The MPS-SW model is part of the Brazilian Software process improvement (MPS.BR) based on CMMI (Capability Maturity Model Integration) and ISO/IEC 12207 and 15504, 20000. This model includes internationally recognized practices to the implementation and evaluation of processes associated with the development of software. The idea is to meet the business needs of the software industry. The model defines seven levels of processes: G to A (highest level). G level processes refer to: Project Management and Requirements management. F level processes are: Measurement, Acquisition, Configuration Management e Quality Assurance. Each level adds processes to lower levels. The whole process is composed of attributes, and results of these attributes (RAP) well defined, which must be documented. The effective implementation of the processes must meet the expected results, which are evidenced by a work product or a significant change in the process state. The certification can be granted from the G level, including.

Table 2: Participant's classification

Level	Quantity	Experience
Beginners	3	Computer technicians
		Developers
Project and/or quality managers	4	Project Coordinator
		Project Manager
		Project Leader
		Quality Manager
Implementing and/or evaluator of model	2	Consultant
		Professor

Table 3: Partial view of the relationship between tasks and Heuristics of Nielsen



The proposed ontology by Pizzoleto [17] aims at offering an alternative way to organize the levels G and F of the MPS-SW model. The intention is to support the understanding of several guides for implementation and evaluation model. The ontology adds terms and explanations of the Project Management Body of Knowledge (PMBOK) and indicators according to the perspectives of the Balanced Scorecard (BSC) model among other indicators of processes. It adds also information provided by experts [17]. Figure 1 presents a part of ontology in OWL (Web Ontology Language).

4.1 Participant profile

The definition of the test participantâĂŹs profiles considered different knowledge levels about the MPS-SW model in theory and practice. Knowledge about the implementation and evaluation model was also considered. The Knowledge levels were defined: (1) People with low model knowledge (Beginners); (2) Project and/or quality managers, with some experience in the model; (3) Consultant and/or evaluator of model (high experience in the model). Table 2 shows the definitions for the quantity and functions of the participants as to the experience.

The "Profile Questionnaire" was composed of questions that approached the items proposed on the systematic: educational background, work experience and experience with the MPS-SW model. This information helped in data analysis.

4.2 Documents for the testing application

All seven documents proposed in step 2 were systematically developed. They have been key so that questions during the tests were objectively answered. These documents had the objective to maintain uniformity of treatment for all participants.

The "Task List" was developed with the amount of fourteen tasks, with three difficulty levels: low, medium and high. The sequence of execution was: three tasks with low difficulty, four tasks with medium difficulty level and seven tasks with a high difficulty level. The tasks addressed the following subjects: - Ontology classification; - Composition of expected results (Planning and schedule project, Document requirements); - Information dealt strategic moments of the MPS-SW model (Milestone review, Project estimation). It must be observed that the definition of tasks considered aspects approached in the ten heuristics of Nielsen. Table II presents a table used for this purpose. Notes were also made in the "Form of data collection".

4.3 Contact with participants and test execution

Nine participants accepted the invitation to schedule the test, four beginner participants in the study of MPS-SW model, two participants with the Project Manager role, one as Quality Manager and two as evaluators/consultants of MPS-SW. It is important to notice that the amount planned has been amended in practice according to the participants' availability. Another important element for the invitation acceptance by these people was to inform the estimated time for the tests. Some people could not participate due to schedule and time problems. Next, the defined date was important to remember them by e-mail and phone call.

Two participants were from different cities of the evaluator, distant of about 500 km. However, for all participants, the location selected to do the tests was their own workplace, considering geographical distance problems, traveling time and daily activities of participants. For this, a structure of a mobile laboratory was mounted: two laptops with screens of 15.6" and with the software Morae TechSmith Recorder and Observer (version 3.3.2) in each one. The computers were interconnected through network with crossover connection. Thus, the Morae TechSmith Observer software was used for notes and comments on what the participant was doing.

On the user's computer was installed Protégé system and version 1.1 of ontology, considered alpha version by the author [17]. The concern with the Protégé system interface influence in the tests was mitigated, teaching previously the participant to use the interface before the execution of the tasks (about five minutes for this). It must be observed that Garcia [18] mentions that even users without experience in ontology editors can perform tasks without too much difficulty (no drastic errors).

The tests were conducted for approximately 45 minutes in the presence only of the participant and the evaluator on the place. âĂIJFig. 2âĂİ, illustrates two participants filling the "Profile Questionnaire". All the tests were performed according to the recommendations in a uniform manner and without the need for reviewer interference with the execution

Fig. 2: Participant during the test sessions

of the task. The evaluator made a scenario to the participant before the execution of each task.

4.4 Tests Analysis

All collected data with Morae Observer system during the tests were imported into Morae TechSmith Manager software (Figure 3). This system provides several types of analyzes, including the examples mentioned earlier in the subsection 3.4. All events that occurred during the execution of the tests can be analyzed with the support of graphs generated by the system. During the tests, videos with the movement made on the screen and the participant's picture on the corner of the screen were recorded. Sound recordings/voice were also made. This recorded material can be accessed, allowing insertion of comments.

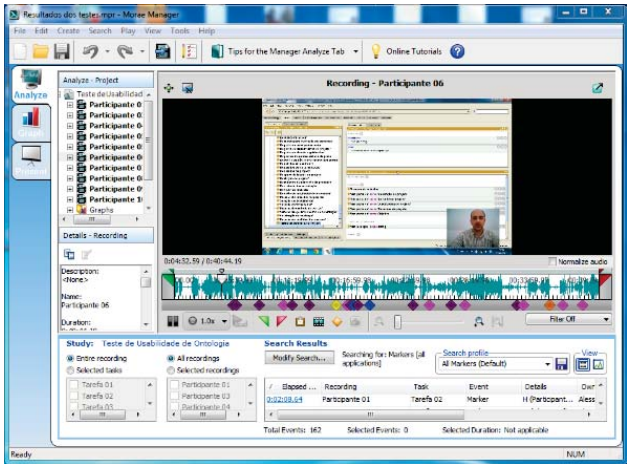


Fig. 3: Morae TechSmith Manager software screen with participant data

All participants were able to complete all fourteen tasks. Some participants asked for help because they were unable to identify some terms used in the ontology. This happened

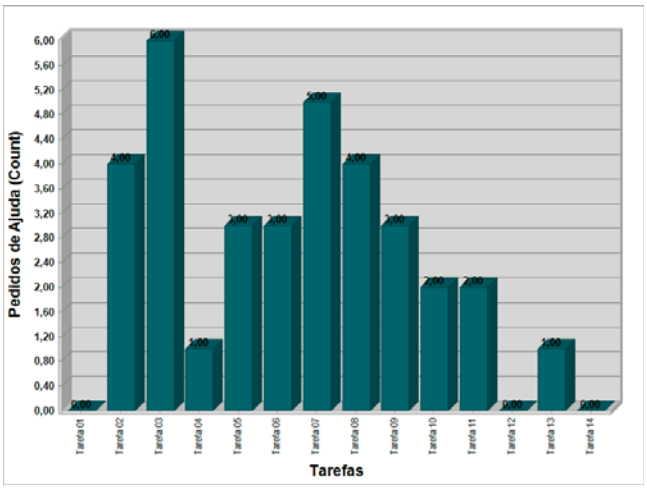


Fig. 4: Average number of requests for assistance for each of the fourteen tasks (x-axis: from 1 to 14)

most often with "beginners" participants due to limited experience with the MPS-SW model. The other participants made a few questions. âĂIJFig. 4âĂİ, shows the average help requests from all participants for each of the fourteen tasks (x-axis: from 1 to 14). Considering graphs as in Figure 4 and the notes taken by the evaluator, it was possible to verify that the great majority of requests for assistance was made when the participant was in a state of navigation and not knew where to go. While the participant was performing the task and went through the tree superclasses and subclasses, the navigation mechanism became clearer as well as the associated information. Thus, requests for assistance have been decreasing with the continuity of tasks.

4.5 Preparation of recommendations

The beta version of the ontology developed was adequate to meet the original purpose. However, some recommendations for changes were presented by the participants and by the evaluator. These recommendations can be summarized in the following requirements: (1) creation of new connection properties between superclasses and subclasses; (2) adjustments on the links between subclasses because chaining failures were identified; (3) identification of parallel flows to facilitate users to access the intended purpose; (4) improvements in the layout of classes; (5) simplification of terms used in the ontology.

All five recommendations are effected with the definition of a set of thirteen tasks presented in Table III. Each task has been performed through a set of activities. Thus, a new version of the ontology was made available for use (v.1.2). This beta version is available in a repository of free ontologies: http://protegewiki.stanford.edu/wiki/Protege_Ontology_Library.

Table 4: Tasks for implementation of the recommendations

Recommendations	Identified tasks
(1) Creation of new connection properties	Identification of new properties.
	Creation of new properties.
	Identifying and building connections between subclasses.
(2) Fault adjustments on the links between subclasses	Identification of failures, analyzing the tests.
	Fix failures.
(3) Identification of parallel flows to facilitate users to access the intended purpose	Identification of subclasses for creating parallel connections.
	Definition of links that would be used.
	Construction of connections between subclasses.
(4) Improvements in the layout of classes	Identification of plugins with better graphics
	Installation and testing of plugins.
	Creating of the installation manual.
(5) Simplification of terms used in the ontology	Identification of the terms highlighted by the test participants.
	Search for alternative nomenclature.
	Replacement of identified and associated information terms.

5. Conclusions

This paper showed a new way of evaluating enterprise ontologies using software usability evaluation techniques with the participation of users. The approach was presented in a systematic way, through six well-defined stages. A case study was presented using a business ontology on the levels G and F of the MPS-SW software quality model. A diagnosis of problems and inconsistencies was generated. The recommendations originated from the analysis and participants comments were implemented, introducing several improvements in the ontology. A beta version of the ontology has been made available in a public repository.

Another important point of the systematic proposed is the human-ontology interaction, because the participant's point of view is analyzed in the test and their satisfaction is measured. Furthermore, the methodology proposed to evaluate business ontologies can be used in other types of ontologies, if appropriate.

References

[1] U. Mike, K. Martin, Towards a methodology for building ontologies. *In Workshop on Basic Ontological Issues in Knowledge Sharing, held in conjunction with IJCAI-95.*, 1995.

[2] B. Eva, *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, ODBASE 2005*, 1nd ed., Ed.. Berlin, Germany: Springer Berlin Heidelberg, 2005.

[3] R. Jeffrey, C. Dana, S. Jared, *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests*, 2nd ed., Ed.. Wiley Publishing, 2008.

[4] F. Reinald, *Padrões Web em Governo Eletrónico*, 1nd ed., Ed.. São Paulo: Ed. Campos, 2010.

[5] A. Gómez-Pérez, *Ontology Evaluation, Handbook on Ontologies*, 1nd ed., Ed.. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004.

[6] G. Aldo, C. Carola, C. Massimiliano, L. Jos, Modelling Ontology Evaluation and Validation. *In 3rd European Semantic Web Conference.*, 2006.

[7] D. Strasunskas, S. L. Tomassen, The role of ontology in enhancing semantic searches: the evoqs framework and its initial. *In International journal of Knowledge and Learning.*, 2008.

[8] A. Duque-Ramos, J. T. FernÃąndez-Breis, R. Stevens, N. Aussenac-Gilles, OQuaRE: A SQuaRE-based Approach for Evaluating the quality of ontologies. *In Journal of Research and Practice in Information Technology.*, 2011.

[9] M. C. SuÃąrez-Figueroa, NeOn Methodology for Building Ontology Networks: Specification, Scheduling and Reuse. *Madrid: Universidad Politecnica de Madrid.*, 2010.

[10] A. Burton-Jones, V. C. Storey, V. Sugumaran, P. Ahluwalia, A semiotic metrics suite for assessing the quality of ontologies. *In Data & Knowledge Engineering.*, 2005.

[11] R. Djedidi, M. A. Aufaure, ONTO-EVO A L an Ontology Evolution Approach Guided by Pattern Modeling and Quality Evaluation. *In Foundations of Information and Knowledge Systems.*, 2010.

[12] A. A. Felix, K. A. Taofiki, S. Adetokunbo, On Algebraic Spectrum of Ontology Evaluation. *In International Journal of Advanced Computer Science and Applications.*, 2011.

[13] M. Poveda-Villalón, . M. C. Suárez-Figueroa, M. Á. García-Delgado, A. Gómez-Pérez, OOPS! (Ontology Pitfall Scanner!): an on-line tool for ontology evaluation. *In International Journal on Semantic Web and Information Systems.*, 2014.

[14] Y. Ma, S. Liu, B. Jin, G. Xu, Inconsistent ontology revision based on ontology constructs. *In SciVerse ScienceDirect Journals.*, 2010.

[15] J. Brank, M. Grobelnik, D. Mladenić, A Survey of Ontology Evaluation Techniques. *In 8th Int. multi-conf. Information Society.*, 2005.

[16] J. Nielsen, *Projetando Websites*, 1nd ed., Ed.. Rio de Janeiro: Campos, 2000.

[17] A. V. Pizzoleto, Ontologia Empresarial do Modelo de Referência MPS para Software (MR-MPS-SW) com foco nos Níveis G e F. *Master's thesis - UNESP.*, 2013.

[18] E. B. Garcia, M. A. Sicilia e S. Sánchez-Alonso, Usability Evaluation Of Ontology Editors. *Knowledge Organization.*, 2005.

Enterprise Architecture of PPDR Organisations

W. Müller

Fraunhofer IOSB Institute of Optronics, System Technologies and Image Exploitation

76131 Karlsruhe, Fraunhoferstraße 1

GERMANY

Abstract - *The growing number of events affecting public safety and security (PS&S) on a regional scale with potential to grow up to large scale cross border disasters puts an increased pressure on organization responsible for PS&S. In order to respond timely and in an adequate manner to such events Public Protection and Disaster Relief (PPDR) organizations need to cooperate, align their procedures and activities, share needed information and be interoperable.*

The paper at hands provides an approach to tackle the above mentioned aspects by defining an Enterprise Architecture (EA) of PPDR organisations and a System Architecture of next generation PPDR communication networks for a variety of applications and services on broadband networks, including the ability of inter-system, inter-agency and cross-border operations.

Keywords: *Enterprise Architecture, Public Protection & Disaster Relief, NAF, OSSAF, System Architecture*

1 Introduction

Public Protection and Disaster Relief (PPDR) organisations are confronted with a growing number of events affecting public safety and security. Some of these events expand from a local to a regional and to an international scale, while others affect from beginning multiple countries. As a consequence, the pressure on PPDR organisations to be able to cooperate in order to respond timely and adequately to such events increases. The need of cooperation demands for aligned procedures and interoperable systems which allows timely information sharing and synchronization of activities. This in turn requires that PPDR organizations come with an Enterprise Architecture on which the respective System Architectures are building. The Open Safety & Security Architecture Framework (OSSAF) provides a framework and approach to coordinate the perspectives of different types of stakeholders within a PS&S organisation. It aims at bridging the silos in the chain of commands and on leveraging interoperability between PPDR organisations. In [1] a methodology was presented, which is based on the Open Safety & Security Architecture Framework (OSSAF) framework [2] and provides the modeling vocabulary for describing a PPDR Enterprise Architecture.

In [3] the process of developing an Enterprise Architecture for PPDR organisations has been described.

The paper at hand presents the results so far of an on-going research being conducted by the research project SALUS[1] (Security And InteroperabiLity in Next Generation PPDR CommUnication InfrastructureS) regarding the PPDR Enterprise Aechitecture and the System Architecture of a next generation communication system for PPDR organisations.

2 Related work

The goal of Enterprise Architecture design is to describe the decomposition of an enterprise into manageable parts, the definition of those parts, and the orchestration of the interactions between those parts. Although standards like TOGAF [5] and Zachman [4] have developed, however, there is no common agreement which architecture layers, which artifact types and which dependencies constitute the essence of enterprise architecture.

[7] defines seven architectural layers and a model for interfacing enterprise architectures with other corporate architectures and models. They provide use cases of mappings of corporate architectures to their enterprise architecture layers for companies from the financial and mining sector.

A layered model is also proposed by [10]. The authors propose four layers to model the Enterprise Architecture: A Strategy Layer, an Organizational Layer, an Application Layer, and a Software Component Layer. For each of the layers a meta-model is provided. The modeling concepts were developed for sales and distribution processes in retail banking.

MEMO [11] is a model for enterprise modeling that is based on an extendable set of special purpose modeling languages, e.g. for describing corporate strategies, business processes, resources or information. The languages are defined in meta-models which in turn are specified through a common meta-metamodel. The focus of MEMO is on the definition of these languages and the needed meta-models for their definition.

The Four-Domain-Architecture [8] divides the enterprise into four domains and tailors an architecture model for each. The four domains are Process domain, Information /

[1] http://www.sec-salus.eu/

Knowledge domain, Infrastructure domain, Organization domain. Typical elements for each domain are also provided. The authors also provide proposals how to populate the cells of the Zachman framework with architectural elements.

The Handbook on Enterprise Architecture [9] provides methods, tools and examples of how to architect an enterprise through considering all life cycle aspects of Enterprise Entities in the light of the Generalized Enterprise Reference Architecture and Methodology (GERAM) framework.

None of the papers addressing Enterprise Architectures covers the special needs of PPDR organizations with their need on timely cooperation, alignment of procedures, and interoperability needs across different organizations.

3 SALUS EA for PPDR organisations

3.1 The SALUS Enterprise Architecture development approach

The SALUS Enterprise Architecture has been designed based on the OSSAF. The OSSAF [2] provides a framework and approaches to coordinate the perspectives of different types of stakeholders within an organisation. It aims at bridging the silos in the chain of commands and on leveraging interoperability between PPDR organisations. One can distinguish the strategic, the operational, the functional, and the technical perspective.

The methodology proposed in [1] for the development of Enterprise Architecture of PPDR organisations, in general and for SALUS specifically, uses NATO Architecture Framework (NAF) [6] as the modeling vocabulary for describing the OSSAF perspectives and views where suitable. The NAF views are modeled with the different elements of the Unified Modeling Language (UML).

The meta-model of the NAF used for the PPDR EA development, together with a description of the core concepts and their relationships has been provided in [Mueller, Reinert]. Also there the tailoring of NAF views for PPDR EA development has been described; especially the strategic and operational perspectives of the OSSAF model (see also Figure1).

Since SALUS is addressing Security and interoperability in next generation PPDR communication infrastructures and not all aspects of PPDR organisations, there is a need for tailoring the Enterprise Architecture development and its artifacts to SALUS use cases. For SALUS only those artifacts of an Enterprise Architecture are relevant which influence the technical development of communication infrastructures. Thus, a funding model of the PPDR organisation (the "PPDR as an Enterprise") or a specific organisation chart of the enterprise or a concrete product configuration used by the PPDR organisation in its daily operations were not in the scope of SALUS. The Enterprise Architecture Components addressed in SALUS are the ones highlighted in Figure 1.

Figure 1: Perspectives and views of an OSSAF-based Enterprise Architecture addressed in the SALUS EA.

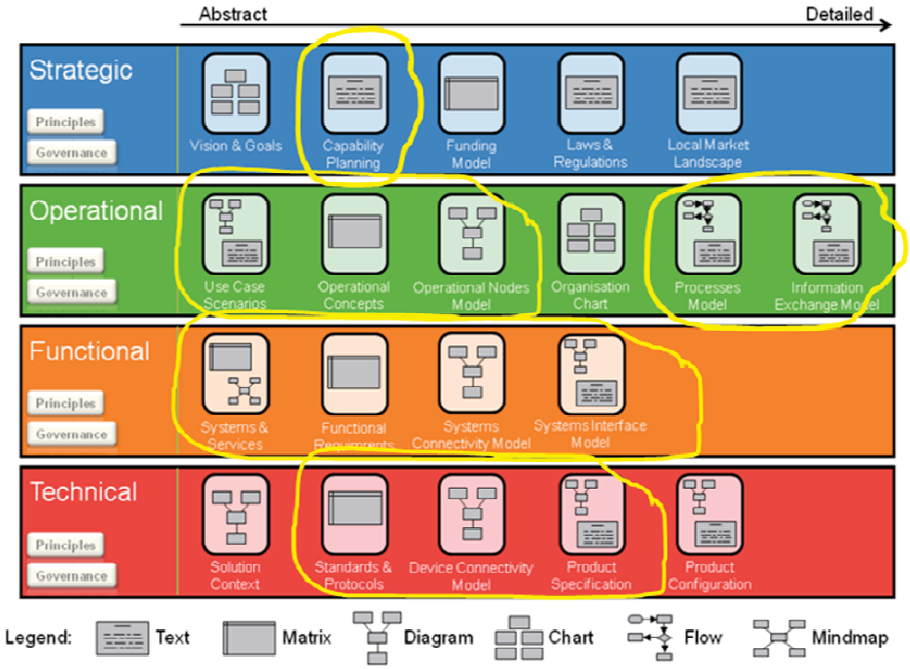
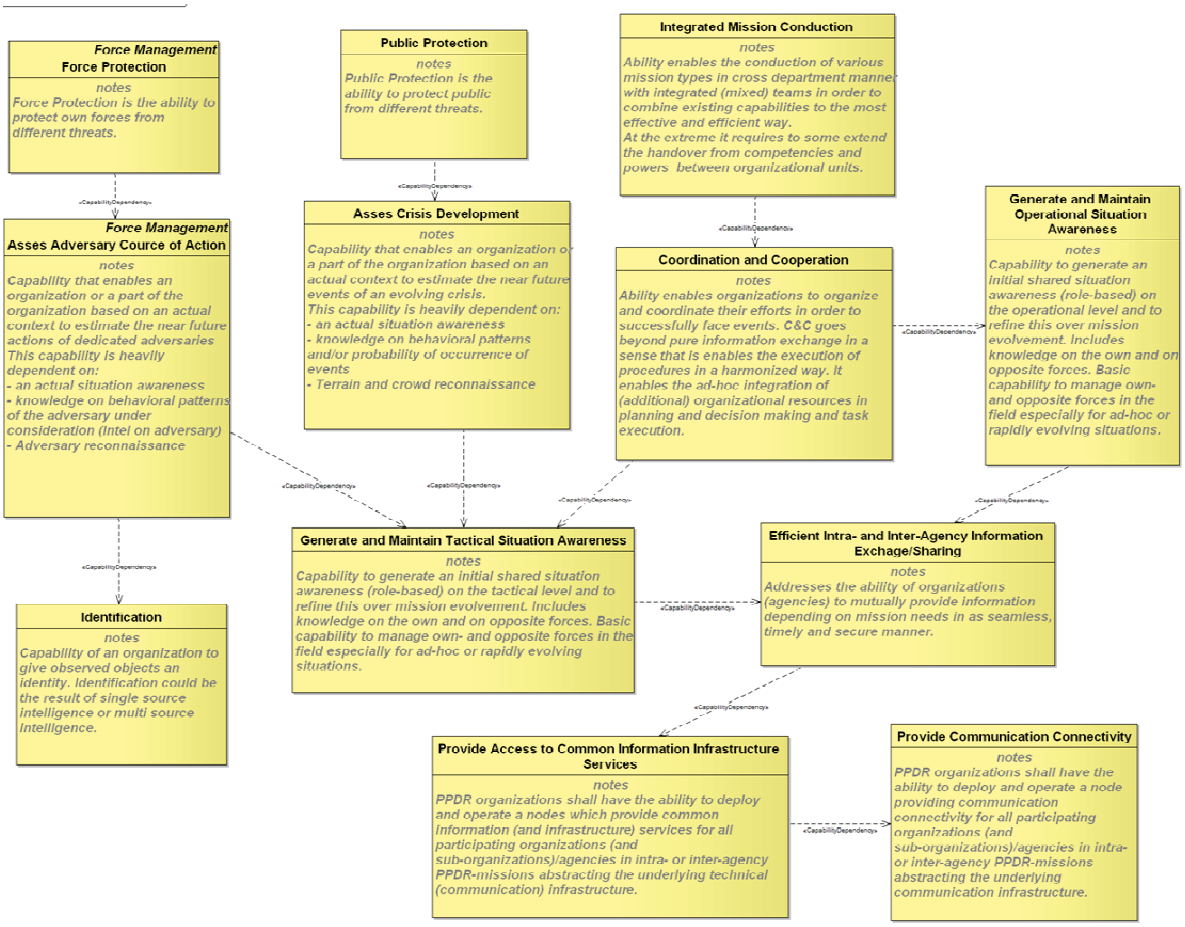


Figure 2: PPDR Capabilities relevant with respect to communication infrastructures



3.2 The SALUS Enterprise Architecture

The methodology proposed in [1] for the development of enterprise architecture of PPDR organizations is using the approach of capability based planning. One can understand a Capability according to [1] as:

”An ability that an organization, person, or system possesses. Capabilities are typically expressed in general and high-level terms and typically require a combination of organization, people, processes, and technology to achieve.”

Using this approach, the following SALUS Enterprise Architecture Capabilities were identified: the capability to protect the public and the citizens - Public Protection; the capability to conduct a mission in an integrated way - Integrated Mission Conduction; and the capability to protect the own forces - Force Protection (see also Figure 2).

The SALUS capabilities also rely on others, like assessing the development of a crisis or the capability to coordinate and cooperate, which in turn depend on the capabilities to generate and maintain the situation awareness. Indeed, for situation awareness capabilities to exchange and share information within an organisation or agency and between organisation and agencies. The exchanging and sharing information capabilities rely on the capability to provide communication connectivity, which is the main capability implemented by the SALUS project.

The capability providing communication connectivity enables the various PPDR operational nodes, like the command and control centers of the different command levels, such as strategic, tactical, and operational on the field; to communicate with each other, exchange information and thus cooperate in order to handle a crisis and to protect the citizens (see Figure 3).

From a functional perspective, SALUS provides a series of services needed for the capabilities to generate and to maintain situation awareness, to provide access to common information infrastructure services and to provide

communication connectivity. These services can be clustered and grouped into the following service taxonomy, as presented in the Table 1.

Table 1. Enterprise Architecture service taxonomy

Service	Taxonomy
Situation Awareness Service	Location and Monitoring Service: – Indoor Location Service – Status Monitoring Service
	Sensor and Tracking Service: – Sensor Data Acquisition Service – Sensor Control Service – Force Tracking Service
Information Assurance Service	Security Service: – Intrusion Detection Service – Resource Authentication Service – Policy Enforcement Service – Forensics Service
Management Service	Management Service: – User Management Service – Group Communication Management Service – Mobility Management Service – Policy Management Service
Network & Information Infrastructure Service	Information and Integration Service: – Message Brokering Service
	Communication Service: – Voice Communication Service, including - Push To Talk (PTT) - Group Call - 1to1 Call - Emergency Call - Ambience Listening – Data Communication Service - Streaming Service - Text Messaging Service
	Interaction Service: – Video Conferencing Service – Chat Service
	Network/Transport Service: – Mobility service - WiFi2LTE Mobility Service - Traffic Management Service – QoS Monitoring Service - Network QoS Monitoring Service – Communication Interworking Service - TETRA2TETRAPOL IW Service - TETRA2LTE IW Service - TETRAPOL2LTE IW Service

Based on the developed Enterprise Architecture, a technical oriented System Architecture was developed.

4 The SALUS system architecture

Nowadays, PPDR organisations are using Private Mobile Radio (PMR) technologies such as TETRA, TETRAPOL or P25 for their communication systems. These technologies do not provide broadband capabilities nor is expected that these technologies will be upgraded in the future. This presents a major limitation in supporting new services and information flows, like those designed in the previous sections. On the other hand, these technologies will continue to exist for at least the next 15 – 20 years due to legal commitments and the huge investments made.

In order to cope with the increasing challenges in day-to-day, planned or unplanned events, PPDR organisations need communication systems and technologies capable of supporting additional capabilities like video and data sharing, within and between PPDR organisations. New technologies, such as Long Term Evolution (LTE) for the long range and Wi-Fi or LTE-U[2] in the short range enable broadband applications and services.

Since narrowband and broadband PPDR systems will coexist, according to the migration roadmap presented in [12], interworking of PMR services between the different wireless access technologies is a major requirement. The PMR services to be supported on all access networks can be split into four main categories: Basic services, PMR supplementary services, telephony supplementary services, and security features. The basic services include the minimum feature set for a conventional PMR network. They consist of registration/de-registration, group affiliation, group calls, one-to-many communications with PTT user request to talk; individual calls, PTT or hook button based; telephony calls to/from an external telephony network, broadcast call, call from a dispatcher to all PPDR users in a group; status, such as predefined set of text messages; and generic text messaging, binary messaging, as transmit sensor information.

The PMR supplementary services are the more advanced services that are essential to PPDR users to operate safely and efficiently. They include priority calls, pre-emptive priority calls, emergency calls, late entry, dynamic regrouping, discreet and ambiance listening from the dispatcher position and location reporting. The telephony supplementary services are services related to public access telephony such as call forwarding features, when busy, or without reply; call hold, call transfer, call barring, incoming and outgoing; and call authorized by dispatcher. The security services are features that are related to the critical use of the wireless communications for PPDR users. They include mutual authentication, the ciphering on the air interface, the end-to-end encryption, the temporary and permanent disabling of a terminal.

The design of the SALUS system architecture takes into account the above mentioned coexistence of narrowband PMR technologies and emerging broadband technologies. It designs interfaces and hand-over mechanisms from Wi-Fi to LTE, TETRA and TETRAPOL to LTE, as well as LTE and Wi-Fi coverage extensions via Mobile Ad hoc NETworks (MANETs) (see Figure 4).

[2] LTE-U (LTE-Unlicensed), operates in unlicensed spectrum, typically in the 5GHz band, to provide additional radio spectrum

Figure 3: PPDR operational nodes and their information exchange need-lines

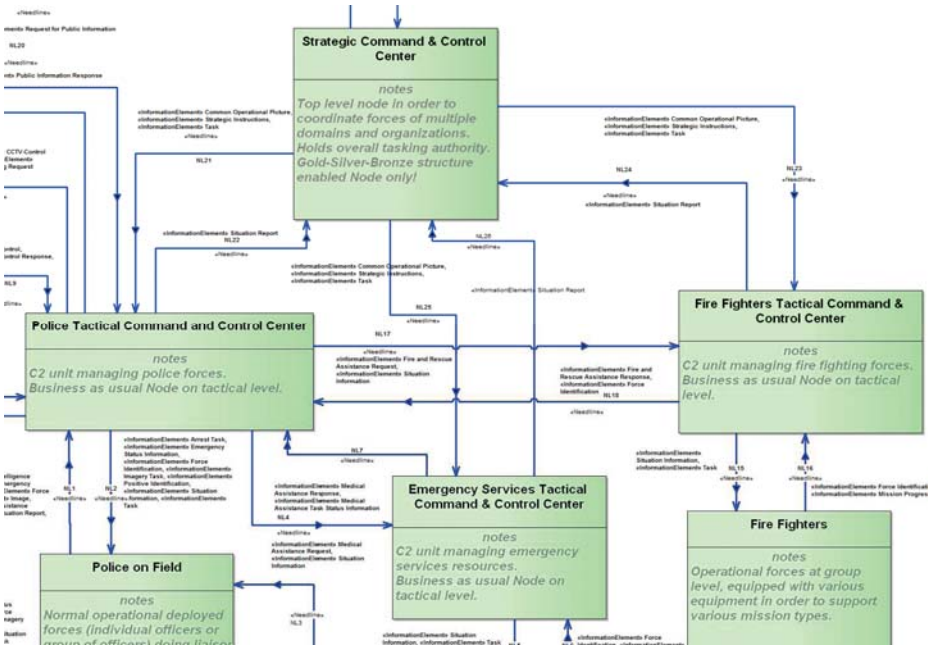
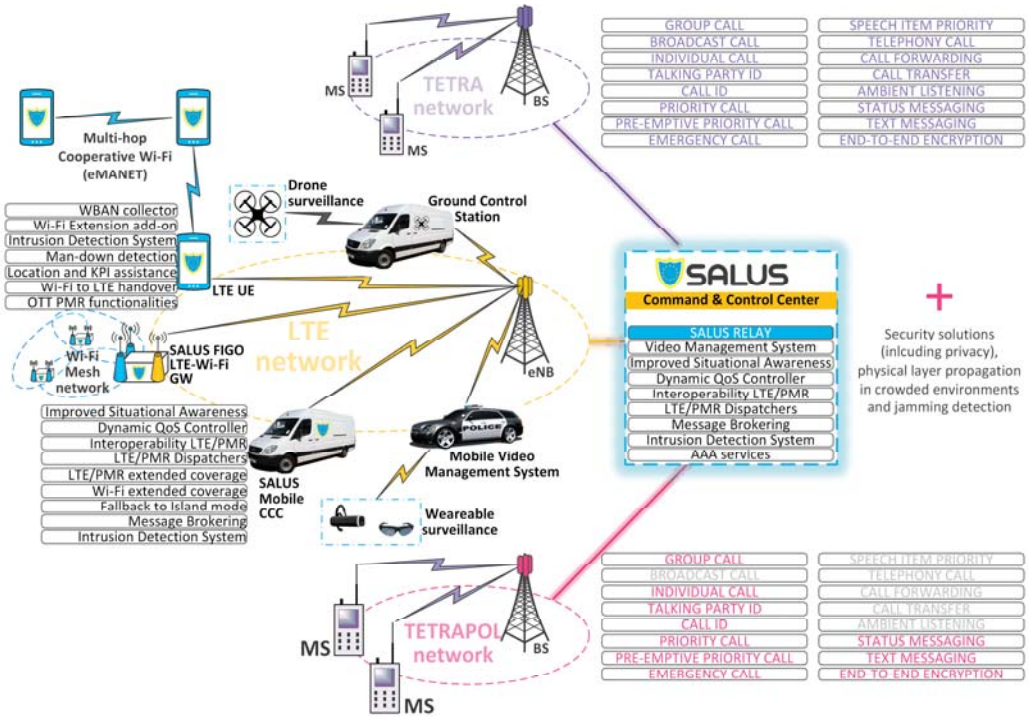


Figure 4: High-level system architecture of the SALUS platform



In accordance with the PPDR Enterprise Architecture, the Command and Control Centre (CCC) is a central piece in the system architecture. It is here where meaningful data is collected, processed and relayed to proper recipients. Besides voice and video support (including group calls), data-driven applications are running in the CCC and the PPDR network. Applications that stream, in real-time, location of PPDR operatives from their hand-held terminals to applications in the CCC, applications that provide terrain characteristics and report measurements to remote geographical information systems (GIS) applications, or applications that by some clever means of processing determine whether a PPDR operative is incapacitated, e.g. is fallen to the ground or report his location are just a few examples. Such applications are by their nature distributed: some run on hand-held terminals of PPDR operatives and some run on computers located at various locations ranging from the CCC to vehicles (like the mobile CCC). For these applications an effective and secure communication means for message exchange is needed. Furthermore, since such applications will often process personal data, the communication means should also respect the privacy of its users.

A key component to respond to these issues is the usage of a Message Broker. It establishes connections between and facilitates the exchange of messages amongst a dynamic number of distributed applications. It achieves these tasks by acting as an application level message router. The Message Broker operates on a well-known network address and accepts connections from various clients. These can be applications running on terminals of PPDR operatives connected to the network via some wireless technology, they can be process-intensive applications running on super-computers in data centres connected to the Internet backbone, or plain situation awareness applications in command and control centres. Once an application connects, the Message Broker enacts the role of an intermediary that facilitates the exchange of messages. The broker provides its own addressing mechanism by which applications address each other, that is, applications do not address each other by their network address but rather by their identities: an identity is an identifier that uniquely determines a single application. Identities allow applications to address each other independently of their network addresses and the translation (or mapping) between identities and the network addresses is something that is an internal matter of the broker. As identities, applications use their public keys. Since it sits between distributed applications, the broker can monitor, authenticate and authorize all message exchanges. To perform these tasks, it has an interface with the AAA services

As depicted in Figure 4, wireless sensor networks and Wireless Body Area Networks (WBAN) are an integral part of the communication system. Sensor data is usually collected by sensors attached to the bodies of in-field deployed PPDR personnel and then sent via their hand-held terminals (UEs) to backend application for processing. The first type of used sensors concern bio-signals, such as the heart rate, blood pressure, temperature and similar kinds of user information. Other kind of personnel wireless sensor information are movement and localization, obtained from accelerometer, gyroscope and GPS sensors. The body signals are used in order to interpret current user health state, location and position allowing an analysis to search for critical events, such as heart attack and falls of users. These data are aggregated and contextually shown on an Improved Situation Awareness application (also known as Common Operational Picture), running on the CCC.

In order to ensure a constant security monitoring of the communication infrastructure, a hybrid network-based intrusion detection system (NIDS) and host-based intrusion detection system (HIDS) approach is used. The approach can either take a signature-based or anomaly-based approach to detect intrusions. By using a hybrid approach, restrictions that are imposed by limited host resources can be overcome, especially when referring to mobile terminals. For example, lack of centralized connectivity may hinder a NIDS, and limited resources may make a sole HIDS approach infeasible. Lastly, the implementation of a hybrid IDS does not create additional infrastructure requirements, since the HIDS part is independent of the NIDS and does not interfere with it

5 Conclusions and further work

An Enterprise Architectures for PPDR organizations with a focus on capabilities of providing access to common information infrastructure services and communication connectivity was presented. The approach is based on the OSSAF and NAF frameworks. It depicts the main capabilities needed by PPDR organisations to perform their mission of ensuring security and safety of the citizens.

The System Architectures and the solution developed within our work and presented in the paper at hand provides the design of a next generation communication infrastructure for PPDR organisations, which fulfills their requirement of secure and seamless end-to-end communication.

The design allows interworking of currently existing narrowband PMR communication infrastructures like TETRA and TETRAPOL with the broadband LTE technology. Furthermore, extensions provided by other wireless technologies, such as Wi-Fi, Bluetooth or ZigBee for WBANs and Wi-Fi or LTE-U for extended coverage have been considered as well.

The design will be evaluated in June 2016 in a live experiment with PPDR users.

Acknowledgement: The work described in this paper was partly funded by the European Commission within the European Seventh Framework Programme under Grant Agreement 313296, SALUS - Security And InteroperabiLity in Next Generation PPDR CommUnication InfrastructureS

6 References

[1] W. Müller, F. Reinert "A Methodology for Development of Enterprise Architecture of PPDR Organisations", Proceedings of the 2014 International Conference on Software Engineering Research & Practice (SERP 2014), pp. 259 – 263.

[2] Open Safety & Security Architecture Framework (OSSAF), http://www.openssaf.org/download

[3] W. Müller, F. Reinert "Development of Enterprise Architecture of PPDR Organisations", Proceedings of the 2015 International Conference on Software Engineering Research & Practice (SERP 2015), pp. 225 – 230

[4] Website Zachman Framework, http://zachman.com/

[5] Website TOGAF, http://www.opengroup.org/togaf/

[6] NATO Architecture Framework Version 3, ANNEX 3 TO AC/322(SC/1-WG/1)N(2007)0004

[7] R. Winter, R. Fischer "Essential Layers, Artifacts, and Dependencies of Enterprise Architecture", Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW'06), IEEE Computer Society, 2006

[8] B. IYER, R. Gottlieb "The Four-Domain-Architecture: An approach to support enterprise architecture design", IBM Systems Journal, Vol 43, No 3, 2004, pp. 587- 597.

[9] P. Bernus, L. Nemes, G. Schmidt (Editors) „Handbook on Enterprise Architecture", Springer, 2003.

[10] Ch. Braun, R. Winter "A Comprehensive Enterprise Architecture Metamodel and Its Implementation Using a Metamodeling Platform", In: Desel, J., Frank, U. (Eds.): Enterprise Modelling and Information Systems Architectures, Proc. of the Workshop in Klagenfurt, GI-Edition Lecture Notes (LNI), Klagenfurt, 24.10.2005, Gesellschaft für Informatik, Bonn, P-75, 2005, pp. 64-79.

[11] U. Frank, "Multi-Perspective Enterprise Modeling (MEMO) - Conceptual Framework and Modeling Languages", Proceedings of the Hawaii International Conference on System Sciences (HICSS-35), 2002, p. 3021ff.

[12] H. Marques et al., "Next-Generation Communication Systems for PPDR: the SALUS Perspective" In: Camara, D., Nikaein, N. (Eds.): "Wireless Public Safety Networks", Volume 1, Elsevier / ISTE Press – Elsevier, London & Oxford, 2015, pp. 49 – 94.

An approach for analysis and flexibility of business rules in legacy systems

Alessandro Viola Pizzoleto[1], **and Antonio Francisco do Prado**[1]
[1]Departamento de Computação, UFSCAR, São Carlos, São Paulo, Brazil

Abstract— *The constant inherent variations in the organizational world, in various domains, require increasingly frequent and varied changes in the rules of business organizations. Consequently, these must be sufficiently flexible so that they can easily adapt to scope change from which they have been designed. This paper presents a service-oriented approach to flexible business rules based on refactoring of source code. Case studies in the field of Education were developed to validate the proposed approach.*

Keywords: SOA; Business Rules; Flexibility; Software Re-engineering; Reverse Engineering

1. Introduction

The inherent challenge in the organizational world increasingly requires agility and innovation so that technological solutions, software, quickly suite the needs imposed by the market and the evolution of its processes. To achieve the goals, this, in turn, should be easy to understand and maintain, especially in its business rules, as they represent the activities of the organization. The software used by organizations today requires great efforts of its responsible to continue operable due to the complexity with which they were designed and developed.

Where legacy software is mentioned, there is the notion that the latter was developed using resources and methods currently outdated and, very often with missing or outdated documentation. Maintaining this type of software can be traumatic, requiring large human effort, and significantly increased the risks to the business, since the source code is complex, not standardized and without comment. Thus, an approach that identifies, classifies, characterizes and refactors business rules contained in it is necessary in order to flexibilize them, hence improving its understanding and reducing the risks in each maintenance, being it evolutionary or corrective.

The flexibility of business rules is not a recent study and has been studied by many researchers in the last decade. A point to note is the main focus of the research conducted and identified to be directly related to business process and not specifically the business rules that are part of the process representing and describing its activities. Some of the solutions are aimed at setting techniques used in the definition of business processes allowing them to be already modeled and implemented with a high level of flexibility [1]. Others are centered in mechanism whose purpose is to measure the degree of flexibility of business processes [2] [3] [4], and there are researches that presuppose the need to define specific languages to make flexible business processes, based on the rules as foundations for the processes to achieve the purpose for which they were created [5] [6].

The remainder of the paper is organized as follows: Section 2 presents the main concepts and technologies used in the proposed approach; Section 3 describes the proposed approach to support the flexibility of business rules in legacy systems; Section 4 evaluates the approach through a case study; Section 5 discusses the related work; and finally, Section 6 concludes.

2. Background

The following subsections introduces the two main concepts to have a better understanding about the proposal: the Business Rules, Business Process Management, Flexibility and Service Oriented Architecture.

2.1 Business Rules

Business rules are abstractions of organizational policy and best practices in an organization, in order to assert their business structure or to control and influence their behavior, as well as to describe operations, definitions and constraints imposed on the organizational business process. In Business Process Management (BPM), these represent the business process activities [7].

When formally specified, the business rules are used in the execution of processes, representing its activities. In software engineering, it is used to support the development of systems that use them and respect their standards [8].

The development based on business rules systems presents some characteristics that facilitate its implementation and maintenance, among them stand out [8]:

- are externalized and shared by multiple systems;
- changes are implemented quickly and at low risk;
- reducing costs arising from changes in logic processes;
- reduction of development time and implementation processes.

Such characteristics significantly increase the flexibility and maintenance of its business rules. Business rules can be represented with a construction IF (condition) THEN, comparing variables with a certain value and then performs

an action when it contains a matching value. Following presents other sentences in order to represent the rules [7]:

- the total value of the order is equal to the sum of the totals of order items plus 10% delivery rate;
- a customer of a bank could not withdraw more than $500.00 per day on his/her account;
- passwords can be at least six characters, including numbers, letters and symbols;
- to rent a car, the customer must present a valid driver's license;
- the maximum number of students per class cannot exceed 30 students.

2.2 Business Process Management

The Business Process Management (BPM) is a concept that unites Business Process Management and Information Technology with a focus on optimization of the organization's results, obtained by improving business processes. To accomplish its use are used methods, techniques and tools to analyze, model, publish, optimize and control processes involving human resources, applications, documents and other information sources [7].

The BPM aims at improving the management of the business environment, providing greater flexibility and autonomy in the operational performance of transactions through it. Therefore, every necessity and business objectives must be met by business processes [8]. According to Kruchten, for a good understanding of business processes should be considered some aspects, such as: goals, business rules and non-functional aspects related to quality, reliability and usability [9]. As a result, organizations from different domains and sizes adopt BPM.

The approach of BPM pursues, to say so, two main purposes: obtaining flexibility in its projects and the formalization of the processes in structured models and diagrams that improve its control and confer greater predictability to organizational activities. Nevertheless, if one considers the needs imposed by the market to the business, it is necessary to consider increasing criticality to adapt business processes to new realities [2].

Business models describes the operation of the business itself, presenting the activities involved, the way they relate and the way we interact with the necessary resources (As Is).

The models exactly describe the business need. In this case, it is necessary to understand the structure and dynamics of the organization, raising the current problems, identify improvements and promote common understanding about the desired situation. These are goals to be achieved.

These targets represent business processes, so that the requirements for its execution are covered. Thus, emerging methodologies for modeling in an integrated approach with existing development methodologies Information System. The modeling among existing stages, is the most visible being a formal instrument of representation of business processes, so that its outcome should be interpreted unambiguously. In addition, the modeling is also a graphical representation of these processes. Thus, it has resulted in a device which utilizes a composite graphics language vocabulary to demonstrate the organization of the workflow.

2.3 Flexibility

The principle of flexibility can be defined as the capacity of equipment, procedures, materials, and components must meet the requirements and circumstances of production and changing needs, without this there are significant variations in the amount of resources needed, and increased time required adequacy and/or use.

Flexibility is then a path to achieve ends such as reliability, cost and speed.

Reliability improves through flexibility, as this helps to deal with unexpected supply disruptions or software implementation. Since costs are reduced with better and full use of equipment, software and resources in general, reducing effort to ensure that the equipment and software start to run catering to new requirements imposed on them [9].

The speed increases with cycle time compression through the elimination of not aggregating activities of value during the preparation of machinery and understanding and easy identification of the functions that will be changed so that the software can perform the changes made or implemented in its activities. It is considered that this generates a major source of competitive advantage, as the company becomes more rapid in its delivery system and the development of new products [9].

2.4 Service Oriented Architecture

The Service Oriented Architecture (SOA) is a design approach that promotes better alignment of information technology with organizational needs. It refers to a style of planning strategy of information technology directly related to the organization's business objectives enabling the translation of the functionality of the applications on standardized services and interrelated [10].

The principle that governs SOA assumes that a large and complex application must be avoided and replaced by a set of small and simple applications, or an application becomes physically composed of several small and specialized modules known as Service, which have features as the ability to be distributed, accessed remotely, interoperable and reusable [11] [12].

The service is defined as the smallest unit of an SOA application representing the completion of a task in a business process upon receiving an input, performs an operation and produces an output [10]. The service is the ability to perform tasks that form a coherent functionality viewpoint of requesters and providers entities. For a service to be performed, there must be used a provider [11].

The advantages found with the use of SOA can be analyzed on two levels: - tactical advantages: focus on software reuse, thus increasing productivity and greater flexibility; - Strategic advantages: focus on the global architecture, which are improved when aligned with the business [13].

3. Proposed Aproach

Sistemas de software jÃą construÃ■dos e implantados requerem manutenÃğÃčo periÃşdica e grande parte dessas manutenÃğÃţes tem origem nas mudanÃğas frequentes das regras de negÃşcio.

Based on presented concepts and technologies, this paper proposes an approach to flexibilize business rules from source-code. The approach aims at providing a practical way for the easing of business rules in the source-code of information systems, identifying duplications and similarities between them. Furthermore, the approach using predefined heuristic identifies the degree of complexity, the need for flexibility and better software engineering concept of being employed to perform the same easing.

The Figure 1, shows an overview of the proposed approach described in Business Process Model and Notation (BPMN). The process consists of three activities: Extract business rules, Refactor business rule and Validate refactoring.

The approach used as input the source-code of information systems. From the source, the Extract Business Rules activity generates a document containing the identified rules and its dependencies. From this document, the Refactor Business Rule activity applies a set of heuristic rules to select the business rule that needs to be refactored becoming more flexible and makes use of software engineering concepts for this. In this activity, if required, adjustments to the database will be realized. The next activity Validate Refactoring applies the concepts verification and software validation to prove that the business rule refactored maintains its functionality and performance. Finally, we have as a result a new source-code.

3.1 Business Rules Extraction

The activity of extracting business rule will use the source-code of the System. An analysis will be conducted to identify and extract the business rules contained in the system code. For this process it is being used the tool case JBrex that meets the needs of this study. The tool is a static analysis of source code, identifying the business rules, its occurrences and generates resources that allow tracing them [14].

The Figure 2, illustrates an example containing the information that is returned by JBrex. This example illustrates the identification of the business rule "ProximoPerfil". In addition to returning the operations for the business rule file also shows the granularity of the functions that make use of the same [14].

The JBrex has a feature that permits to maintain traceability with the source code. Upon analysis of the code and detection of the rules, annotations are included which allows for the traceability between the generated document and the corresponding part of the code. The Figure 3, illustrates how is the identification [14].

3.2 Refactoring Business Rules

Each business rule identified will be analyzed by verifying the need for flexibility. This analysis will be supported by pre-defined heuristics. For this project were defined 17 heuristics. Among the heuristics it can be mentioned that analyzes the complexity of the rules and analyzing the number of parameters defined for the rule.

A list of rules is generated, one-by-one the rules will be reviewed by heuristics in order to determine the best method to use in relaxation. These heuristics have been defined following the standards used in software engineering. For this project were defined 10 heuristics of flexibilization, this number may increase during project execution. Among the heuristics, one can cite that include: parameterization of all fixed values and rule of outsourcing, through the use of SOA, when it is used in many different points of system.

3.3 Validate refactoring

Once the refactoring stage of business rules is completed, tests should be performed in order to verify and validate that the change made in the code is functional and, especially if the features of the rules have not changed.

For this approach were set to be executed such inspection. These tests serves to verify the new code written in refactoring. This type of testing work statically and allows various defects can be found in a single inspection. Not having the need to run the system and test of refactorings that did not undergo this type of testing becomes less costly as it will be applied only in artifacts that have changed [15] [12].

After complete inspection tests and corrected the faults found, the approach provides verification and validation tests must prove that the system has confidence and is ready for use. The verification is to check whether the system meets the functional and non-functional requirements specified, since the validation aims to ensure that the system meets expectations and customer requirements [15] [12].

Even as the aim of easing of business rules, a point that should be tested is the improved performance of the business rule, then tests should be performed comparing the performance of the previous version with the new refactored version of the business rule. To meet this objective performance tests will be performed and the performance in the implementation of the rules will be measured and compared [12].

To prove the improvement in performance tests will be load testing compounds - it will check whether the new routines support a given desired load; estres test - that will determine the maximum capacity of the new routines of the

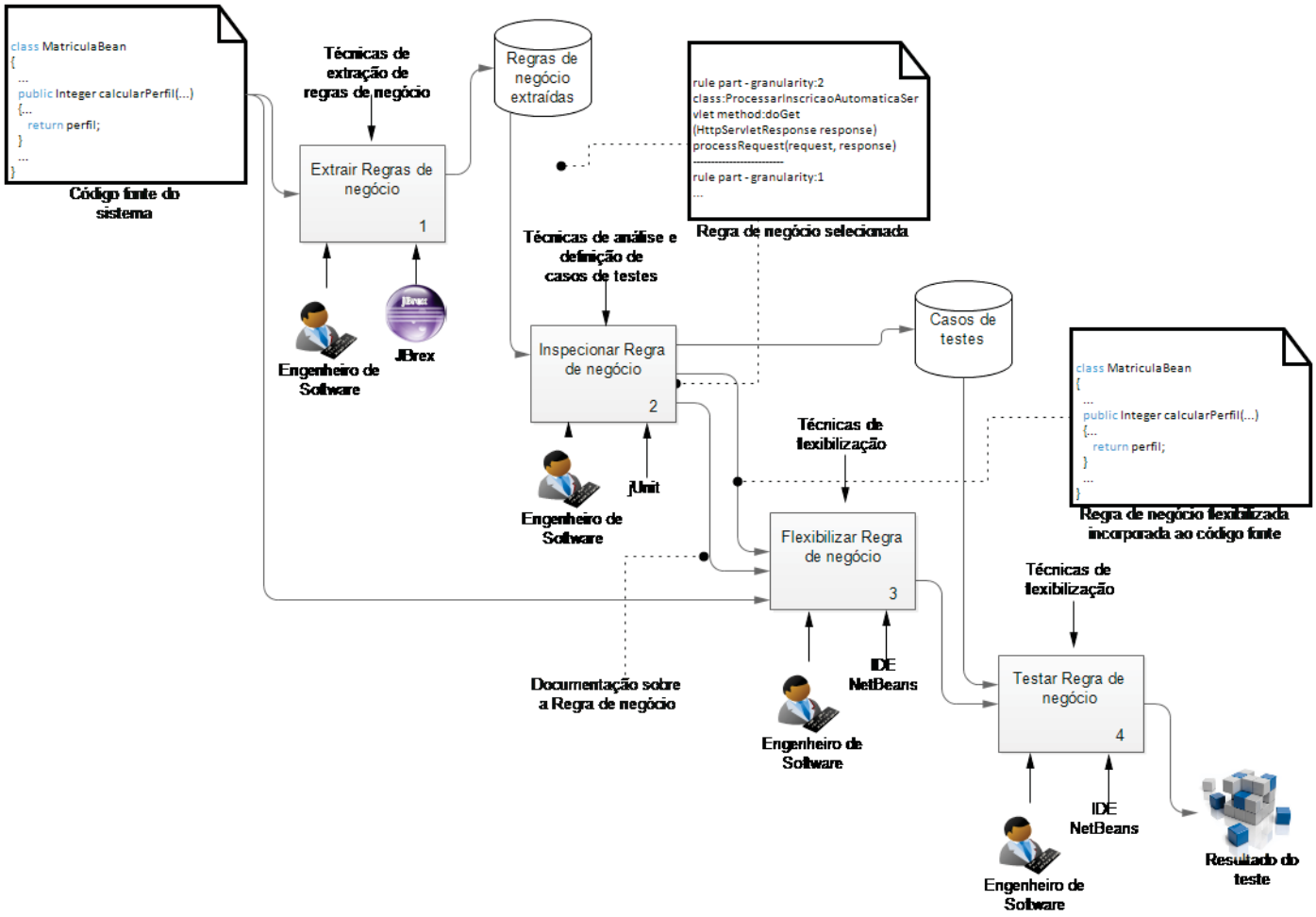


Fig. 1: Overview of the Proposed Approach

```
*********************************
ruleId:1 - variable:proximoPerfil
-------------------------
rule part - granularity:2
class:ProcessarInscricaoAutomaticaServlet
    method:doGet(HttpServletResponse response)
processRequest(request, response)
-------------------------
rule part - granularity:1
class:ProcessarInscricaoAutomaticaServlet
    method:processRequest(HttpServletResponse response)
if (errosValidacao.isEmpty())
inscricao.inscreverAutomaticamente(fIA.getAno(), fIA.getSemestre(),
    fIA.getTipoCurso(), fIA.getTeste())
-------------------------
rule part - granularity:0
class:InscricaoAutomaticaBean
    method:inscreverAutomaticamente(String teste)
proximoPerfil = 1 + calculo.calcularPerfil(ano, semestre,
    numeroDeTrancamentos, numeroDeAfastamentos, matricula)
-------------------------

*********************************
```

Fig. 2: Business rules identified

system, and stability test - which will check if the routines degrade performance over time [15].

```
//ReaMethod:R_proximoPerfil_1_G_1_D_1 | ReaMethod:R_proximoPerfil_1_G_0_D_0
public Integer calcularPerfil(String ano, String semestre,
        Integer numeroDeTrancamentos, Integer numeroDeAfastamentos,
        Matricula matricula) {
    Integer anoParam = Integer.parseInt(ano);
    Integer semParam = Integer.parseInt(semestre);
    Integer anoDeIngresso = Integer.parseInt(matricula.getAnoIngresso());
    Integer semDeIngresso = Integer.parseInt(matricula
            .getSemestreIngresso());
    Integer perfil = (anoParam - anoDeIngresso) * 2
            + (semParam - semDeIngresso) + matricula.getDelta()
            - numeroDeTrancamentos - numeroDeAfastamentos;
    return perfil;
}
```

Fig. 3: Granularity identification and traceability in the source code

For this approach some cases automated tests have been developed, but not in the case study presented in Section 5.

4. Case Stusy

This section presents a case study to evaluate the effectiveness of the proposed approach. The case study was performed in academic domain of Academic Management

364System (ProGradWeb) into the Federal University of So Carlos (UFSCar). The system was developed in Java and consists of 60.9 KLOC, organized into 7 packages and 167 classes.

This study sought to evaluate the efficiency and effectiveness of the implementation of the proposed approach. The extraction of business rules, making use of returned JBrex 95% as a result of these, in amounts from 987 of the 1039. Making use of heuristic analysis, it was concluded that 51 rules need to go through the process refactoring [14].

This was achieved quantity of rules making use of only two of the 17 defined heuristic analysis. The two heuristics used are: a) Duplicate rules - which rules identifies duplicate; b) similarity rules - which identifies the similar rules, namely rules with the same functionality but different codes as.

Among the rules for this study was selected to rules which serves to calculate the "student profile". This rule was found in duplicity in six points of the code (Figure 4) and has three other similarities (Figure 5), totaling nine occurrences.

Continuing the proposed process, it is necessary to use the heuristics of easing. They were defined for this step 15 heuristics to help. In working with this rule were used two: a) Outsourcing - whose goal is to outsource the rule, and may be in a system class or even becoming the rule in a WebService; b) parameterization - which aims to parameterize all fixed values contained in the rule. The new business rule code is shown in Figure 6.

Aided by the tracking code provided by JBrex is possible to identify all modules of source code where the business rules are located and the duplicate or similar code was replaced by the function call as shown in Figure 7 [14].

Completed the necessary adjustments in the source code, validation and verification tests were performed. The purpose of the tests is to identify faults that might have been inserted with the change in the code, check if the functionality of the system has not changed because of the changes and measure the efficiency of the implementation, ie if the changes do not become sluggish System.

In order to prove that refactoring the system was properly executed and demonstrate the gain with the use of the approach, when maintenance is needed, a new test was performed. For its execution had to the aid of four users (developers), two working on the original System Code (1 and 2) and two in the refactored code (3 and 4).

A simple task was requested to replace a parameter in the business rule and the inspection of code running on proving that maintenance works. The results can be seen in Table 1.

Front shown in Table 1, it can be concluded that by making use of the proposed approach we obtain significant results. Comparing using the time to perform the maintenance task and testing the 90% gain in performance, thus allowing organizations to readily suited to the requirements imposed on them.

Table 1: Change, Resources and Testing Time

Users	Implementation		Test	
	Points	Time	Points	Time
1	9	00:31	9	01:03
2	9	00:37	9	01:08
3	1	00:02	1	00:09
4	1	00:02	1	00:07

5. Related Work

As mentioned above, the problem in flexible processes and business rules is not new. For years researchers have been seeking effective and possible solutions to the problem presented in this paper. In the academic literature, several authors have different methods that can be used to derive the flexibility of business processes [16] [4] [17]. Regev developed a taxonomy for relaxation processes in BPMN (Business Process Management Notation) with three orthogonal dimensions [16]:

- Abstraction level of change: this dimension assumes the existence of two levels of processes: - setting level or process model (process type), and level of practical implementation of the model (process instance); and - evaluates how changes to a level implies another level of abstraction;
- subject of change: refers to the different perspectives involved in the process, for example: - Functional perspective: describes what must be accomplished by the process; - Operational perspective: it focuses on activities that are performed in the process;
- properties of change: refers to the specific characteristics of the changes (duration, extent, etc)

Thus, the authors analyze how the characteristics of these mentioned dimensions imply on flexibility degree. Schonenberg evaluates a set of automation process tools, from this analysis, develops an alternative taxonomy that classifies the kind of flexibility based on completeness degree of process definition with respect to its configuration, that is, sets the process in design time (design-time) or at run time (run-time) [18].

Wesker emphasizes the existence of flexibility both in explicit representations in models of business processes as the software tools that support process automation [17].

In all the approaches described flexibility is defined as a technical artifact attribute, which could be achieved mainly through settings or changes the artifact showing a technological view of flexibility. While such approaches are useful for the development of new tools and techniques to support the process, there remains a need for empirical studies on how flexibility can be obtained in practical situations organizations.

The technological approaches are not able to analyze how organizational and practical BPM models can be combined to assist in changing the rules nor the types of resources required to achieve the flexibility.

```
Integer anoParam = Integer.parseInt(ano);
Integer semParam = Integer.parseInt(semestre);
Integer anoDeIngresso = Integer.parseInt(matricula.getAnoIngresso());
Integer semDeIngresso = Integer.parseInt(matricula
        .getSemestreIngresso());
Integer perfil = (anoParam - anoDeIngresso) * 2
        + (semParam - semDeIngresso) + matricula.getDelta()
        - numeroDeTrancamentos - numeroDeAfastamentos;
return perfil;
```

Fig. 4: Duplicate code for the business rule identified

```
if(String.valueOf("345").indexOf(rsetEnfase.getString("codstatus_matr")) > -1)
  iPerfil = 0;
else {
  iPerfil = (Integer.valueOf(sAno).intValue() - Integer.valueOf(sAnoIni).intValue()) * 2 +
        Integer.valueOf(sSem).intValue() - Integer.valueOf(sSemIni).intValue() + 1 +
        rsetEnfase.getInt("delta_matr");
```

Fig. 5: Source-code similar to business rule identified

```
public Integer calcularPerfil(String ano, String semestre, Integer numeroDeTrancamentos,
        Integer numeroDeAfastamentos, Matricula matricula,
        @Default("0") Integer proximoPerfil, @Default("2") Integer fatorAjusteAno) {
   Integer anoParam = Integer.parseInt(ano);
   Integer semParam = Integer.parseInt(semestre);
   Integer anoDeIngresso = Integer.parseInt(matricula.getAnoIngresso());
   Integer semDeIngresso = Integer.parseInt(matricula.getSemestreIngresso());

   Integer perfil = proximoPerfil + ((anoParam - anoDeIngresso) * fatorAjusteAno + (semParam - semDeIngresso) +
           matricula.getDelta() - numeroDeTrancamentos - numeroDeAfastamentos);

   return perfil;
}
```

Fig. 6: Refactored source-code and outsourced

```
MatriculaBean calculo = new MatriculaBean();
proximoPerfil = calculo.calcularPerfil(ano, semestre, numeroDeTrancamentos, numeroDeAfastamentos, matricula, 1, null);

MatriculaBean calculo = new MatriculaBean();
iPerfil = calculo.calcularPerfil(Integer.valueOf(sAno).intValue(), Integer.valueOf(sSem).intValue(),
          1, 0, rsetEnfase.getInt("delta_matr"));
```

Fig. 7: Changing the source-code to use the function

The flexibility of business rules can be classified in some dimensions:

- business rules: increase adaptability towards new lines of business, mergers, acquisitions and opportunities;
- exceptions: easy exception handling, allowing rapid adaptation to new reality of the organization.

For dealing with the flexibility of business rules, there is a need for methods to be used to measure the degree of flexibility. For that reason were extracted and modified from the literature some methods, such as those presented by Kasi, which makes use of three mechanisms: the time, cost, and adaptability [3].

- time: is directly related to the time required to obtain a satisfactory response to the business rule context of change;
- cost: cost is directly linked to the need for implementing the changes and adaptations to be carried out in the business rule;

- adaptability: is directly linked to the ease with which changes are made in the business rule.

According Kasi, by meeting these mechanisms, one can say how flexible is the business rule, since a change can be performed quickly, at low cost and high adaptability. This allows greater efficiency in business rule change of context without the need for additional actions [3].

6. Conclusions

This paper presents an approach to support refactoring business rules making them flexible. The proposed approach makes use of JBrex tool, which extracts by static analysis, business rules system source code. Furthermore, the approach is based on heuristic that uses software engineering concepts to define which rules will be refactored and how will be performed refactoring.

The approach consists of three activities: Extracting Business Rules, refactor Business Rules and Validate Business Rule. Refactored Business rules become more flexible, so spends less time for the organization fits the new internal requirements or imposed by the market, thereby reducing risks and costs in maintaining the system. The approach helps to increase the useful life of information systems and thus helps to improve the ROI.

The feasibility of the approach was confirmed by the case study applied to a real system in the academic domain, showing success by making flexible the business rules.

This review allowed verifying the scalability of the proposed approach to large information systems, i.e., information systems with more than 100 KLOC.

As future work, will be developed a Domain Specific Language (DSL) to describe the heuristics. Thus, the approach becomes refinable and easily maintained, since it would allow the inclusion and/or exclusion of new rules. Furthermore, the case study will be replicated for systems developed to other platforms or languages in order to compare the results.

References

[1] W. v. d. Aalst, A. t. H. B. K. e A. B., Workflow Patterns, *Distributed and Parallel Databases.*, pp. 5-51, 2003.

[2] T. v. Eijndhoven, M. E. Iacob e M. L. Ponisio, Achieving Business Process Flexibility with Business Rules, in *12th International IEEE Enterprise Distributed Object Computing Conference*, pp. 95-104, 2008.

[3] V. Kasi e X. Tang, Design attributes and performance outcomes: A framework for comparing business processes, *Southern Association for Information Systems.*, pp. 226-232, 2008.

[4] A. Schnieders e F. Puhlmann, Variability mechanisms in e-business process families, in *International Conference on Business Information Systems*, 2006.

[5] BRG. (2000). Defining business rules what are they really?. [Online]. Available: http://www.businessrulesgroup.org/first_paper/br01c0.htm

[6] OMG. (2014). Business Process Modeling Notation v1.2 - Final Adopted Specification. [Online]. Available: http://www.omg.org/docs/formal/09-01-03.pdf

[7] OASIS. (2014). OASIS Web Services Business Process Execution Language (WSBPEL) TC. [Online]. Available: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel.

[8] M. Milanovic, D. Gasevic e L. Rocha, Modeling Flexible Business Processes with Business Rule Patterns, in *IEEE International Enterprise Distributed Object Computing Conference*, pp. 65-74, 2011.

[9] J. Becker, K. Bergener, Mueller e F. Mueller-Wienbergen, Documentation of Flexible Business Processes - A Healthcare Case Study, in *AMCIS 2009 Proceedings*, 2009.

[10] M. P. Papazoglou, Cloud blueprints for integrating and managing cloud federations, in *Software Service and Application Engineering*, pp. 102-119, 2012.

[11] M. P. Papazoglou, P. Traverso, S. Dustdar e F. Leymann, Service-Oriented Computing: State of the Art and Research Challenges, in *Computers*, pp. 38-45, 2007.

[12] R. S. Pressman, *Engenharia de Software - Uma Abordagem Profissional*, 7nd ed., Ed. MCGRAW HILL, 2011.

[13] M. P. Papazoglou, P. Traverso, S. Dustdar e F. Leymann, Service-oriented computing: A research roadmap, in *International Journal of Cooperative Information Systems*, pp. 223-255, 2008.

[14] V. Cosentino, J. Cabot, P. Albert, P. Bauquel e J. Perronnet, A model driven reverse engineering framework for extracting business rules out of a Java application, in *Rules on the Web: Research and Applications*, pp. 17-31, 2012.

[15] I. Sommerville, *Engenharia de Software*, 9nd ed., Ed. Rio de Janeiro: Pearson Education, 2011.

[16] G. REGEV, P. SOFFER e R. SCHMIDT, Taxonomy of Flexibility in Business Process, in *International Workshops on Business Process Modeling, Development and Support*, pp. 90-93, 2006.

[17] M. Weske, *Business Process Management: Concepts, Languages, Architectures*, 2nd ed., Ed. Berlin: Springer, 2012.

[18] M. H. SCHONENBERG, R. S. MANS, N. C. RUSSEL, N. A. MULYAR e W. M. P. van der AALST, Towards of Process Flexibility, in *International Conference on Advanced Information Systems Engineering*, pp. 81-84, 2008.

System to locate job opportunities for graduating students

Suhair Amer and Fei Shen
Department of Computer Science,
Southeast Missouri State University, One University plaza, Cape Girardeau, MO, USA 63701
samer@semo.edu

Abstract – *this paper explains the analysis, design, implementation and evaluation of web-based prototype that lists computer science and computer information system job opportunities to graduating students from these degrees. It is not a search engine but a non-traditional website that lists links to all search engines that perform specifically this task. The average time to complete this task is 11.20min. The mode is 11min, and the middle is also 11min. The standard deviation is 2.68. The average score they give for interface appeal is 8.0/10.00.*

Keywords- interaction design, finding a job

1. Introduction

It used to be that when designing a system, a lot of time and code is devoted to the user interface. This had changed with modern windows mangers, interface builders and toolkits. It used to be that the user interface portion of code was between 29% and 88% [Sutton and Sprague 1978]. Artificial intelligence applications it was about 40% to 50% of code [Fox 1986][Mittal et al. 1986]. It is believed that user interface software is more difficult than creating other kind of software because it requires iterative design, need to apply software engineering techniques and sometimes multiprocessing is required to deal with asynchronous events [Myers 1992]. Interaction among users and devices have evolved from one-to-one to many-to-many interaction relationship [Grguric et al. 2016]. Many tools have been created to ease interface programming [Hartson and Hi 1989][Myers 1992]. For example, MacApp tool from Apple can reduce development time by a factor of five [Schmucker 1986]. User-centered design and having effective user engagement is an important component to a successful system [Smith and Dunckley 2016]. This paper explains the analysis, design, implementation and evaluation of a non-traditional web-based prototype that locates, specifically, computer science (CS) and computer information systems (CIS) job opportunities for graduating students. The techniques used to analyze, design and evaluate the system were adopted from [Rogers et al. 2011].

2. Analysis and design

First we identified design goals by asking questions similar to the following:

- Does the interface allow people to search CS/CIS related job opportunities?
- How long does it take for users to search CS/CIS related job opportunities, and can people maintain high productivity by using this interface?
- What happens when an error occurs?
- Does this interface provide all functions needed to search for CS/CIS related job opportunities?
- How long will take the user to learn how to use the interface?
- What kind of support is provided?
- Will user enjoy their experience using the interface?
- Do they feel this interface helpful?

Next, based on a couple of scenarios 2, use cases were developed. In use Case 1:

- Systems displays the options for input job keywords
- Systems displays the options for frequent job search keyword tags for selection
- Users input the job search keywords in the search box
- Users click the "Search Button" for search
- System displays all the results list about related job opportunities
- Users click the bookmark symbol to bookmark the intended job opportunities.

In use Case2:

- System displays the options for CS/CIS job search tags
- Users click the tag to search the CS/CIS job opportunities
- Systems display several checkboxes for several frequently used job search websites
- Users press enter key to find all the websites they choose
- Systems display all the results related to the website they choose, showing the CS/CIS job opportunity search

Next the Volere Shell [Figure 1] shows requirement.

Requirement #: R01
Requirement type: Job search
Event/use case #'s: searching/listing
Description: the systems list up all corresponding search results
Rationale: enter the search keywords and search for corresponding results
Originator: FS
Fit Criterion: testing by users to finding out all related job opportunities related the search requirement.
Customer Satisfaction: Degree of users' happiness if this requirement is implemented successfully. Scale from 1= uninterested to 5 = extremely pleased
Customer Dissatisfaction: Degree of users' unhappiness if this requirement is not the part pf the final product. Scale from 1= hardly matters to 5 = extremely displeased
Priority: rating of customer value
Conflicts: display the results as much as possible
Supporting material: users' needs, requirements and tasks;
History: the first version, in 4/15/15

Figure1: Showing requirements using Volere shell.

With regard to the Mental Model, we understand that when people want to find something, they initially know what they are looking for. They have some basic information about the product, usually a name or some keywords. They usually assume that the platform they will use will have functionalities they want available and it will be simple as just providing the term they are looking for. They also assume that this will be completed fast. This is not always true, results provided may not be all relevant and may take time to list. The erroneous mental model may lead to some restrain on the interactive design and some learnability issues.

With regard to the enhanced conceptual model, two conceptual models were examined. The first one uses a traditional search process where the users input the searched keyword to an information center which finds related information and displays it. The second locates and searches all other websites and displays a summary of the information to the user. In this model, the information center is only a media to connect source databases with the users. When both models are compared, the first is more traditional and is close to users' mental model for a search process. But in perception, memory, and problem solving aspects, the second conceptual model has more advantages because it does not require lots of memory and location to store the data. It is only a media that connect the sources and users. It also provides more possible results.

When designing the interface, several design issues and principles should be applied such as visibility, feedback, constraints, consistency and affordance.

The first design issue is concerned with WIMP and GUI issue. Icons sometimes will become the pervasive feature of the interface. Icons should be very consistent and should bring positive feelings to users especially for functions that are not familiar to users. The second design issue is visualizing information. It is very important to decide on how search results should be displayed which we think will be through a list. The third design issue is related to web design. Some general design principle can be applied, such as visibility, feedback, constraints, consistency and affordance.

Two designs were investigated. The first design uses a traditional search process interface, which has its own database. This will require more implementation time and efforts, requires allocating memory storage and can't provide enormous data results. However, it will be easy to use. The second design is a non-traditional web-based search interface which integrates the results gathered from other job-search websites. At first it will not be easy to understand for first-time users. This will required us to provide help to users. Also, the interface can't directly show all results. Users have to click each individual URLs to check out the results. For this project, the second design was chosen because it was different.

3. Implementation

Figure 2 shows the main interface of the system which lists different URLs that are used to search for CS and CIS job opportunities. The user can choose among three options: main /home page, help page (figure 3), and contact me page (figure 4). HTML and CSS were used to implement the web-based application. The Home page, lists all job search websites together having already the results of the search. The user then selects a link to access each websites' search results (figure 5). The major obstacle in developing such a web-based application

was dealing with style compatibility of the different browsers because each had its own style in displaying their results.

Figure 2: home page of the system.

Figure 3: help page of the system

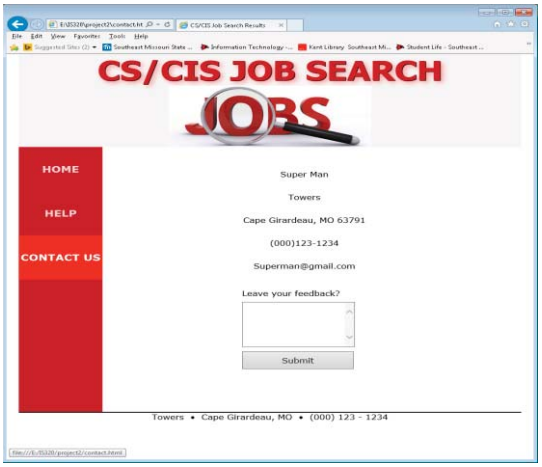


Figure 4: Contact us page of the system

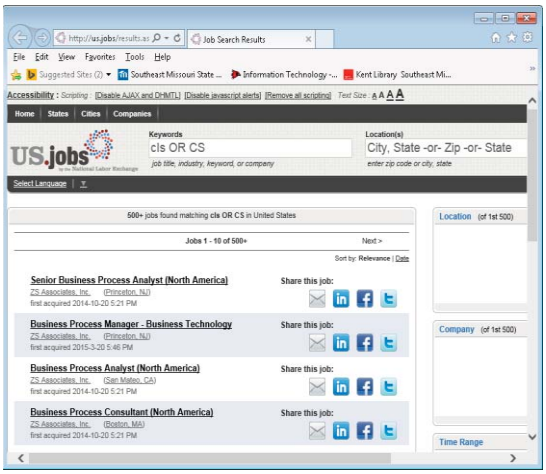


Figure 5: Sample of output page when the user searches for "CS or CIS" job which is placed in the keywords input box.

4. Evaluation and results

Ten test subjects/users were asked to test and evaluate the system. We were evaluating their attitude towards the different pages and how they interact with it. We asked them to provide their opinion and we observed their reactions while using it. We were mainly focusing on identifying any problems with this interaction model. Questions asked: What are the issues? What's the value of this nontraditional interface? can it compete with other job-search websites? Will this interface fulfill the requirement of finding the CS/CIS job opportunities? Will it produce complete results? Is it going to perform the required task fast? Will users trust the results? Are the search engines chosen trust-worthy and acceptable by the users? Will users find it easy to use? How long will it take them to learn to use this interface? Since this interface will be compared against other search engines that perform the same task, it was important to ask the users if they will prefer using ours against the rest? Is the style of this interface acceptable and appealing? Is it distinctive?

First, we evaluated the interface with regard to how long will it take a new user to get familiar with the interface and use it to obtain desired results using a mathematic method analyze. The second evaluation was concerned with the appearance of the website and how users feel about this website. We used a questionnaire. In this questionnaire, there are questions about the general appearance, the basic functions, and the convenience.

The third evaluation asks the users to give feedback regarding their experience using this interface when compared to other types (using open-ended questions).

Several practical issues were identified regarding this evaluation. Subject/users have different backgrounds. They may not want to search only CS/CIS job opportunities. They were informed that the website or model is for searching CS/CIS jobs, but they still wanted to change the keyword unconsciously. So it is very important to make sure they understand what the task is.

Subjects also have different computer and network usage knowledge. As a result, the time spent to complete the same task varies accordingly. They also have different ideas about how a website should be functional and how to be arranged.

To make sure that subjects will not take into consideration the feelings of the developer, the developer decided not to inform them that they are testing her system to remove any bias that may occur.

We also considered the urgency of needing to use such a search. It is different between having to test the system and to really wanting to find a job. This is very hard to resolve at this point as the users may require more functionalities later on when they really need to use the website.

Since we know that each person has his/her own taste regarding style and preference regarding appearance, we prepared very specific questions to make sure that bias will not affect the results

Ten subjects were asked to complete the task of searching for 10 possible CS/CIS job opportunities using our interface. This includes them using the interface for the first time and using help link (if needed). We recorded the time taken to complete this task. Table 1 summarizes the findings. Average time was 11.20 minutes.

Then the subjects were asked to complete questionnaires on their own and no one was proctoring them. Table 2, table 3 and table 4 are summaries of their answers. In general, most of them had positive attitude regarding the functionality of the interface, appearance and style and ease of use.

Table 1: Time taken to complete the search task

User	Min
1	11
2	17
3	12
4	6
5	9
6	10
7	12
8	11
9	11
10	13
Avg.	11.20

Table 2: Summary of the Questionnaire's questions regarding system functionalities

Functionalities:	Disagree	Neutral	Agree	Total
Provides enough search possibilities	1	3	6	10
Provides enough links to other websites	2	4	4	10
Have instructions for how to use it	0	2	8	10
Can leave feedback	0	0	10	10
Have basic contact information	0	0	10	10
Feel helpful about how the functions work	0	3	7	10
Feel good to do the search in this way	0	1	9	10
Total	3	13	54	70

Finally, the subjects were asked to give their opinion regarding the interface and an overall score. Table 5 summarizes the overall score regarding the use of the interface. The average is 8.0/10.

The users also provided feedback regarding the interface. They stated that they liked the design. They liked that they did not have to search many websites to get this information. It was simple to use. They wanted to be able to search other jobs and other locations and allowing narrowing job options. One person pointed out that there should be a way to verify websites.

Table 3: Summary of the Questionnaire's questions regarding appearance and style

Appearance and Style	Disagree	Neutral	Agree	N/A
The website is easy to follow	0	1	9	10
The color scheme makes me feel good	0	0	10	10
The font/size is good to distinct	0	2	8	10
The layout/arrangement in this website is good	0	0	10	10
The figure/image in this website is good	0	1	9	10
This website has a good style	0	2	8	10
Total	0	6	54	60

Table 4: Summary of the Questionnaire's questions regarding Ease of use

Easy to use	Disagree	Neutral	Agree	N/A
I know how to use this website to search job	0	3	7	10
The instruction is easy to follow	0	2	8	10
I think this method of search can be accepted	0	0	10	10
Total	0	5	25	30

Table 5: Summary of the overall score regarding interface use

User	Overall score regarding interface 0..10
1	8
2	8
3	9
4	9
5	8
6	7
7	6
8	9
9	8
10	8
average	8.0

The final step was to evaluate the collected data. Ten subjects were asked to use the system and then provide feedback. All of them were undergraduate students from different majors. 5 of them are males and 5 of them are females. All of them had previous experience using a search engine.

Each subject conducted the search individually and without interruption with us observing their behavior and recording time. We did not provide any help or instructions while they are doing the search.

Their first task was to search for 10 possible jobs and we recorded the time needed to complete the time and noticed that some of them tried to change the search option. The questionnaires were, then, completed by the subjects without our presence. The first one was checking if the interface provides the basic functionalities that we promised the interface would provide. Other questionnaires asked about the visual effects and appearance of the interface and about how easy or how much confusing was it to use the interface.

The open-ended questions which asked the subjects to provide feedback regarding how to improve the interface provided valuable information and suggestions for future work.

5. Conclusion

The system is designed for graduating students at our university looking for CS/CIS jobs. Analysis, design, implementation and evaluation were performed. 10 subjects evaluated the system and provided feedback. The overall satisfaction rate is about 8%, which is corresponded to the overall score they give. The time they used to finish the search task is faster than searching several websites. The users were happy with the interface design and the idea of the project. The average time to complete the search task is 11.20min. The mode is 11min, and the middle is also 11min. The standard deviation is 2.68. So the data varies very little, all data is in the 99.9%. Most of them use the interface during a very reasonable time. The average score they give is 8.0 / 10.0 for design.

6. References

[Fox 1986] Mark Fox. Private Communication. Carnegie Group, Inc. Pittsburgh, PA. 1986.

[Grguric et al. 2016] Grguric, Andrej, et al. "A Survey on User Interaction Mechanisms for Enhanced Living Environments." ICT Innovations 2015. Springer International Publishing, 2016. 131-141.

[Hartson and Hi 1989] H. Rex Hartson and Deborah

Hi. “Human Computer Interface Development: Concepts and Systems for its Management”. Computing Surveys 21, 1 (March 1989), 5-92.

[Mittal et al. 1986] Sanjay Mittal, Clive Dym, and Mahesh Morjaria. “Pride: An Expert System for the Design of Paper Handling Systems.” IEEE Computer 19, 7 (July 1986), 102-114.

[Myers 1992] Brad Myers. State of the Art in User Interface Software Tools. In H. Rex Hartson and Deborah Hix Ed. Advances in Human Computer Interaction, Volume 4, Ablex Publishing, 1992, pp.

[Rogers et al. 2011] Yvonne Rogers, Helen Sharp, and Jenny Preece. *Interaction Design: Beyond Human-computer Interaction.* Chichester, West Sussex, UK.: Wiley, 2011. Print.

[Schmucker 1986] Kurt Schmucker. “MacApp: An Application Framework”. Byte 11,8 (Aug. 1986), 189-193.

[Smith and Dunckley 2016] Smith, Andrew, and Lynne Dunckley. "HUMAN FACTORS IN SOFTWARE DEVELOPMENT-CURRENT PRACTICE RELATING TO USER." Human-Computer Interaction: Interact'95 (2016): 380.

[Sutton and Sprague 1978] Jimmy Sutton and Ralph Sprague, Jr. A Study of Display Generation and Management in Interactive Business Applications. Tech. Rept RJ2392, IBM Research report Nov. 1978.

The Relationship Between AIC and The Quality Product

Lina khalid Ahmed

Software Engineering Researcher, Amman, Jordan
lenaalhafed@gmail.com

Abstract - *one of the main goals of software architecture is to build a software product with high quality and then improve this product according to the quality and the business goals of the product. Architecture influence cycle (AIC) has the impact on building any product with high quality by introducing the factors that influence software architecture and those that are influenced by software architecture then presenting the overall cycle that affects the goal of the product. Architects need to know the basis, the nature as well as the priority of these influences as early in the cycle as possible.*

Keywords: *Software Architecture, Quality Attribute, Business Goal, Architecture Influence Cycle, Architecture Significant Requirements, Architectural Design Decisions.*

1 Introduction

Software architecture is a set of structures that are built for a reason, which comprise software elements, relations, and properties of both. All software systems are built to satisfy the goals of organizations (business goals), so the software architecture is described as a bridge between those business goals and the final resulting system. The good thing is that software architecture is built upon known techniques that ensure achieving these business goals [1].

Business goals and qualities are the basis on which software architecture is built, so the main goal of the architect is to identify the requirements that affect the quality of the business goals which ultimately affect the overall structure of the system. We must classify quality attributes as the system properties, and they are separate from the functionality of the system.

This paper introduces the relationship between the architecture influence cycle and building a product with high quality by introducing factors that affect the architecture as well as factors affected by it. It also describes the meaning of Architecture Significant Requirements (ASR) and their role in this cycle.

This paper is classified as follows: Section 2 describes the related works and includes all the authors who have worked in this area. Section 3 defines AIC with all the details of the factors that influence and others that are influenced by software architecture. Section 4 defines relationships between software architecture, business goals and qualities. Section 5 describes the results which present how AIC affects software architecture in building high quality products. Here, Architectural Significant Requirement (ASR) is defined and its role in the cycle is presented.

2 Related works

Many researchers work on Architectural design and explain how to achieve high quality with a variety of techniques.
Many papers also apply the effects of certain factors on case studies. [2] Presents both practical and theoretical benefits from a case study using Architectural Business Cycle (ABC) to understand how to manage software architecture in automotive manufacturing. It represents the role of ABC in modifying the environment and defining the context of the interviewer. So both the theoretical framework and the Interview methodology that used in this case should be possible to be used for studies at other organizations.
A comparison between software architectures of five industries is done in [3] and from the comparison, authors extracted a general software architecture design approach. Moreover, this paper finds an ideal pattern and from this pattern, the author derives an evaluation that can be used for further method comparisons.
Enterprise Architecture (EA) is one type of software architecture which describes the business structures and it is the most important technology that is being used today in business organizations. [4] Defines and describes the fundamental Business Rule Life Cycle (BRLC) by integrating Enterprise Architecture with Enterprise Decision Management (EDM). It starts by defining the business rule and its qualities then shows its function. Another study is [5], it presents how architectural design decisions affect achieving the goal that the software is built upon. This is done through a design fragment concept and the quality control on these fragments.
This paper presents the relationship between AIC and producing a system with high quality.

3 Architecture Influence Cycle

There are factors that influence building architectures and these factors have an effect on the architect and ultimately on the entire system. This is called an Architecture Influence Cycle [1]. From this definition we conclude that the architecture itself has some factors that influence it as well as other factors that are influenced by.

3.1 Factors influencing software architectures

The architect is influenced by several factors when building the architecture of the system, including:

- **Stakeholders**: a stakeholder is anyone who has a stake in the success of the system. The problem is that each stakeholder has his own goal and concern when building the system. Documenting the requirements and capturing the system's qualities is beneficial for the stakeholder. Here, the role of the architect is to capture all the requirements that do not explicitly appear and referee all the conflicts that frequently appear. Early engagement of stakeholders allows the architect to understand the constraints of the task, manage expectations, negotiate priorities, and make tradeoffs. So the most valuable advice that should be given to the architect is: *know your stakeholder* [1].

- **The development of business environment**: the architecture of the software can be formed by the business/mission concerns such as: time to market, use of legacy system, cost, plans for long-term infrastructure and many other factors that influence the organization, so the architect must know these factors [6].

- **The technical environment:** here, the technical environment means all the technical parts that exist at the time the architecture is to be designed. These technical parts are what determine the new technology to be assembled and added to the infrastructure and that is why this particular factor is always changing. Technical parts include: patterns and styles, social networking, and aspect–oriented programming [6].

- **Professional background and experience of the architecture:** architects always make choices according to their past experience that is why architects must have certain knowledge and skills and this is why an architect's choices might be influenced by experience and training [1, 6].

3.2 Factors influenced by software architectures

Once the architecture is completed and the system is built, both will affect the technical environment, the business goals and social experience.

The architecture also affects the requirements of the stakeholder by giving the customer the chance to receive a more reliable system and certainly with a fewer defects so the requirement must have a way of being negotiated rather than just being an order.

All in all, it must be noted that the architect is affected by the architecture he built. If the architect built the architecture for any system and it worked, he will repeat it again on the future. On the other hand, he will avoid it if it failed. This shows that the architecture will affect the experience and knowledge of the architect and in some cases it will affect the technical environment that is used to improve the system.

That is what is called the cycle of influence. All the factors that affect the architecture when building the system get affected by the architecture when the system is finally built. Figure 1 represents the Architecture Influence Cycle (AIC).

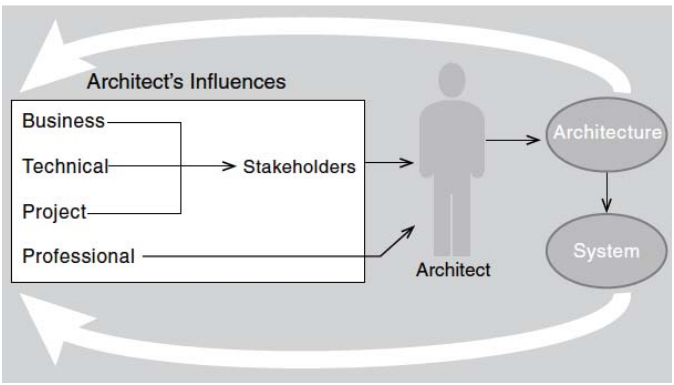


Figure 1. Architecture Influence Cycle as defined by Kazman et al. [1]

To summarize, architecture is more than the technical environment and the functional or non functional requirements. All factors that were described earlier must work as a unit that affects the architecture and the system architect must be aware of that in order to be a competent architect.

The two principles in building any software architecture are [7]:

- The quality attributes that drive the software architecture
- Architecture centric activities, a method of software architectural design. The main goal of it is helping the software development team to build the architecture of the entire system in an iterative way through its life cycle.

4 Software architecture, Business Goal and Requirements

Software architecture or any solution of it cannot be designed well without the understanding of the business goals, qualities and the relationship between them, which makes the relationship a very critical part in designing any system.

4.1 Software architecture and requirements

The requirements for any software system come in different forms, but all these forms encompass the following types [1]:

- Functional requirements: these requirements state what the system should do, and how to react at runtime events.
- Quality attribute requirements: this type of requirements determines the qualification of the functional requirements of the overall system.
- Constraints, which are the architect's design decisions such as the specific programming language that is used through building the system.

According to these types of requirements, the architecture has its response to each type of them as follows:

- Functional requirements are satisfied by assigning responsibilities throughout the design or to specific architectural elements. Assigning responsibilities means building the early design decisions.

- Quality attribute requirements are satisfied by various structures designed to the system, and the behaviors and interactions that populate that structure.

- The constraints are satisfied by accepting the design decisions.

The architectural design is affected by three types of requirements described in figure 2.

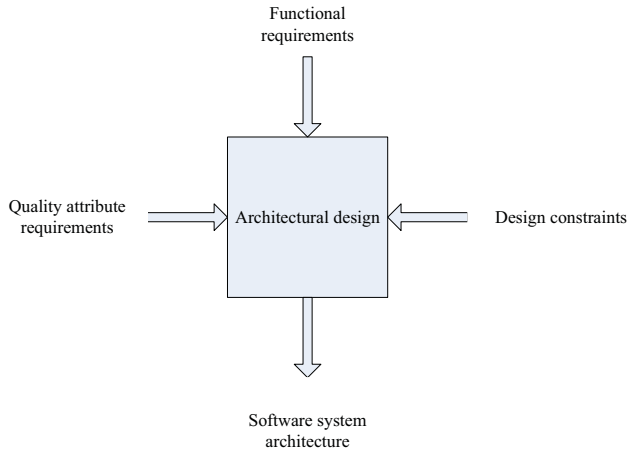


Figure 2. Influences of requirements on Architectural design.

The main point is, structuring the architecture needs these three types of requirements, not just the functionality requirements.

4.2 Business goals and software architecture

Business goals shape the architecture. The architect makes sure that the system architecture has high performance, reliability and security which are suitable to the business goals that are built upon them [8].
The architect needs to know the system's business goals because these goals are what determine the quality attribute requirements of the system and that helps architect to design the architectural decisions of the system. This will be defined later on in this paper. Knowledge of business goals enables the architects to know the tradeoffs and discard the requirements that are useless for the system.
Each organization has its own business goals for the system under development. Ideally, the system will satisfy the business goals and it is the responsibility of the architect to design a system that is able to do that.
Building a design based on business goals is done through a method that is used to generate, prioritize and refine the quality attribute scenarios before the software architecture is completed. This is what is called QAW (Quality Attribute Workshop). The scenarios in this method help us to better describe quality attributes [1].
The stakeholders are connected to the QAW early in the life cycle in order for the stakeholders to discover quality attributes. This method increases the communication between stakeholders, clarifies quality attributes and decides the early design decisions which are applied to the system. Design decision concepts play an important role in designing any software architecture and they are applied on the architecture design processes. For instance, the main decisions are: the main approach for structuring the system, the strategy that is going to be used to control the system, the style that is going to be used to build the software architecture, how to evaluate the architecture, and how it should be documented. [6].

4.3 Business goals and qualities

Business goals are quality attributes the system is expected to achieve such as cost, schedule, and time-to-market.
For example, time-to-market can be achieved by reusing some elements or deploying a subset of the system, while the goal cost can be achieved through having a budget for the development effort which must not be exceeded.
The idea is that different architectures will need different development costs. For instance, an architecture that needs technology that does not exist in the organization will be more costly than one that takes advantage of assets already in house. An architecture that is highly flexible will typically be more costly to build than one that is strict [9].

5 RESULT

The factors that affect the entire system and its quality are decrbied in previous section. This section describes the relationship between AIC and quality attribute (which is the result of this work). Figure 3 concludes this relation; it describes how the architect receives these factors and operates on them to achieve high quality products.

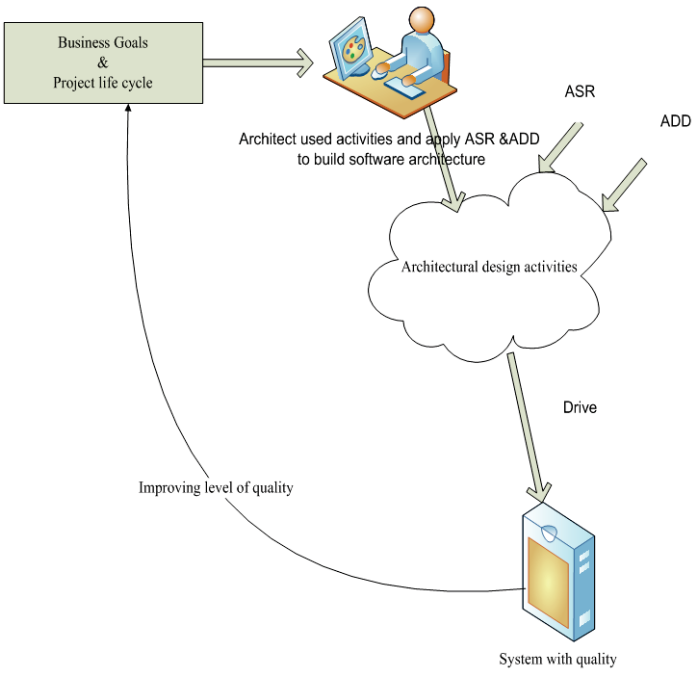


Figure 3. The relationship between AIC and quality attributes

Regardless of the methodology he uses to build the system, the architect should know the basic goals of the system, understand ASR (Architecture Significant Requirement), and choose the important design decisions to put them into the process of building the architecture.

Important definitions that appear in this cycle are:

ASR and Architecture Design Decisions (ADD).

ASR can be defined as a set of requirements for a software system which drive the architectural design; that are why they are significant. This means that these requirements have a deep effect on the architecture of the system. There are many techniques for gathering requirements from the stakeholders; for example, object oriented analysis uses use-case and its scenarios [10, 1]. Furthermore, ASR can be gathered by understanding the business goals because quality attribute requirements can often be derived from business goals

Any technique used to extract ASR should be helpful to record them in one place so that that can be reviewed, referenced, used to justify design decisions, and revisited over time if changes are to be made on the system [1].

ADD can be defined as a set of decisions and considered alternatives that directly influence the design of software architecture. The important thing is that these decisions are achieved through set of reputations which ultimately affect building a system of high quality.

ADD is positioned in the first step of architectural activities, but these decisions are modified through these activities until they are appropriate.

These two important concepts are used by the architect through architectural activities. These activities, which are shown in figure 4, include:

- Architectural analysis

This is the first step of the cycle which defines the problem of the system and models the user's requirements for the system to do.

- Architectural synthesis

This step is the core of the cycle; it processes the ASR and finds new requirements that influence the architecture. The main advantage of this step is moving the requirements from the problem space to the solution space.

- Architecture evaluation

This activity ensures that the architecture is according to the ASR that are proposed and that it certainly achieves the specific goal the system is built upon. This must be the basic part of every development methodology because it has many advantages of which the most important is to validate the functional and quality attribute requirements. Another important advantage is to improve the architecture. One important technique that is used to evaluate the architecture is ATAM (Architecture Tradeoff Analysis Method); it is the best known scenario-based evaluation method. The main purpose of this technique is to assess the consequences of architectural decisions according to the business goals and quality attribute requirements. This paper did not go in details with this technique; first because the evaluation is not the core and second because there are variety of techniques for performing architecture evaluation, and each has a different cost and provides different information. However, according to this paper, ATAM is the most appropriate technique.

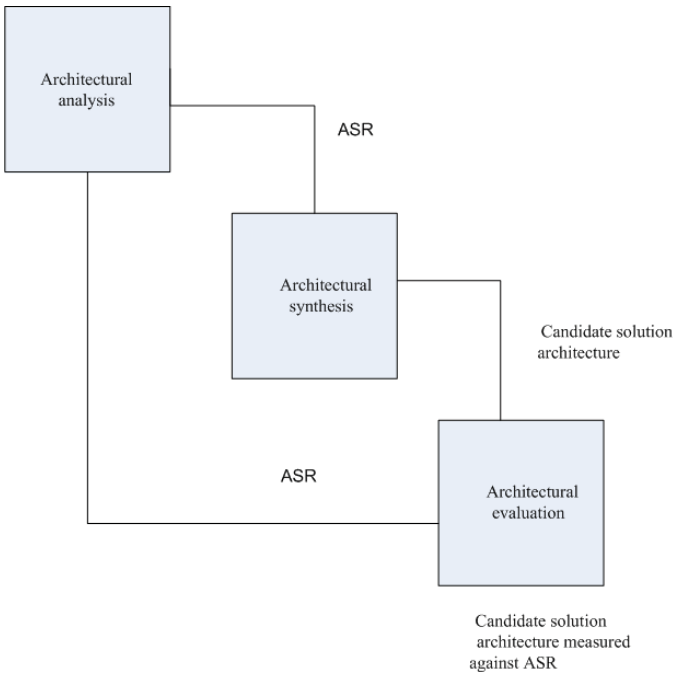


Figure 4. Architectural design activities

Figure 4 shows that the ASRs go through architectural evaluation in which they are evaluated against candidate solution architectures.

After completing the architectural process, the architect can build a system according to its architecture and this system will certainly be a high quality product. If the architect need to develop the capabilities of the system, he can go back to the business goal and the methodology and in some cases he judges the technical environment to have a new system environment.

The relationship between AIC and the quality attributes appears through building the architecture and through improving the capabilities of the system.

6 Conclusion

The idea of this paper is building a software system with high qualities. This work shows that the factors that influence the architect and their relationship with producing systems with high qualities needs to introduced and defined. Moreover, it shows that sometimes we need to go back through cycle to improve the quality and that achieves the goal of the system. That is the reason of changing the environment of technical support or negotiating with stakeholders about the requirements of the system.

7 References

[1] Bass L., Clements P., Kazman R., "Software Architecture in practice", 3rd edition, Addison-Wesley, 2013.

[2] Eklund U,Olsson C," A Case Study of the Architecture Business Cycle for an In-Vehicle Software Architecture,"IEEE,2009.

[3] Christine H., Philippe K. b, Robert L., Nord, Henk O, Alexander R., Pierre A., "A general model of software architecture design derived from five industrial approaches," The Journal of Systems and Software 80, PP: 106–126, 2007.

[4] Tortolero A," Enterprise Architecture and the Business Rules Life Cycle," Innovations Software Technology, 2008.

[5] Khaled L., Achieving Goals through Architectural Design Decisions, and Journal of Computer Science (JCS) .ISSN:1549-3636, PP: 142-1429, 2010.

[6] Software Architecture: Principles and practices. An on line training, SEI, Carnegie Mellon university, 2016.

[7] Nortrop L.,"Architecture Business Cycle Ensuring Product Qualities,"SEI,Carnige Mellon University,2004.

[8] Khaled L., Driving Architectural Design through Business Goals, International Journal of Computer Science and Information Technology.Vol.8. No.3. ISSN: 1947-5500,2010, PP:68-71.

[9] http://etutorials.org/Programming/Softwarearchitecture in practise 2nd edition.

[10] Khaled L., Architectural Design Activities for JAS, International Journal of Computer Science and Information Technology, Vol.6. No.2, ISSN: 1947-5500, 2009, PP: 194-198.

The process of developing a system to find tutors

Suhair Amer and Kolton Benoit
Department of Computer Science,
Southeast Missouri State University, One University plaza, Cape Girardeau, MO, USA 63701
samer@semo.edu

Abstract- *This paper describes the process of developing a system to find tutors. This was a project required in a Human Computer Interaction undergraduate course where analysis, design, implementation and evaluation are required.*

Keywords: HCI, analysis, design, tutor

1. Introduction

The amount of information stored and to be retrieved is massive. Information can be retrieved from the internet using specialized tools known as search engine. This is viewed as a simple software program that searches for the needed information based on keywords supplied by a user. A search can be performed using syntactic or semantic analysis. A large number of Semantic Web search engines have emerged recently which are based on different design principles and provides different levels of support for users and/or applications. There are more advanced techniques used for retrieving information. More information regarding semantic search can be found at [Liu and Zhang 2010] [Miller 2008][Spivack 2009] [Madhu 2011] [Sudeepthi 2012] [Ding et al. 2004] [Radhakrishnan 2007] [Bhagdev et al. 2008] [Chiang et al. 2001] [Bhagwat and Polyzotis 2005] and[Zou 2008]. This paper describes the experience of an undergraduate student who attempted to develop a simple system utilizing human computer interaction concepts.

2. Analysis and Design

The first step was to identify usability and user experience goals such as keeping track of tutors for CS and CIS courses, encouraging contact between tutor and the user (the student), being able to easily find tutors in subject user is searching for, being able to easily navigate the interface, and providing an "I don't know" option to help users find a tutor. The system should be helpful, motivate the user to contact a tutor, and be useable under high stress.

Then usability and user experience questions were formulated. Examples of such questions are:

- Can the system store/keep the information on the tutors?
- Can it search for appropriate tutors depending on what the user needs?
- What if it can't find a suitable tutor?
- What if the use doesn't know what they need help for?
- How easy is it to navigate the interface? Can the user be lost in it too easily?
- How does the system allow the user to contact the tutor(s) it found?

Then user's needs, requirements and main tasks are formulated. The user needs to be able to search for a tutor to help them with their class. Upon finding a potential tutor, the user needs to be presented with multiple methods of contacting them. These methods could be more formal, such as through email or telephone, or less formal, such as various social media outlets. The system should also provide a picture of the tutor to facilitate the old-fashion communication method: face-to-face.
If the system cannot find a suitable tutor, due to a lack of such a tutor or poor search criteria, the system should direct user to a group of default or "trusted" tutors; people who may be able to help, or point them to someone who can help, regardless of what criteria are provided. An "I don't know" option would provide the user with the trusted tutors list. Table 1 relates user intention to the system's responsibility.

Table 1: relating user intention to the system's responsibility

User Intention	**System Responsibility**
Access system.	Welcome the user.
Search for tutors.	Obtain criteria from user.
Find suitable tutors.	Find tutors based on criteria and present them to the user.
Select a single tutor from suitable tutors.	Display tutor profile, including contact info.

4. Scenarios and Use Cases

Based on several scenarios some use cases were created. Next is an example of a use case:

1. The system displays a welcome message to the user, also identifying the system itself.
2. The system provides the user with a search prompt.
3. The users chooses the general subject they need help with and clicks the submit button.
4. If the system finds at least one tutor that matches the criteria:
   a. The system presents the user with the list of tutors.
   b. User selects a tutor, go to 7.
5. If the system cannot find at least one tutor:
   a. Display error message to the user.
   b. Display list of "trusted" tutors.
6. If no criteria (or the "I don't know" option) was selected:
   a. Display error message to the user.
   b. Display list of "trusted" tutors.
7. Display tutor's profile.

The requirements are also presented in a Volere Shell format (tables 2 to 12).

Table 2: Requirement 1 in Volere Shell format

Requirement # 1		Use Case 1, step 1
Description:	Display a welcome message, identify system.	
Rationale:	Welcome the user, make sure they know that they are accessing the system they are looking for.	
User Satisfaction:	3	
User Dissatisfaction:	2	
Priority:	Low	
Dependencies:		

Table 3: Requirement 2 in Volere Shell format

Requirement # 2		Use Case 1, step 2
Description:	Provide the user with a working search feature.	
Rationale:	Allows the user to search for tutors based on the class they have or the subjects that they have selected.	
User Satisfaction:	5	
User Dissatisfaction:	5	
Priority:	Highest	
Dependencies:		

Table 4: Requirement 3 in Volere Shell format

Requirement # 3		Use Case 1, step 4a
Description:	System searches for and displays to user a list of applicable tutors.	
Rationale:	The user generated criteria gathered needs to be used to find an applicable tutor.	
User Satisfaction:	5	
User Dissatisfaction:	5	
Priority:	Highest	
Dependencies:	Requirement 2	

Table 5: Requirement 4 in Volere Shell format

Requirement # 4		Use Case 1, step 5
Description:	Display a list of "trusted" tutors if the system cannot find an applicable tutor with the given criteria.	
Rationale:	Poor criteria may result from a stressed or unsure user; this shouldn't derail their search for a tutor.	
User Satisfaction:	4	
User Dissatisfaction:	4	
Priority:	Mid	
Dependencies:	Requirement 3	

Table 6: Requirement 5 in Volere Shell format

Requirement # 5		Use Case 1, step 6
Description:	A catch-all "I don't know" option, skips to "trusted" tutor list.	
Rationale:	If the stressed user can't provide good criteria, this also shouldn't prevent them from finding someone who can help.	
User Satisfaction:	4	
User Dissatisfaction:	4	
Priority:	Mid	
Dependencies:	Requirement 3	

Table 7: Requirement 6 in Volere Shell format

Requirement # 6		Use Case 1, step 7
Description:	Display the selected tutor's contact profile.	
Rationale:	Show the user the contact information they are searching for.	
User Satisfaction:	5	
User Dissatisfaction:	5	
Priority:	Highest	
Dependencies:	Requirement 3	

Table 8: Requirement 7 in Volere Shell format

Requirement # 7		Use Case 1, step 2, backend function
Description:	Allow instructors/administration to	

	add tutors to the system.	
Rationale:	Tutors added solely through the source code make it difficult to add tutors; it would require changing the source code and recompiling anytime a new tutor was needed.	
User Satisfaction:	3	
User Dissatisfaction:	3	
Priority:	Mid to high	
Dependencies:		

Table 9: Requirement 8 in Volere Shell format

Requirement # 8		Use Case 1, step 7, backend function
Description:	Do not display a tutor if it is after a certain date.	
Rationale:	Tutors are students too: they graduate and leave. The former tutor shouldn't be contacted by new users after a date that is specified when the tutor is added to the system. The function could also be configured to remove the tutor entirely.	
User Satisfaction:	3	
User Dissatisfaction:	3	
Priority:	Low	
Dependencies:	Requirement 3	

Table 10: Requirement 9 in Volere Shell format

Requirement # 9		Use Case 1, step 7, backend function
Description:	Only display a tutor's phone number (if it was provided) during a certain time frame.	
Rationale # 9	Prevents new users from finding a tutor's phone number and calling or text messaging them at an inappropriate time. Cannot affect users who already have the tutor's phone number (obviously).	
User Satisfaction:	3	
User Dissatisfaction:	1	
Priority:	Low	
Dependencies:	Requirement 3	

Table 11: Requirement 10 in Volere Shell format

Requirement # 10		Use Case 1, step 1, additional features
Description:	Show recently viewed tutors on homepage.	
Rationale:	Keep users from needing to re-search for a tutor they have viewed before.	
User Satisfaction:	4	
User Dissatisfaction:	2	
Priority:	Low	
Dependencies:	Requirement 7	

Table 12: Requirement 11 in Volere Shell format

Requirement # 11		Use Case 1, step 4a, backend function
Description:	Sort tutors in a "queue" style.	
Rationale:	Reorder the tutors as users view their profile; prevent a single tutor being at the top of the page, hopefully balancing the load of users across all tutors.	
User Satisfaction:	3	
User Dissatisfaction:	5	
Priority:	Low	
Dependencies:	Requirement 3	

Regarding the conceptual model, the system's main design aspect is to keep it simple, and provide the user with limited options to work with. Since the system's task is to provide a student user with contact information of a tutor, it should be kept fairly simple to accomplish. The method for finding a tutor would be similar to an Advanced Google search: selecting specific criteria to narrow down the results of a query.

Regarding the mental model, ideally, the user's mental model will be almost as simple as the conceptual model. Requiring an unnecessarily complex mental model would put undue stress on a student that is potentially stressed. The user should only need to know how to navigate to the system is and what class they want help in. After arriving to the page, the user should select their criteria to the best of their abilities and choose a tutor from the resulting list. They then can choose how to contact the tutor based on the information that is presented to them.

Upon analysis of the above information, the conceptual model should be tweaked to allow the user a fail-safe way of finding a tutor. A user needing to find a tutor is probably going to be stressed out, not being able to find one in the system would add to that stress.
In addition to the primary method explained in the original conceptual model, a secondary method for finding a tutor would be a list of "trusted" tutors, to act as a "hub" of sorts. This list would be comprised of tutors who are more experienced, thus more likely to be able to help the student or know the criteria

needed to find someone that can help. An "I don't know" option, would take advantage of this "hub." This "hub" could also be used as a last resort, should the system not be able to find a tutor with the specified criteria.

Then we identified potential interface design problems such as the following:

- The system may not be able to scale well:
  - Number of courses supported:
    - More courses mean more options a user would need to search through.
  - Difficulty (100, 200 levels, etc.):
    - The higher difficulty of the courses will mean that we are less likely to have knowledgeable tutors available.
    - Knowledgeable tutors may not be available as the tutors may graduate, transfer, etc.
- Tutor's contact information:
  - A tutor may not be willing to share any contact information other than their University provided email address, due to privacy concerns.
  - A tutor may not *have* any other methods of contacting them, aside from their email address.
  - A tutor may not check their email or other contact methods for new messages.
    - A method of contact built into the system would have the same problem.

Two initial designs were investigated. The first design is a combination of search and results. It does not require a search or an "I don't know" button because the list is to update automatically as the user changes the information in the list boxes. It would be implemented using AJAX in VB (or C#) ASP.NET. The second design is very similar to the first, but splits the search and results sections into 2 pages. This lessens the chance of information overload on the user and makes it simpler to implement. It could be implemented in VB (or C#) ASP.NET or a WCF Windows form VB (or C#) application. Both use a server to host the tutor data. Both designs share the same tutor profile. It lists the name, a picture (if available), contact information, a quote (if available), their CS, CIS, and other technology courses taken and skills that they are known to have. The administration portal page is not shown. Both designs will work, however, the first design does have the risk of the user becoming lost in the mass of information. This is, obviously, a bad situation to put a stressed student in. The second design solves that problem by splitting them up into more than one page, at the cost of slightly more complexity. Due to the development time constraints of this project, the second design using ASP.NET would be faster to implement into a working system chosen.

3. Implementation

The system was developed using Visual Basic. Figure 1 to 7 show different forms that are used by the system and output produced when testing the system.

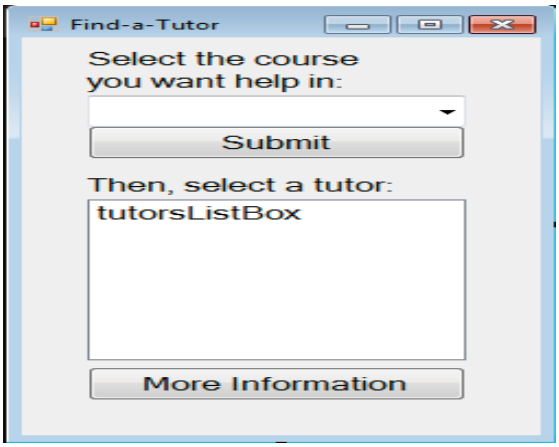


Figure 1: Main form

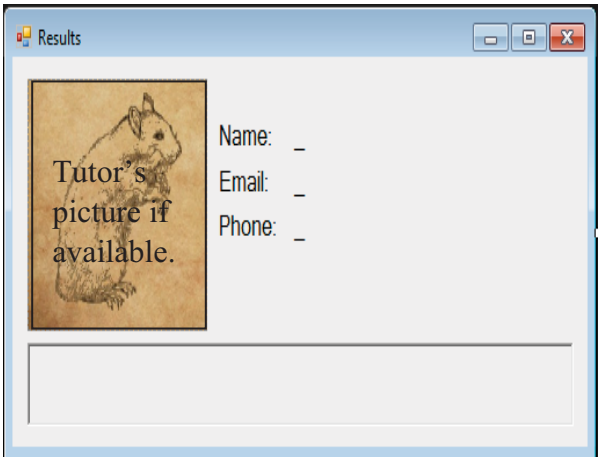


Figure 2: More information form

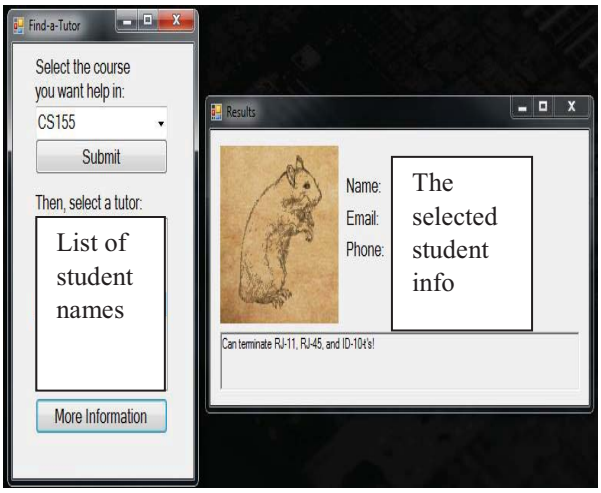


Figure 3: output of Test 1: Search for CS155 tutors.

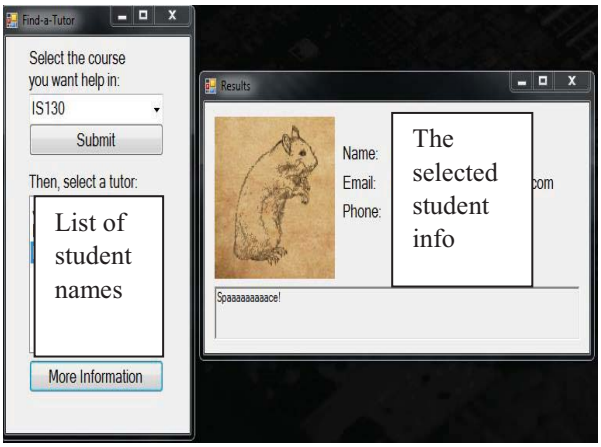


Figure 4: output of Test 2: Search for IS130 tutors.

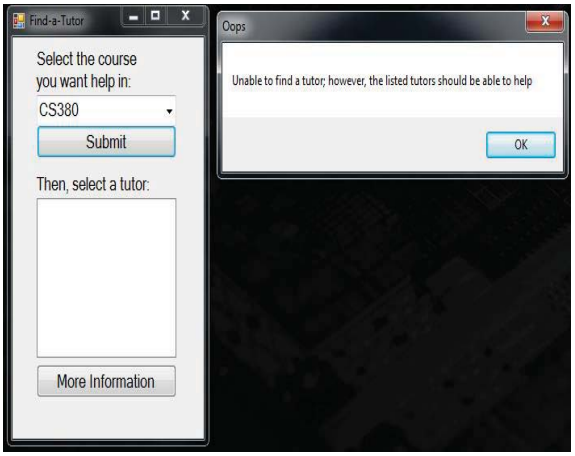


Figure 5: output of Test 3: Search for CS 380 tutors.

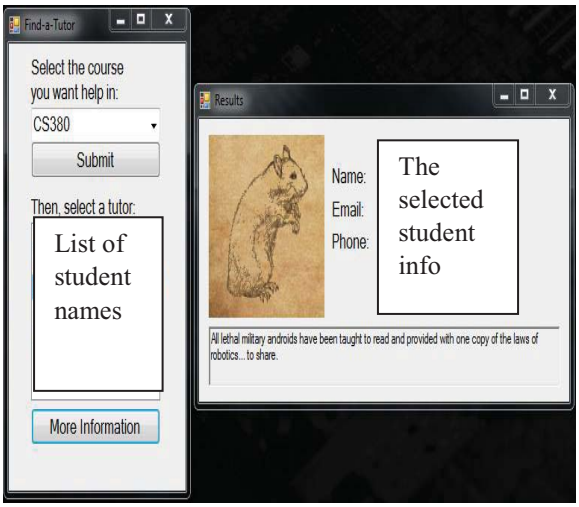


Figure 6: output of Test 3 continued

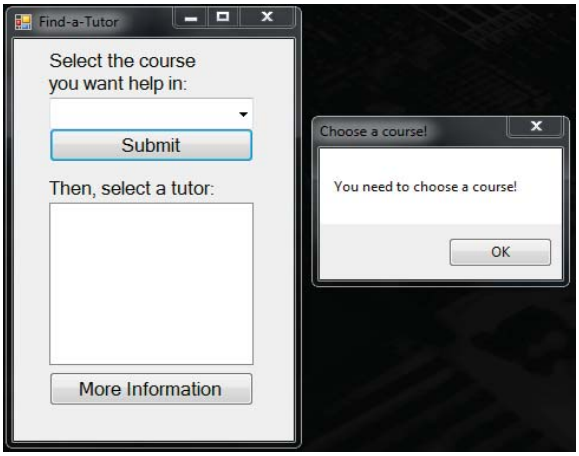


Figure 7: output of Test 4: Error handling; no course selected.

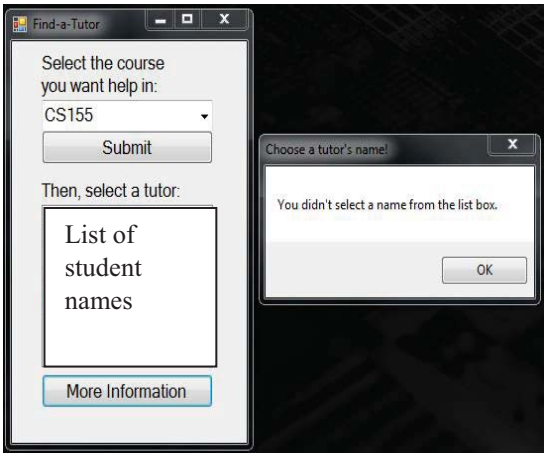


Figure 8: output of Test 5: Error handling; no tutor selected.

4. Evaluation

The goal of this evaluation is to test the effectiveness of the developed program The program's main task was to provide a way for students to find tutor's for their course. The goal of the evaluation is to have at least 90% of test participants successfully use the program and find a tutor.
Some of the explored questions are:

- Does the user find the interface simple? Complex?
- Does the user seem to get lost?
- Does the user find a tutor?
- Does the user find the information provided useful?
- If this system (or a similar system) were fully implemented, would they use it? (Or have used it in lower level courses?)
- If this system (or a similar system) were fully implemented, would they be interested in being one of the available tutors?
- Are there any complaints about the system?

The evaluation method involved going to the Lab and asking random students there to use the program and ask their opinions on it. This field test presents a situation that many students may find themselves in: they have access to a computer, the software they need for their courses, are surrounded by other students, and have network connectivity to use the program. The test participants would anonymously search for a predetermined course and choose a tutor from the results. They would need to use the program in a corner of the room with as few other participants as possible; so as to prevent the next participants from having an unfair advantage on how to use the program.

Despite the evaluation method being simple, it still presents a few practical issues. The first problem is, of course, time. There was (officially) a total of 4 weeks for the project, 1 week was for planning and design, 2 weeks for implementation, and finally the last week is for evaluation. Another issue was that participants are proficient enough that they can either find help on their own or are at a certain course level were there would be few to no tutors available. Those same participants, being students with majors focusing on programming, are also likely to try to break the program. Bug testing is important, but not when the concepts behind the system need to be tested!

The next step was to collect data. The participants were asked to use the prototype program to search for a tutor in a defined course. Any problems observed by the handler or reported by the user were noted. Participants were also asked about their thoughts on the program (what did they like, dislike, etc.) and if they would have used such as system (either in the future or the past, whichever is more applicable). 8 subjects provided the following information:

Problems encountered while testing the system included comments such as that they did not use the drop-down list correctly, they tried typing the course data into the box, they attempted to use the "More Info" button before clicking the submit button, some had no problems, but found the warning for "no tutors found" too vague.

When they were asked about "What do you like / dislike" answers included comments such as "The program is really quick to access." "I like how simple it is.", "It is simple." "The window is small; it doesn't take up the whole screen.", "That warning does not make sense. It doesn't find any tutors, but then shows tutors?", "I don't like having to click more than once to access the data." "It'd be nice if the [tutor contact info] screen would show the other courses they have knowledge of.", "I like the simplicity." "I wish the list auto updated.", "I think it'd be better if it showed the tutor information in the same window.", "The quotes in the tutor information seem useless."

When they were asked about "Would you use the system" answers included comments such as "I would probably use it and wouldn't mind if I were a tutor that could be found using it.", "I would use it.", "I probably would have been a tutor 2, 3 years ago. But, I wouldn't have use it to look for tutors.", "I don't know.", "Yes."

The validity of the evaluation is difficult to determine; the evaluation was to let people use the program to find any problems with it, this openness allows for anything to happen. Unfortunately, this instance of this evaluation was biased by the test subjects themselves: they were all mid to upper level CS/CIS students. They were experienced enough to know how to use the various interface elements, where a low level student may not. The tests show the program mostly works as designed. It does show that there are few problems with the interface, but the concept behind it works well enough. The test participants were not recorded with an electronic device in any way. They were also not asked to give their name or any other information in regards to them. They were not asked to sign a release.

5. Conclusion

For the student developing this system, the implementation was harder than he originally planned for within the time limit. He decided to not develop it as web-based, but as a Visual Basic Windows Form desktop application. When testing the prototype all of the test subjects managed to find a tutor. The tests show the program mostly works as designed. Subjects' comments do show that there are a few problems with the interface that would improve performance.

6. References

[Bhagdev et al. 2008]Ravish Bhagdev, Sam Chapman, Fabio Ciravegna, Vitaveska Lanfranchi, and Daniela Petrelli. Hybrid search: effectively combining keywords and semantic searches. In Proceedings of the 5th European semantic web conference on The semantic web: research and applications, ESWC'08, pages 554–568, Berlin, Heidelberg, 2008. Springer-Verlag.

[Bhagwat and Polyzotis 2005] Deepavali Bhagwat and Neoklis Polyzotis. Searching a file system using inferred semantic links. In Proceedings of the sixteenth ACM conference on Hypertext and hypermedia, HYPERTEXT '05, pages 85–87, NY, USA, 2005. ACM.

[Chiang et al. 2001] Roger H. L. Chiang, Cecil Eng Huang Chua, and Veda C. Storey. A smart web query method for semantic retrieval of web data. Data Knowl. Eng. , 38(1):63–84, July 2001.

[Ding et al. 2004] Li Ding, Tim Finin. Anupam Joshi, Yun Peng, R. Scott Cost, Joel Sachs, Rong Pan, Pavan Reddivari, Vishal Doshi "Swoogle: A Semantic Web Search and Metadata Engine". CIKM '04

[liu and Zhang 2010] Zhusong Liu and Yuqin Zhang. Research and design of e-commerce semantic search. In Information Management, Innovation Management and Industrial Engineering (ICIII), 2010 International Conference on, volume 4, pages 332–334, 2010.

[Madhu 2011] G. Madhu, Dr.A. Govardhan, Dr.T.V. Rajinikanth "Intelligent Semantic Web Search Engines: A Brief Survey" International journal of Web & Semantic Technology (IJWesT) Vol. 2, No. 1, January 2011

[Miller 2008] Paul Miller. Powerset shows semantic search solution. http://www. zdnet. com/blog/semantic-web/ powerset-shows-semantic-search-solution/141, May 2008.

[Radhakrishnan 2007] Arun Radhakrishnan, "Semantic Search Lexxe : Search Engine that Answers Exact Queries", Search Engine Journal July 20, 2007.

[Spivack 2009] Nova Spivack. The road to semantic search the twine. com story. http://www. novaspivack. com/uncategorized/ the-road-to-semantic-search-the-twine-com-story, December 2009.

[Sudeepthi 2012] G. Sudeepthi, G. Anuradha, Prof.M. Surendra Prasad Babu, "A Survey on Semantic Web Search Engine", IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 2, No 1, March 2012

[Zou 2008] Guobing Zou, Bofeng Zhang, Yanglan Gan, and Jianwen Zhang. An ontology-based methodology for semantic expansion search. In Fuzzy Systems and Knowledge Discovery, 2008. FSKD '08. Fifth International Conference on, volume 5, pages 453–457, 2008.

ISBN: 1-60132-446-4, CSREA Press ©

SESSION

EMBEDDED SYSTEMS + SOFTWARE COMPLEXITY AND QUALITY ISSUES

Chair(s)

TBA

ISBN: 1-60132-446-4, CSREA Press ©

ISBN: 1-60132-446-4, CSREA Press ©

Model Driven Architecture for Development Test Automation Tools

Wilson Hissamu Shirado
Department of Computer
Londrina State University
Londrina-PR, Brazil
hissamu.shirado@gmail.com

Jandira Guenka Palma
Department of Computer
Londrina State University
Londrina-PR, Brazil
jgpalma@uel.br

Vanessa Matias Leite
Department of Computer
Londrina State University
Londrina-PR, Brazil
vanessa.matiasleite@gmail.com

***Abstract*— Embedded systems are present in various branches of the human activity, to the point of up to 98% of the produced processors in world have been allocated in applications of this type in 2002. Given this wide applicability of embedded systems, it is common to find them in critical systems or equipment, making the testing phase during the development process of crucial importance. This proposal seeks for new development paradigms, called Model Driven Development, also known as MDD and MDA (Model Driven Architecture). Against this backdrop, this work aims to establish a process of test automation system's development for embedded systems that can be oriented by these architectures and reconcile these two research fronts.**

Keywords: Embedded Systems; Software Testing; Testing Automation; MDA.

I. INTRODUCTION

The manufacturing industry requires a high dynamics of releases of new products with differentiated features and functionalities depending on the market demand [1]. One way usually adopted to meet these requirements is achieved through application of electronics and embedded computing in various products and branches of human activity [2] [3].

However, despite these facts, problems in these components may result in high costs arising from recalls, damage to corporate image or in extreme cases, it may cause irreparable losses to its users. Thus, testing is an important step in the process of development of software embedded in these products. However testing activities are characterized by being a very slow, costly and repetitive phase in the development process. Polo [4] states that testing activities can represent up to 60% of all the efforts and costs of a software development project.

Faced with this scenario, according to Polo [4], Fewster and Graham [5] and [6] Binder, applying automated test methods can contribute significantly to save time and other resources when compared to manual testing processes, since it allows quickly identification of faults, agility in the debugging process, capture and analysis of results consistently in addition to facilitate the execution of regression tests throughout the development process. Nevertheless, despite all of the advantages pointed out by the authors, currently this practice is not widespread within software development companies [7]. Among the factors that contributes for this situation are the fact that the development of an effective program of test automation involves itself a development cycle, which in many cases involves a high cost of implementation and maintenance of these automation systems [7] [8].

In another perspective, researches and other initiatives directed to improve processes and methodologies focused on the development of software have been emerged, such as the proposals MDD-Model-Driven Development and MDA-Model Driven Architecture, the latter being an implementation of the MDD presented and maintained by the OMG – Object Management Group.

This work is organized as follows: section 2 discusses the MDA concepts, software testing and other theoretical concepts that will provide the theoretical basis on the subject. Section 3 presents the proposal of this study, which is developing a process that aim to model and code functional test automation systems based on MDA. Section 4 presents the analysis results that are obtained from an experimental test and bibliography. Finally in section 5 the conclusions of this work are presented.

II. THEORETICAL FOUNDATIONS

Test activities are present in the development processes of any product, not being different for software products. However, during the process of software development, testing activities are generally expensive in terms of time and cost, representing between 30% to 60% of all costs of a project depending on the criticality and complexity [7] [4].

Automation of part of tests in a project is cited by several authors [7] [4] as a solution for the reduction of these adverse impacts, so that various technologies and tools for automation of tests have been proposed and developed over time, as the technologies of capture and replay, XUnit frameworks and automation tools for testing data combination [4]. However, it is relevant to note that the test automation is a task that goes far beyond simple application or use of a technology or tool. The development of an efficient automation system involves an own development cycle covering carious phases, from planning, modeling, coding and implementation of automated tests [8] [9].

In front to this scenario, it is introduced the concepts of Model-Driven Development-MDD. The MDD is a

methodology that focuses on creating models as the main class of artifacts for software development [10]. The MDD provides guidelines, languages, methods, transformation models and tools to support the representation of requirements and subsequent transformations that allow the generation of a complete technology solution at the end of the process [11]. The Model Driven Architecture-MDA is an implementation of the MDD proposed by OMG that emphasizes primarily the transformations between models, being fundamentally supported by technologies and standards maintained by the OMG as UML, MOF and XMI.

A. Embedded systems

According to Heath [12] embedded systems are defined as systems developed to control a function or set of functions, based on microprocessors, which are not designed to be programmed as personal computers.

A branch of industrial activity that makes frequent use of embedded systems in its products is the automotive sector. An example of application of these systems are the ECU's (Electronic Control Unit) which are installed as various modules types in different locations in a car, being responsible for numerous tasks such as, electronic injection control, traction and stability control, tracking, etc. According to Broy et. al [13], it is estimated that currently, a car has about few tens of thousands of lines of coding embedded.

The development of automotive embedded systems is a fairly complex task depending mainly on the limitations mentioned above as well as the high degree of demand of automakers and OEM's (Original Equipment Manufacturers) in terms of quality and reliability of its products, arising from increasing demand for, comfort, safety and quality of its consumers.

B. Software testing and test automation

According to Pressman [14] the testing activities consist of a controlled execution of software in order to find an error. Thus, it is considered as a critical element of software quality assurance, representing the final review of the specification, design and code generation.

Thus, many companies have focused their efforts on achieve the completeness of tests in the shortest time possible, leading to automation of the test activities [15]. Dustin [8] sets the test automation as automation, management and execution of test activities, including the development and execution of test scripts and verification of test requirements through automated tools. Thummalapenta [16] adds that test automation is an activity in which one seeks to create a mechanically interpretable representation of a manual test case. In addition, Kusarinen [7] points that the researches carried out within the software engineering field have two primary objectives: reduction of production costs and increasing the quality of software products. Once the software testing activities represent a role of great importance to these two questions, the proper implementation of automation in software testing processes can properly enable an interesting combination of these two goals.

An important factor for test automation focusing embedded systems is related to the simulation of the environment in which the SUT- System Under Test must act, as well as the stimuli provided to the SUT for this simulated environment. The studies [17], [18], [19] and [20] address the subject, proposing ways to implement such simulations, without however perform a proper modeling of these environments. In addition, all these papers aim solutions to a specific business in which the embedded systems are present, especially the automotive sector.

C. Model Driven Architecture- MDA

The MDA is an initiative that has as its main objective, to extract value from models and modeling processes, thus providing a way to deal with the high complexity and interdependencies that exist in software systems. Thus, the MDA standard proposed by OMG propones that there are several ways to add value to a development process models, such as [21]: using models as a means of intercommunication, automatic derivations of models via automation, simulation and implementation of models and automated extraction of model's information.

The main feature of the MDA focuses on transformations between models. The automation of the transformations of high-level models to executable information systems provides a number of advantages, citing reduces in cost, time, risks of production and maintenance of a system, acting at the same time in the refinement of the system to the specific domain of the problem [21]. However, not all models are suitable for automatic transformations. A model needs to be complete and precise enough to describe the information of the system.

The MOF-Meta Object Facility is a specification proposed by the OMG in order to create a behavioral pattern that can dictate forms of models sharing between various computational tools [22].

The models and relationships between models are one of the main points on the MDA development approach. In this approach, three categories of models are stablished, according to the adopted point of view and the "intention" of the model:

Computation Independent model (CIM): this model represents a vision of the system in which the understanding is independent of computer knowledge. The requirements modeling of a system is usually accomplished by using a computer-independent model, focusing on the role of the system in the environment in which the system will operate [23].

Platform independent model (PIM): the PIM is a platform-independent system implementation. The PIM focuses simultaneously on the description of the operation of the system and hide features related to a particular platform, and thus can be reused across different platforms [23].

Platform specific model (PSM): the PSM is a vision of the system that considers specific details to the development platform. It is created from the platform-independent model (PIM) [11]. A PSM can already be regarded as an implementation of the system and should provide all the information needed for the construction and operation of the system [23].

It is important to note the difficulty in the realization of the automatic transformation from CIM to PIM, there are only few works that discuss this type of transformation. The most part of analyzed papers propone only the PIM-PSM transformation, working with CIM to PIM transformations in manual way. This is due to the non-existence of artifacts and/or specific rules defining which elements should be used on the CIM and consequently there is no standardization for mapping CIM-PIM tranforms, this can be observed in several works [24], [25], [26], [27].

III. Proposal of an MDA for Development of Test Automation System

In this work, a system of test automation for embedded systems is proposed. Testing activities on embedded systems are usually performed by means of stimuli (inputs) that are produced by input components, then the SUT checks if the device reacts as described in test cases, i.e. if the SUT presents the desired outputs in the respective output component. As the proposal is to perform automated testing without using the human senses, it is also necessary to identify which equipment are required to perform the capture of outputs automatically, and compares them with the desired output, these elements are inserted into the system and were named capture components.

Thus, the purpose of this work is to create a process of developing functional test automation systems for embedded systems, establishing means to automatically execute the test cases, also establishing ways to insert the components/entities responsible for capturing the State of the device under test to the model.

The scope of this work addresses the process of automation of the tests, not covering the creation of test cases. Thus, the proposal of this study starts assuming that the test team already has a document with the information necessary to perform the tests, which are normally carried out manually.

Aiming reducing the gap between the CIM and PIM, in this work, it was developed a syntax that allows a better writing and organization of test cases, and also, enables a standardization of test documents, making possible the application of automatic transformations from CIM to PIM of automated functional tests in embedded systems.

In this way, since the MDA adopts mainly the UML as a modeling language, in this work were employed as target models of CIM to PIM transformations the UML's Class diagram to structural modeling and the sequence diagram for the modeling of automation scripts. These elements are created in order to submit them to MDA automatic transformations for system's code generation.

Thus, it was proposed a development process for creating test automation tools based on the guidelines of the MDA and the activity diagram presented in Figure 1. In the next section, it will be made a brief explanation of each activity present on the diagram. The completeness of all these activities is described in Shirado [28].

A. Computation Independent Model –CIM

CIM models are related to the application's domain. This proposal aims to minimize the gap between CIM and PIM observed so far, so that in this work it was prepared a proposal of analysis and interpretation of test documents that enable the creation of a data form that describes the problem using a defined syntax.

So it is necessary to fill the form with the following information extracted from the tests document: Class In: classification of input components; Components In: Components responsible for providing stimuli to the SUT; Behavior In: behaviors presented or performed for input components; States In which a particular States input component can take during their operation; Parameters: Possible information or messages that may be forwarded to the SUT for an input device; Classes Out: classification of output components; Components Out: components responsible for capturing a behavior from the SUT; Behavior Out: components responsible for performing a behavior in response to a given input; States Out: States that a given component of output can assume; Classes Cap: classification of capture components; Components Cap: components responsible for capturing a behavior from the SUT; Behavior Cap: behaviors or actions that can be performed by a component.

In Figure 2 is described the base syntax created using the BFN - Backus-Naur Form, without the terminal symbols. This syntax helps establishing a standardization for test description, once tests are typically written in natural language and in different ways depending on the person who written it. This syntax standardizes test documents, making possible the

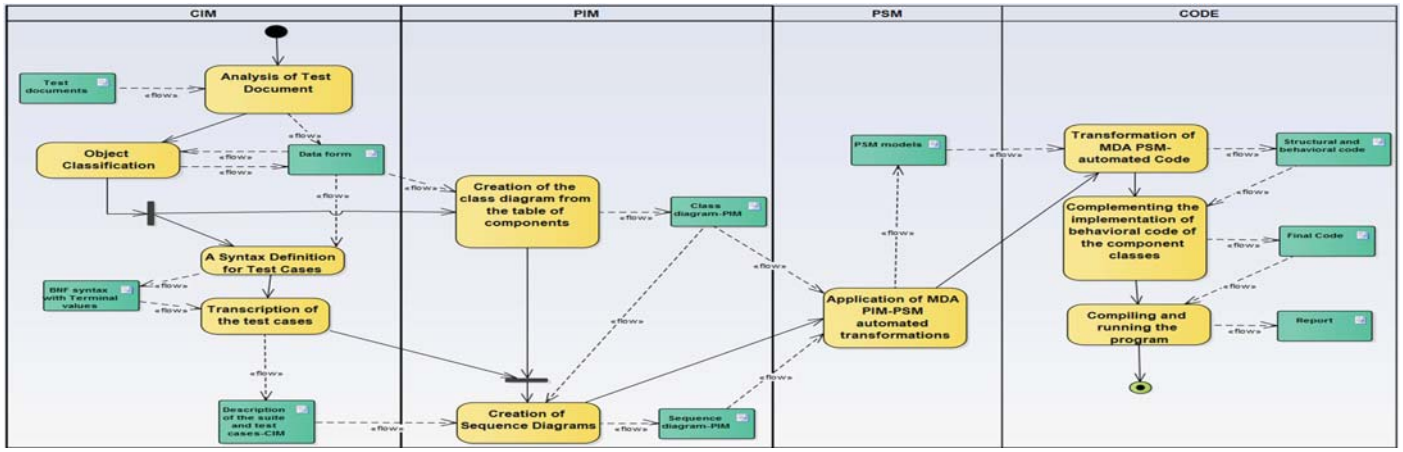


Figure 1: Proposed development process [28].

<Test Suit> ::= <Test Suit Name> ' : TestSuit' ' { ' (<Test Case> ';') * ' } ;'

<Test Case> ::= <Test Case Name> ':' <Test Case Class> '{ '<Test Setup> ';' (<Test Step> ';') * ' };'

<Test Setup> ::= ' Setup: { ' ('(' (<Component In Name> <Component In Class> [' : state = '<State>]) * ')' | (' ('<Component Out Name> <Component Out Class> [' : state = '<State>]) * ')'| ' ('(<Component Cap Name> <Component Cap Class> [' : state = '<State>] ')') * ' };'

<Test Step> ::= <Step Name> ': {' [('(' <Step In> ')') *] [(' : (' <Step Out> ')') *] [(' : (' <Step Cap> ')')*] ': (' <Assert> ') ;'

<Step In> ::= <Behavior In> '(' [< Parameter > *] ')' ' , ' <Component In Name > ';' <Step Cap> ::= <Behavior Cap> '(' [(< Parameter >) *] ')' ' , ' <Component Cap Name> ';'

<Step Out> ::= <Behavior Out> '(' [(< Parameter >) *] ')' ' , ' <Component Out Name> [' : ' <State>]';'

<Assert> ::= '{ assert({ ' <Component Out Name> ' }, {' <Component Cap Name> ' }), ' <Test Case Name> ' };'

Figure 2: Syntax proposal [28]

application of automatic transformations of CIM to PIM within the domain of applications for Automating functional tests in embedded systems.

1) Analysis of Test Document

This proposal starts with a detailed analysis of the test cases, which are independent computing documents (CIM). Through the difficulty observed to execute automatic transformations of test document from CIM to PIM in the MDA approach, it was structured a process of interpretation and analysis of this document in order to enable automatic transformation of CIM to PIM.

The selected tests for automation should then be reviewed and the following information should be identified:

- What are the inputs provided to the SUT and the components responsible for these interactions;
- What are the outputs or expected behaviors in response to a provided entry and the components responsible for carrying out these behaviors.

This analysis consists of an evaluation of all test cases contained in the document one by one, selecting those that have greater possibility/ feasibility of automation following criteria found in literature review of [17], [18], [19] and [20] and technical or materials restrictions.

Once identified and selected, the test cases should be organized and used for filling out a form that contains in addition to the information already explained, other important information as the possible states for each component and any messages or parameters that can be supplied to SUT.

2) Object Classification

Once the form is pre-populated, it will then be submitted to a refining process in which related data in the fields of components will be sorted according to common characteristics and behaviors. It is important to note here that these data represent future objects that will be part of the automation system, so that this classification is of utmost importance since it will determine a set of which classes could be created within the class diagram of the test automation system.

Once this classification is finished, the established classes should be listed in the fields Classes In, Classes Out and Classes Cap from the form.

3) Syntax complementation for test cases

After filling out the form, the next activity proposed by the process described in Figure 1 is related to the generation of a document that follow a formal syntax and defined using a grammar in BNF. The choice for the use of the BNF language was mainly because that is the language adopted as syntax specifications by the OMG on the formal specification of the UML. Once one of the basic philosophies adopted in this study is to use and maintain the standards and recommendations of the OMG for modeling and MDA, it was made the option to use the BNF language.

At this point, it is necessary to finish the definition of the language to be adopted for each test case, the terminal symbols of the syntax in Figure 2 will be defined based on the data present on the form. The label of each field of the form corresponds to a non-terminal symbol and each value listed in this field represents a terminal element of the syntax.

4) Transcript of the test cases

After the completion of the syntax and complementation of values of terminal elements according to the context and characteristics of the SUT, this syntax will be used for transcription of the test suites and test cases present in the test.

In that way, there will be three sub activities in this stage:

- The test cases will be taken from the test one by one and will be transformed from textual description in natural language to a formal description using the syntax defined;
- In addition to the transcription of the steps explicitly presents in the tests, it will be added to the test cases the operations related to the capture processes of the behaviors of the SUT;
- If necessary, the test cases above generated can be grouped into the same test suite, to automate their execution.

Finally it is worth mentioning the fact that all test case must be obligatorily within a suite of tests, even though this test suite contains a single test case.

B. Platform Independent Model - PIM

The PIM models in MDA's proposal are models that aggregate computational concepts and information to the models. Thus, the modeling of the PIM will be made through the classes and sequence diagrams of UML, for being two of the more usually used diagrams in systems modeling. In addition, it is also notable the natural associativity among these diagrams, which provides a structural and behavioral description of a software, including. Thus, this section is

dedicated to the explanation of the activities involved in the creation of these diagrams.

1) Class Diagram Creation

For the development of the class diagram that will model the entities of the automated functional testing tool, it is necessary to make some considerations about the structure and purpose of this diagram. The classes to be created and included in this diagram must fit into one of three categories of classes previously defined: the first category of classes will represent the components to be tested in the SUT, that is, models representing the device to be tested; a second category, will be used to model other equipment and devices to aid the test executions, especially regarding to the interaction and behavior/state capture presented by the physical SUT; and finally a third category in which classes will be responsible for shaping and executing the suites and test cases. The basic structure of the diagram is presented in Figure 3. The new classes that are inserted to this structure must be inserted exclusively through inheritance, implementations and operations overhead. So, this structure acts as a Framework, in which basic functionality necessary for the execution of the tests have already been implemented.

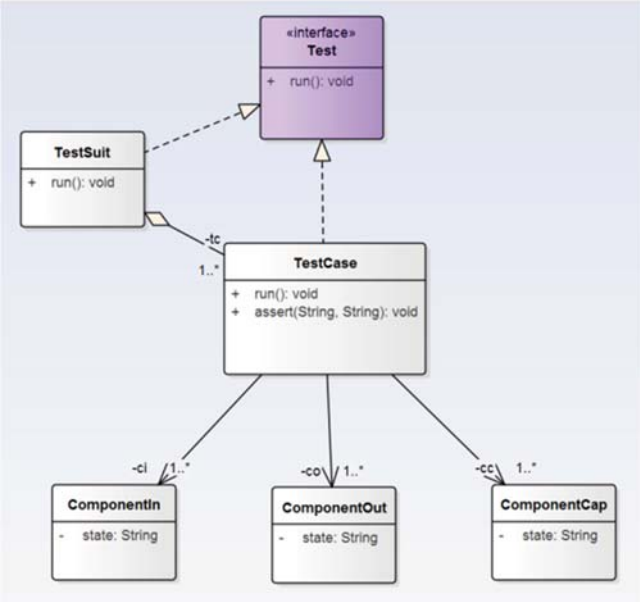


Figure 3. The initial structure of the class diagram [28].

In addition, another point of relevance to the automation of the tests is to capture the behavior presented by the SUT. The behavior of the embedded system varies according to its application and purpose, and may take a wide range of possibilities, such as the lighting of an indicative light, a message on an LCD display, variations in analog pointers, triggering various actuators and so on. Thus, it is of great importance that the behavior of the embedded system has been fully understood in the previous activities of the process.

2) Sequence Diagrams

The UML sequence diagrams model the sequence of events as well as the interaction between various objects through messaging and invocation of methods among them [29]. In light of these particularities, the sequence diagram has very useful features for modeling of test cases, which largely consist of sequences of activities to be carried out in order to verify and validate behaviors of a system.

Thus the sequence diagram was chosen for behavioral modeling of test cases in this proposal. Each sub-class derived from the TestCase class modeled on previous activity must implement a method called run(), responsible for execute the test case, and the modeling of the internal behavior of this method will be made by means of a sequence diagram. For this, it will be used the information present in the test cases written using the BNF syntax proposed in this work. That is, each row < Test Case >:: = < Test Case Name > ': ' < Test Case Class > ' {' < Test Setup > '; ' (< Test Step > '; ') * '}; ' will be turned into a sequence diagram, as shown in Figure 4.

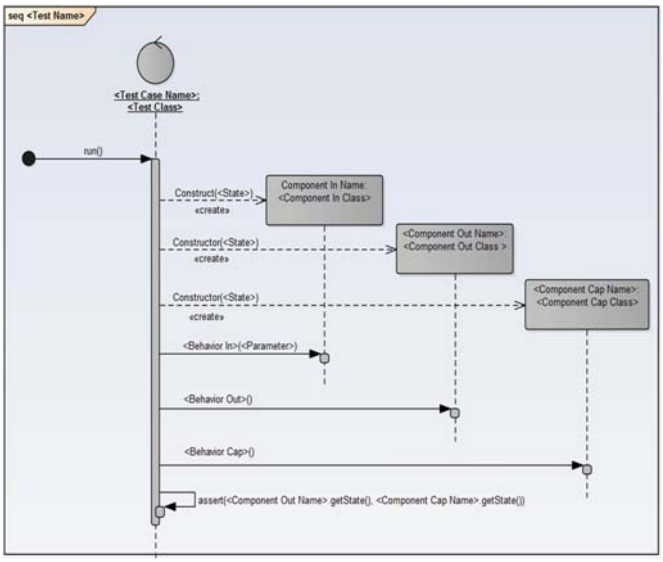


Figure 4. Addition of Lifelines instantiated to diagram [28].

C. Platform Specific Model - PSM

Until the present moment, it were traced the activities related to modeling of the test automation tools proposed in this work, and as a result of these activities, it were generated class and sequence diagrams which describe the structure and behavior of a system for automatic execution of functional tests in embedded systems, in the form of PIM models. Given this, and following the proposal of the MDA, this section will be dedicated to address the steps related to the transformations of PIM to PSM models.

1) Application of MDA automated transformations in class diagrams and Sequence diagrams

Once the proposed diagrams modeling have been finished, the modeled diagrams shall be submitted to MDA transformations from PIM to PSM models. Thus, at this stage it is necessary to define the desired target language for the tool.

The transformations from PIM to PSM models are performed through the mapping of the elements that make up a UML model or other language based on the MOF meta-model. This mapping must establish a relationship between elements of the target and source models. The adoption of a standard form for the representation of the elements of a model such as the XMI - XML METADATA INTERCHANGE, allows

besides the interchangeability of models between different modeling tools and processing, the identification of each element present in the source and target models from the tag structure established for the UML. That is, the format of models sharing XMI, allows each element of a model to be mapped and later transformed to different platforms. However, it should be noted that this proposal does not aim to create the mapping of the UML models for a specific platform, but apply existing mappings used in tools created and maintained by companies associated with the OMG.

Therefore, using the structure established by the OMG in which the necessary rules for mappings and transformations are defined and also using the XMI format that organizes all data in a model through its structure tags, tools which offer support to MDA, already implement mappings of the models elements to different platforms. On OMG's web page it is possible to find a list of tools and partner companies that support the MDA, such as CASE tools Enterprise Architect from Sparx Systems company, Rational Software Architect by IBM, OptmalJ ny Compuware among others.

D. Code

After the analysis and structuring of the CIM models and later transformations of CIM to PIM and PIM to PSM, the next step is related to the transformation from the platform-specific model to the source code to the specified language. Thus, in this section are covered the activities of the proposed process related to this transformation, namely: transforming model PSM to code, implementation of the behavioral code of classes and finally, the compilation and execution of the test automation tool.

1) *Transformation from PSM to code*

It is important to note that among the classes modeled and undergoing transformations so far, those that represent components of input, output and the SUT's behaviors capture, will have only its structural code generated automatically. The TestCase classes and TestSuit in turn will be transformed and will have their structural and behavioral codes generated automatically by MDA transformations. The behavioral feature of each of these classes is represented by the run() method, which will have its code based on its sequence diagram.

Structurally, the MDA presents all rules and information necessary to map of model elements to code elements of a specific programming language. It is achieved by the several layers of models and met-models defined in MOF standard. However, the most common mappings and transformations implemented by MDA tools focus on static diagrams and structural transformations as the class diagram. Behavioral diagrams transformations are still poorly supported in tools.

However, once the sequence diagram modeled in this proposal focus on the specific domain of software testing, it is possible to generate a diagram which has exactly what should be present in the final code. Thus, the mapping process can include all the elements of the sequence diagram stablishing a relationship between these model elements and the code elements through the information present in the XMI version of the model that describes all the graphic components used to model the diagram.

2) *Behavioral code Implementation*

Once all the transformations of PSM to code proposed by the MDA have been finished, all structural code and part of the behavioral code of the test automation tool is complete. However, it should still be performed behavioral implementations of the methods of the sub classes of component classes. These implementations must be carried out by a programmer, once the modeling of these methods can result in very complex models.

3) *Compiling and running the program*

Finally, after the completion of all activities previously described, the proposed process has achieved its end. Then the compilation of the code using an appropriate compiler for the target language must be performed, enabling the running of the tool responsible for automatically execute test cases modeled.

IV. Results Analysis

To analyze the results of this proposal, it was conducted an experimental test, in which were used a test case of an automatic automotive headlamp control system, controlled to a light sensor and a three-position key, On, Off and Automatic adapted from Bringman and Kramer [17]; It was also adopted the MDA tool Enterprise Architect as a modeling tool. The full experimentation can be found in Shirado [28].

It was observed that few works deal with the automatic transformation of CIM to PIM, being this fact observed as a great difficulty related to the MDA approach. Most of the analyzed works deal with the PIM-PSM-code transformation, executing CIM to PIM transformation manually. This fact is due to the lack of specific rules and artifacts that define which elements should be used on the CIM and consequently there is no standardization for mapping CIM-PIM. This can be observed in several works [24], [25], [26], [27].

In front of this fact, the creation of the experimental test form and the BNF syntax demonstrated the feasibility of converting CIM to PIM for a specific test case, which made possible to make a mapping of elements aiming the transformation of the CIM model to PIM models of class and sequence diagrams. In this way, it is possible that, from the CIM model, automatic transformations be carried out from a mapping of the BNF syntax elements defined in this study to the tags of an XMI document of classes/sequence diagrams. This task should be done by means of a transformation tool that obeys the rules stablished on XMI standard and the syntax defined in this study.

In addition, the test automation process is related to the simulation of the environment in which the SUT must act, as well as stimuli provided by this environment. The works of [17], [18], [19] and [20] address this issues, suggesting ways to implement simulation, without however perform proper modeling of these environments. In addition, all these works propose solutions to a specific business in which the embedded systems are present, especially the automotive sector.

The proposal presented here differs from what was reported in the paragraph above, this paper is handled the required modeling proposed environment and not only the implementation as in the works cited.

V. CONCLUSION

The transformation of CIM to PIM are usually little addressed in works related to MDA approach, since the own OMG does not set a standard for the CIM. In front of this scenario, this work proposed a BNF syntax for Standardization of test documents, supported by a structured data form responsible for condense the main information about the tests to be automated. Thus, a CIM and/or test documents written using this syntax enables the creation of tools that automate the transformation of CIM-PIM, once it enables the establishment of a mapping between source and target model elements.

The transformations of PIM models for PSM and PSM to code also were addressed in this study, the structural models transformations already exist in CASE tools that offer support to the MDA approach. For behavioral models transformations, in case, the sequence diagram, which is usually is not supported by CASE tools, this work produces sequence diagram for each test case that can be transformed directly into code without requiring additional data, using the XMI document.

However, even though the code generated by following the guidelines of the MDA still requires manually coded additions regarding to the contents of methods of classes that specialize the classes of components, these methods are easily implemented, since it does not involve the logic implementation of the SUT, which acts as a black box in the process. These methods are generally responsible for sending input signals to the SUT or capture an expected behavior in response to this entry. However, these implementations performed manually in the code can be reused in different test cases, so in a long-term perspective, it is possible that the use of the process proposed in this work have as a product, a repository of models and codes, which will allows a gradual reduction of the need for hand-coding in the process, to the point where test automation systems can be generated completely from templates and reuse of code components. This characteristic, combined with the inherent advantages of the MDA approach allows certain advantages to the process of development and implementation of automated testing tools for embedded software once the modeling performed significantly improves documentation of testing.

REFERENCES

[1] Karsai, Gabor, et al. "Evolving embedded systems", Computer 43.5 (2010): 34-40.

[2] Malinowski, Aleksander, and Hao Yu. "Comparison of embedded system design for industrial applications." Industrial Informatics, IEEE Transactions on 7.2 (2011): 244-254.

[3] Barr, Thomas W., Rebecca Smith, and Scott Rixner. "Design and implementation of an embedded python run-time system." Presented as part of the 2012 USENIX Annual Technical Conference (USENIX ATC 12). 2012.

[4] Polo, M. et al. “Test Automation”. [S.l.]: IEEE Software, v. 30, 2013.

[5] Fewster, Mark, and Dorothy Graham. “Software test automation: effective use of test execution tools”, ACM Press/Addison-Wesley Publishing Co., 1999.

[6] Binder, R. V. "Testing Object-Oriented Systems: Models, Patterns and Tools. 1999." (1999): 261.

[7] Kasurinen, Jussi, Ossi Taipale, and Kari Smolander. "Software test automation in practice: empirical observations." Advances in Software Engineering 2010 (2010).

[8] Dustin, E., Rashka, J., Paul, J. “Automated Software Testing: Introduction, Management, and Performance”. 13ª. ed. Boston: Addison-Wesley, 2008

[9] Rätzmann, Manfred, and Clinton De Young. “Software Testing and Internationalization”. Lemoine International, Incorporated, 2003.

[10] Vara, Juan Manuel, and Esperanza Marcos. "A framework for model-driven development of information systems: Technical decisions and lessons learned." Journal of Systems and Software 85.10 (2012): 2368-2384.

[11] Teppola, Susanna, Päivi Parviainen, and Juha Takalo. "Challenges in deployment of model driven development." Software Engineering Advances, 2009. ICSEA'09. Fourth International Conference on. IEEE, 2009.

[12] Heath, S.” Embedded Systems Design”. 2ª. ed. Burlington: Newnes, 2003

[13] Broy, Manfred, et al. "Engineering automotive software." Proceedings of the IEEE 95.2 (2007): 356-373.

[14] Pressman, Roger S. "Software Engineering" (2006).

[15] Ramler, Rudolf, and Klaus Wolfmaier. "Economic perspectives in test automation: balancing automated and manual testing with opportunity cost." Proceedings of the 2006 international workshop on Automation of software test. ACM, 2006.

[16] Thummalapenta, Suresh, et al. "Automating test automation." Software Engineering (ICSE), 2012 34th International Conference on. IEEE, 2012.

[17] Bringmann, Eckard, and Andreas Kramer. "Model-based testing of automotive systems." Software Testing, Verification, and Validation, 2008 1st International Conference on. IEEE, 2008.

[18] Huang, Yingping, et al. "Design validation testing of vehicle instrument cluster using machine vision and hardware-in-the-loop." Vehicular Electronics and Safety, 2008. ICVES 2008. IEEE International Conference on. IEEE, 2008.

[19] Yongfeng, Yin, Liu Bin, and Zheng Bentao. "On test script technique oriented automation of embedded software simulation testing." Computer Science and Information Engineering, 2009 WRI World Congress on. Vol. 7. IEEE, 2009.

[20] Moon, Huichoun, et al. "Automation test method for automotive embedded software based on AUTOSAR." Software Engineering Advances, 2009. ICSEA'09. Fourth International Conference on. IEEE, 2009.

[21] OMG. MDA Guide. OMG. Boston. 2014.

[22] OMG. *http:* //www.omg.org/mof/. [S.l.]: [s.n.], 2015.

[23] OMG. MDA Guide. [S.l.]: [s.n.], 2003.

[24] Calic, Tihomir, Sergiu Dascalu, and Dwight Egbert. "Tools for MDA software development: Evaluation criteria and set of desirable features." Information Technology: New Generations, 2008. ITNG 2008. Fifth International Conference on. IEEE, 2008.

[25] Gailliard, Grégory, et al. "Transaction level modelling of SCA compliant software defined radio waveforms and platforms PIM/PSM." Proceedings of the conference on Design, automation and test in Europe. EDA Consortium, 2007.

[26] Guttman, Michael, and John Parodi. “Real-life MDA: solving business problems with model driven architecture.” morgan kaufmann, 2006.

[27] Alves, E. L. G.; Machado, P. D. L.; Ramalho, F. “Automatic generation of built-in contract test drivers”. Software and Systems Modeling Journal. , New York, v. 13, 2014.

[28] SHIRADO, W. H. “Model Driven Development of Functional Test Automation Systems For Embedded Systems”. 118p. Master’s Thesis (Master Degree of Science in Computer Science) – State University of Londrina, Londrina-PR, 2016. unpublishe

[29] Guedes, Gilleanes TA. "UML 2- A Practical Approach” , 1st Edition." (2008).

Fractal Beauty of Programming Style

Ron Coleman, Pritesh Gandhi
Computer Science Department, Marist College, Poughkeepsie, NY, United States

Abstract - *The literature on how to write programs with good style is extensive but it contains almost no references to measuring sensori-emotional or aesthetic values in source. We study this problem and propose source beauty can be measured with a simple, relativistic model based on the fractal dimension. We hypothesize the model is statistically related to yet different from software complexity. Experiments using GNU/Linux source as the test bed indicate the model is only weakly or moderately correlated with software complexity yet in directions consistent with anecdotal prescriptions for good style. The data further indicates removing comments do not improve source beauty but neither does adding them, a finding at odds with advice in some style guides. Yet mnemonics are correlated with software complexity, which comports with model predictions and some style guide recommendations to make code "self-documenting." These results suggest the model may help developers craft and maintain beautiful codes as a best practice.*

Keywords: Programming style, fractal analysis, software complexity, readability

1 Introduction

In 1974 Donald Knuth, author of *The Art of Computer Programming* [1], gave a talk on computer programming as an art [2]. He noted that in 1959, when the *Communications of the ACM* first began publication, the editors had hoped someday to see "a transition of programming from an art to a disciplined science." [2] The implication, Knuth observed, was that "there is something undesirable about an area of human activity that is classified as an 'art'; it has to be a Science before it has any real stature." For Knuth, art in the context of programming was synonymous with the classical definition of art, namely, skill or application of knowledge. "When I speak about computer programming as an art," he said, "I am primarily thinking of it as an art form, in an aesthetic sense. The chief goal of my work as educator and author is to help people learn how to write *beautiful programs*" (his emphasis).

Knuth was not addressing what is beauty in the ontological sense but rather what is knowable about such beauty in the epistemological sense that he could apply and teach, e.g., good programming style. He did not give detailed instructions but instead referred to *The Elements of Programming Style* [3], which did. Software engineers at AT&T Bell Labs automated some of these ideas in *cb*, the C beautifier [4], the first program to beautify other programs in accordance with what was known about reliable, well-written code, in this case, the K&R style [4]. The modern successor of cb for the Linux environment is *indent*, which can generate K&R as well as GNU, BSD, and Linux styles [6]. Yet indent and the many other utilities like it have two main shortcomings. First, they only reformat code; they don't refactor it. Second, the utilities do not quantify the value of their beautifying treatments, leaving programmers to reason about aesthetic outcomes with ad hoc arguments and guesswork rather than metrics. Coleman and Gandhi [7] hypothesized that programming style might be related fractals since fractals are often associated with beauty [8, 9]. To test their hypothesis, they studied C programs from the GNU/Linux repository and showed that changes in style were systematically correlated with changes in fractal dimension [10]. The authors did not propose the fractal dimension was itself a measure of beauty but noted instead a weak correlation with lines of code.

In this paper, we extend the work of Coleman and Gandhi [7] and propose a simple, relativistic fractal model of source beauty source called the *beauty factor*. We hypothesize that the model is statistically related to yet different from software complexity. To test our hypothesis, we use the same test bed as Coleman and Gandhi [7]. We show that beauty factors from de-beautifying and beautifying treatments are correlated with software complexity in expected directions and are mostly consistent with prescriptions in style guides. However, the correlations are weak/moderate. In other words, software complexity and beauty factors are indeed related but they are not proxies for one another. Beauty factors appear to measure other values in source that may aid in crafting and maintaining beautiful codes.

2 Related work

Although the literature on programming style is extensive [3, 6, 11, 12, 13, 14, 15, 16], it gives very little attention to measuring aesthetic values in source code. The justifications for good style are frequently readability and "defensive programming," although the relationship between readability and defects/failures is controversial [17, 18, 19, 20]. The assumption is that reading code is similar to reading prose [21]. Thus, readability is about understanding code [22] and aesthetics, appreciating it [7]. Teague and Lister [23] find reading code precedes writing it; we posit looking at code precedes even understanding it and that act has value. An example is *Distellamap* [24], a visualization of the *Pac-Man* code found in an Atari 2600 cartridge [25]. While some individuals may comprehend 6507 assembly code, which is not known for its high degree of readability, that that code is on exhibit at the Museum of Modern Art is prima facie evidence of its aesthetic value. *Beautiful Code* [26] studies

conceptual beauty in the design and efficiency of algorithms and data structures, testing and debugging; they are not considering style. Kokol, et al, reported evidence of fractal structure and long-range correlations in source [27, 28, 29, 30]; however, they were investigating not style but software complexity using lexical analysis of a small sample of randomly generated Pascal programs. Coleman and Gandhi [7] used Fractop [31] and found strong correlations between style and fractal dimension. They also identified weak correlations with lines of code, which the authors interpreted as evidence of robustness of the fractal measure. In this paper, we use the same test bed of 114 C files as Coleman and Gandhi [7] except we strip the files of non-functional syntax external to functions and generate separate files each with one function, which increases the number of files to analyze by almost tenfold. Our work is similar to [32, 33, 34] except these efforts were measuring aesthetic appeal in artificially intelligent pathfinding in videogames. While our focus is software development, our approach implies programs are works of art and thus, we borrow conceptually from others who have worked in this area investigating fractal analysis of paintings and masterpieces [35], in particular Pollack's "action paintings" [36, 37, 38, 39].

3 Methods

3.1 Some working definitions

For the purposes of this paper, we define *style* as the lexical form or structure of source code. Coleman and Gandhi [7] surveyed different style guides and found general agreement on three rules they called the "basic tenets":

1) Use white space.
2) Choose meaningful or mnemonic names.
3) Include documentation.

We define *beauty* as the measure of style. The model we give below assigns a quantitative value.

There is no consensus on what *software complexity* is which may explain why there are more than 100 different software complexity metrics [40]. Thus as a starting point, we define software complexity as lines of code (LOC) or McCabe's cyclomatic complexity, M [41]. Hatton [42] found LOC and M are strongly correlated.

3.2 Treatments

The first step after extracting the C file from the database is to manipulate its style through one of the de-beautifying or beautifying regimes below.

Table 1 Source code treatments

Treatment	Type	Tenet
None	Baseline	∅
De-beautify	Remove indents	1
	Randomize indents	1
	Non-mnemonic refactoring	2
	De-comment	3
Beautify	GNU [43], K&R [3], BSD [44], Linux [45] styles	1
	Mnemonic refactoring	2
	Re-comment	3

Removing indentation strips all white space on the left side of each line. Randomizing indentation inserts 20-40 randomized spaces on left side of each line. De-comment removes all external (to a function) comments; internal comments are not affected. Non-mnemonic refactoring changes the occurrence of high frequency names to be less mnemonic. See Coleman and Gandhi [7] for the algorithm. The indent command applies one of GNU, K&R, BSD, or Linux (device driver) styles to a file. Mnemonic refactoring changes the occurrence of high frequency names to be more mnemonic. Again, see Coleman and Gandhi [7] for the algorithm. Re-comments inserts at least one randomly selected comment from a database of comments compiled from the test bed.

3.3 Artefacts

After manipulating the source style, we generate an artefact, which is an in-memory bitmap representation of the source code. It is encoded in two ways: literal artefact method (LAM) or block artefact method (BAM). LAM builds a "literal" representation, that is, as it would appear say, in a text editor with fixed width font. Consider `hello.c` in the figure below. This is its LAM encoding if it were written to an image file.

```
#include <stdio.h>
int main(int argc, char** argv) {
  printf("Hello, world!");
  return 0;
}
```

Figure 1. hello.c with LAM encoding

BAM builds a "block" representation. In this case, each character, except for spaces is represented by a graphic, a filled rectangular block. Spaces are blanks without blocks. That is, it obliterates the text in favor of these graphics. The same `hello.c` file BAM encoding is shown in the figure below if it were written to an image file.

Figure 2. hello.c with BAM encoding

BAM removes language dependencies of the programming language and complements the mnemonic refactoring and documentation analysis while preserving the layout. Finally, data from Coleman and Gandhi [7] shows LAM and BAM encodings are 95% correlated. Thus, we use only BAM encoding in our investigation here.

3.4 Software complexity

LOC is the simple line-by-line count, including comments if any. M, McCabe's cyclomatic complexity, [41] is given by the

following equation:

$$M = \pi + 1 \quad (1)$$

where π is the number of decision points in the function. For the C language, the decision points are `if`, switch-`case`, `for`, `while`, `do-while`, the operators, `&&` , `||`, and `?:`.

3.5 Fractal dimension

The final phase measures the fractal dimension of the artefact, which is input to B, the beauty model (see below). Mandelbrot [10] described fractals as geometric objects, which are no-where differentiable and self-similar at different scales. We use the geometric interpretation based on reticular cell counting or the box counting dimension. Mandelbrot also said fractal objects have fractional dimension, D, a non-whole number, called the fractal dimension. Mathematically, D is given by the Hausdorff dimension [46]:

$$D(S) = \lim_{\epsilon \to 0} \frac{\log N_\epsilon(S)}{\log 1/\epsilon} \quad (2)$$

where S represents a set of points on a surface (i.e., coast lines, brush strokes, or in our case, source lines of code in artefact form), ϵ is the size of the measuring tool, and $N_\epsilon(S)$ is the number of objects or subcomponents covered by the measuring tool. For fractal objects, $\log N_\epsilon(S)$ will be greater than $\log 1/\epsilon$ by a fractional amount. If the tool is a uniform grid of square cells, then a straight line passes through twice as many cells if the cell length is reduced by a factor of two. A fractal object passes through more than twice as many cells. The artefact is S in Equation 2. For ϵ, we use grid sizes of 2, 3, 4, 6, 8, 12, 16, 32, 64, and 128 measured in pixels, which are the default settings of Fractop [31].

4 Beauty model

Let S be some source code, the "baseline," before treatment. Let T be a treatment such that T (S) = S' is source where S' is functionally identical to S. When T=∅, then S=S' in function as well as style. Finally, let D (S) be the baseline fractal dimension before submitting S to treatment T and D(T(S)) = D(S') is the fractal dimension after submitting S to treatment T. Thus, we define, B(S|S'), to be the beauty factor (or "beauty") of S relative to S' where B ϵ R. Namely,

$$B(S|S') = 10\ log_{10} \frac{D(S)}{D(S')} \quad (3)$$

The units are decibels. We have the following interpretations of B:

1. If B < 0, the beauty of S improves given changes implied by T.
2. If B ≥ 0, the beauty of S does not improve given changes implied by T.

When B>0, the aesthetic appeal decreases using T, in other words, too much of a good thing may work against the code, just as too little when B<0. This is a reason D by itself is not necessarily a measure of beauty. For instance, some of Pollack's masterpieces have a D 1.45 to 1.72 and outside this range representations become increasingly merely lines or un-textured surfaces [38]. When B=0, the beauty remains unchanged by T, that is, S=S'.

The model predicts positive correlations with LOC and M when T de-beautifies the code according to the basic tenets. The model further predicts negative correlations with LOC and M when T beautifies the code. In other words, decreasing B is concomitant with longer, more complicated codes and the opposite for increasing B. Second, T that randomly distorts the style in a significant way results in little or no correlation of B with LOC and/or M.

5 Experimental design

We use 114 files of the GNU Core Utilities version 8.10 [47] except we reduce each file to exactly one function or procedure and internal comments, if any, which generates a total of 1,043 files. We measure LOC and M once for each file and D and B for each treatment in Table 1. The Kolmogorov-Smirnov test [48] indicates the distributions of these values over all files are non-Gaussian and thus, we assess all correlations non-parametrically using Spearman's rho or ρ [48]. Source code use to process the files and run the experiments can be found online [49].

6 Results

In this section we give the results of experiments on the test bed of C files from the GNU Core Utilities.

6.1 LOC, M, and D

The table below gives the correlation matrix for LOC, M, and D (S).

Table 2 Correlation matrix of LOC, M, and D (S)

	LOC	M	D (S)
LOC	1.00	0.86	0.35
M	0.86	1.00	0.16
D (S)	0.35	0.16	1.00

The figure below gives the scatterplot of LOC vs. M.

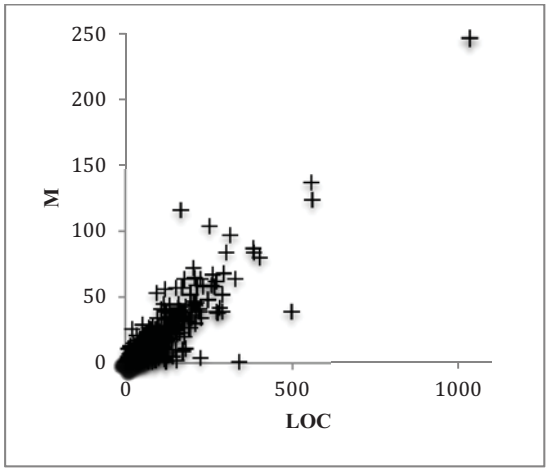


Figure 3 Scatterplot of LOC vs. M

The two figures below give scatterplots of D (S) vs. LOC and D (S) vs. M.

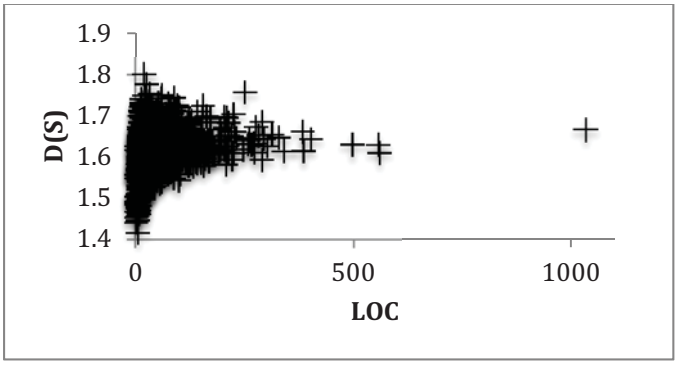


Figure 4 Scatterplot of LOC vs. D (S)

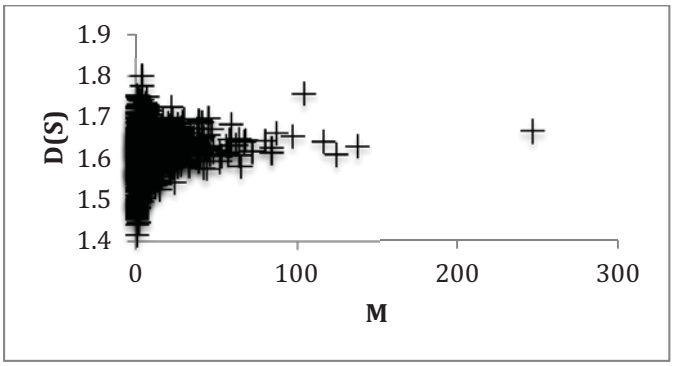


Figure 5 Scatterplot of M vs. D (S)

6.2 Beauty analysis

In this section we give results of the beauty analysis using the beauty factor, B. The table below gives the frequencies of the different beauty factors when de-beautifying and their correlations with LOC and M.

Table 3 Beauty factors for de-beautifying treatments

	Frequencies			ρ	
Treatment	B<0	B=0	B>0	B v. LOC	B v. M
Random indents N=20	25	1	1,017	-0.02	-0.03
Random indents N=40	11	0	1,032	0.03	0.02
Remove indents	991	12	40	-0.54	-0.57
De-comment	137	217	689	0.20	0.04
Non-mnemonic	103	188	752	0.41	0.45

Dividing B<0 and B≥0 as Bernoulli trials according to interpretations of the beauty model in Section 4, each treatment is statistically significant with $P<10^{-6}$.

Table 4 Beauty factors for beautifying treatments

	Frequencies			ρ	
Treatment	B<0	B=0	B>0	B v. LOC	B v. M
GNU style	812	121	110	-0.44	-0.49
K&R style	922	0	121	-0.42	-0.49
BSD style	712	0	331	-0.42	-0.47
Linux style	955	0	88	-0.45	-0.54
Re-comment	448	0	595	0.02	0.04
Mnemonic	697	255	91	-0.29	-0.28

Each treatment here is similarly statistically significant with $P<10^{-6}$. The exception is re-comment, which does not improve the aesthetic appeal with $P\approx10^{-6}$.

7 Discussion

7.1 Correlation matrix

The correlation matrix of Table 2 and the scatterplot of Figure 3 confirm that LOC and M are strongly correlated with each other as Hatton [41] had found. However, there are only weak correlations of LOC with D and M with D. Indeed, the scatterplots in Figure 4 and Figure 5 are generally flat. This suggests LOC and M are related to D but they are not proxies for D. We note furthermore that since fractals are by definition scale invariant, unlike LOC and M, the measure D is insensitive to file size. In other words, Table 2 suggests that D is a more robust measure than LOC and M.

7.2 De-beautifying treatments

Referring to Table 3, the data shows that de-beautifying treatments generally does not improve the aesthetic appeal since generally B≥0. The exception is removing indentation where B<0. However, as Coleman and Gandhi [7] found, removing indents is a contrarian indicator and thus, removing indentation does not add aesthetic appeal.

As for the correlations in Table 3, the data indicates that B vs. LOC and B vs. M have virtually no relation when randomizing indentation. This agrees with commonsense since by definition there is no pattern to random indentation. However, removal of indentation is moderately anti-correlated in both LOC and M in relation to B. To account for this, we refer to Hindle, Godfrey, and Holt [50] who showed that indentation is strongly correlated with M. In other words, higher M implies deeper nesting and consequently, more indentation to remove. Consequently, no indentation and aesthetic appeal tend to work in opposite directions, which is consistent with experience and anecdotal prescriptions in style guides [7]. Table 3 offers statistical support for this conclusion.

The aesthetic appeal of removing comments is not correlated with M, which we would expect since there is no functional code in comments. However, removing comments is weakly correlated with LOC. We don't believe this difference between LOC and M is meaningful for practical purposes. Yet we note for reference that sometimes there is code or pseudo code in comments, that is, codes that have been commented-out and remain in the source for documentation and/or legacy purposes.

Non-mnemonic refactoring and software complexity (the last row in Table 3) are moderately correlated which seemed to us initially surprising. Yet we would expect more decision points tend to involve more names (i.e., variables). In other words, there are more high frequency long names to shorten if there are more decisions; hence the code might be more aesthetically appealing if those names were more mnemonic (e.g., longer).

7.3 Beautifying treatments

Referring to Table 4, we first note that B is negatively correlated with LOC and M except when adding comments (i.e., re-comment), which we discuss below. In other words, while increasing aesthetic appeal doesn't necessarily reduce software complexity, the negative correlations of with LOC and M with GNU, K&R, BSD, Linux, and mnemonic style changes suggest beauty and software complexity are incompatible. This comports with longstanding advice of style guides; namely, there is virtue in brevity and simplicity [3].

Non-mnemonic refactoring (Table 3) and mnemonic refactoring (Table 4) are virtually opposites of one another in terms of beauty, which is what the model predicts. That there are more counts where B=0 for mnemonic suggests the analysis identified more longer names to de-beautify than shorter names to beautify. However, the difference (217 vs. 255) is not statistically significant (P≈0.92).

The other case where there is B=0 frequency count is GNU styling. It agrees with our expectations since the GNU Core Utilities is a GNU project presumably following the GNU style guide. Thus, we would expect S=S' and consequently, B=0. That this is unique among the indent styles is confirmation that beauty factors detect the GNU style where we expect to find it. Yet, approximately 78% of files do not conform to GNU style which is something needing further consideration. Perhaps the issue is a matter of degree in which B is smaller for GNU style compared to the other styles, as we would expect. Indeed, this is what the data shows in Table 5 below: the median B (in decibels) for GNU style is closest to zero than the other styles.

Table 5 Median B by style in decibels

Style	dB
GNU	-0.034
K&R	-0.096
BSD	-0.041
Linux	-0.127

A further interpretation of B might be as a quantitative style checker. While a detailed comparison of GNU, K&R, BSD and Linux styles is beyond the scope of our study, Table 5 suggests the GNU Core Utilities diverge least from the GNU style, which again is what we expect. Furthermore, that observed style in actual source code of the test bed diverges most from the Linux style and least from the BSD style seem to us consistent with general but reasonable interpretations of recommendations in the respective style manuals [43, 44, 45].

As for comments, removing them does not appear to improve source beauty (Table 3) but neither does adding them (Table 4). This finding is at odds with advice in most style guides. Yet it is consistent with model predictions. Namely, comments tend to affect code layout in a random manner, just as random indentation. From this we conclude 1) reading code and appreciating it are not the same; and 2) a more reliable way to improve aesthetic value may be through mnemonics or "self-documenting" style, which Tables 3 and 4 show are correlated in common sense directions with LOC and M.

8 Conclusions

We have shown that software complexity, namely, LOC and M, are related to beauty factors but they are also different. The question remains about readability and beauty. In manipulating comments, we identified indirect evidence that readable code and beautiful codes are not necessarily the same things. Future research needs to investigate this question. Future work might also look into the alternate interpretations of beauty factors. We only looked at the C language and at that, the GNU/Linus repository. Future research will study other languages and repositories to understand how the fractal dimension and beauty factors relate to them.

9 References

[1] Donald E. Knuth. *The Art of Computer Programming*, Vols 1-4A, Addison-Wesley, 2011

[2] Donald E. Knuth. "Computer Programming as an Art," *Communications of the ACM*, 17 (12), 1974

[3] Brian Kernighan and P.J. Plauger. *The Elements of Programming Style*, McGraw Hill, 1978

[4] Oracle, Inc. "cb – C program beautifier", https://docs.oracle.com/cd/E24457_01/html/E22003/cb.1.html, retrieved 28 April 2016

[5] Brian Kernighan and Dennis Ritchie. *The C Programming Language*, Prentice Hall, 1978

[6] Carlo Wood, Joseph Arceneaux, Jim Kingdon, and David Ingamells. "Indent", edition 2.2.10, for Indent Version 2.2.10, 23 July 2008, https://www.gnu.org/software/indent/manual/indent.pdf, retrieved 28 April 2016

[7] Ron Coleman and Pritesh Gandhi. "Fractal Analysis of Good Programming Style", *Proceedings Second International Conference on Computer Science & Engineering*, Dubai, UAE, 28-29 Aug 2015

[8] Heinz-Otto Peltgen and P.H. Richter. *The Beauty of Fractals*, Springer, 1986

[9] Jennifer Ouellette. "Pollock's Fractals," *Discover Magazine*, 1 November 2001.

[10] Benoît Mandelbrot. *Fractal Geometry of Nature*, Freeman, 1982

[11] Google, Inc. "Google style guide", https://github.com/google/styleguide, retrieved 23 November 2015

[12] Twitter, Inc. "Effective Scala," http://twitter.github.io/effectivescala/, retrieved 23 November 2015

[13] NOAA. "General Software Development Standards and Guidelines," NOAA, National Weather Service Office of Hydrologic Development. NOAA, 2008

[14] Steve Oulline. *C Elements of Style: The Programmer's Style Manual for Elegant C and C++ Programs*, M&T, 1992

[15] Trevor Misfeldt, Gregory Bumgardner, Andrew Gray, and Luo Xiaoping, *The Elements of C++ Style*, Cambridge, 2004

[16] Allan Vermeulen and Scott W. Ambler. *The Elements of Java Style*, Cambridge, 2000

[17] Vincent Y. Shen, Tze-Jie Yu, Stephen M. Thebaut, and Lorri R. Paulsen. "Identifying Error-Prone Software--An Empirical Study," *IEEE Transactions on Software Engineering*, Volume 11, Issue 4, April 1985, pages 317-324.

[18] William T. Ward. "Software Defect Prevention Using McCabe's Complexity Metric," *Hewlett-Packard Journal*, 1989, p64-68

[19] Norman E. Fenton and Martin Nell. "A critique of software defect prediction models," *IEEE Transactions on Software Engineering*, 1999, Volume 25, Issue 5, Sept 1999, p675-689

[20] Andreas Zeller. "Bugs Reside in Complex Code," from Myths in Software Engineering, http://www.slideshare.net/andreas.zeller/myths-in-software-engineering, retrieved April 28, 2016

[21] Rudolf Flesh. "A new readability yardstick," *Journal of Applied Psychology*, 1948, 32 (3), p221-233.

[22] Daryl Posnett, Abram Hindle, Premkumar Devanbu. "A simpler model of software readability," *Proceedings of the 8th Working Conference on Mining Software Repositories* (MSR '11), 2011, p73-82

[23] Donna Teague and Raymond Lister. "Programming: reading, writing and reversing," *ITiCSE '14 Proceedings of the 2014 conference on Innovation & technology in computer science education*, 2014, p285-290

[24] Ben Fry. *Distellamap*, 2003 Museum of Modern Art, NYC, NY, United States

[25] Ben Fry. *Visualizing Data*, 2008 O'Reilly.

[26] Andy Oram and Greg Wilson (eds.). *Beautiful Code: Leading Programmers Explain How They Think*, O'Reilly, 2007

[27] Peter Kokol. "Searching for fractal structure in computer programs," *SIGPLAN notices*, 29 (1), 1994

[28] Peter Kokol, and Janez Brest. "Fractal structure of random programs," *SIGPLAN notices*, 33 (6), 1998

[29] Peter Kokol, Janez Brest, and Viljem Zumer. "Long-range correlations in computer programs," *Cybernetics and systems*, 28 (1), 43-57, 1997

[30] Peter Kokol, Vili Podgorelec, and Janez Brest. "A wishful complexity metric," in H. Combes (ed.), *FESMA*, p. 235-246

[31] David Cornforth, Herbert Jelinek, and Leo Peichl. "Fractop: A Tool for Automated Biological Image Classification," *Proc. Sixth Australia-Japan Joint Workshop on Intelligent and Evolutionary Systems*, 2002, p1-8

[32] Ron Coleman. "Fractal Analysis of Pathfinding Aesthetics," *International Journal of Simulation Modeling*, Volume 7, Number 2, 2008

[33] Ron Coleman. "Fractal Analysis of Stealthy Pathfinding," *International Journal of Computer Games Technology*, 2009.

[34] Ron Coleman. "Long-Memory of Pathfinding Aesthetics," *International Journal of Computer Games Technology*, 2009.

[35] Peter Gerl, Carola Schönlieb, and Kung Chieh Wang. "The Use of Fractal Dimension in Arts Analysis," *Harmonic and Fractal Image Analysis*, 2004, p70-73

[36] Jim Coddington, John Elton, Daniel Rockmore, and Yang Wang. "Multifractal analysis and authentication of Jackson Pollock paintings," *Proc. Computer Image Analysis in the Study of Art* (SPIE 6810), 2008, doi: 10.1117/12.765015

[37] R.P. Taylor, R. Guzman, T.P. Martin, G.R.D. Hall, A.P. Micolich, D. Jonas, B.C. Scannell, M.S. Fairbanks, C.A. Marlow. "Authenticating Pollock paintings using fractal geometry," *Pattern Recognition Letters*, Volume 28, Issue 6, 2007, p695-702

[38] Richard P. Taylor, Adam P. Micolich, A., and David Jonas. "Fractal analysis of Pollock's drip paintings," *Nature* 399, 422 (3 June 1999), doi:10.1038/20833

[39] Mohammad Irfan and David Stork. "Multiple visual features for the computer authentication of Jackson Pollock's drip paintings: Beyond box counting and fractals," *Proc. SPIE 7251*, Image Processing: Machine Vision Applications II, 72510Q, 2009

[40] De Tran-Cao, Ghislain Lévesque, and Jean-Guy Meunier. "A Field Study of Software Functional Complexity Measurement," *Proc. 14th International Workshop on Software Measurement*, 2004

[41] Thomas McCabe. "A Complexity Measure," *IEEE Transactions on Software Engineering*, SE-2 (4), 1976

[42] Les Hatton. "The role of empiricism in improving the reliability of future software," *Proc. Practice and Research Techniques*, 2008. TAIC PART '08. Testing: Academic & Industrial Conference

[43] Stallman, R. (2015). *GNU Coding Standards*, Samurai Media Limited, 2015

[44] FreeBSD. "FreeBSD Kernel Developer's Manual," https://www.freebsd.org/cgi/man.cgi?query=style&sektion=9, retrieved 7 May 2016

[45] Linus Torvalds. "Linux Kernel Coding Style," http://slurm.schedmd.com/coding_style.pdf, retrieved 7 May

[46] Benoît Mandelbrot. "How long is the coast of Britain? Statistical self-similarity and fractional dimension," *Science*, 156 (3775), 1967, 636-638

[47] Free Software Foundation. (2015, November 23). Coreutils – GNU core utilities. Retrieved November 23, 2015, from Free Software Foundation: http://www.gnu.org/software/coreutils/coreutils.html

[48] W.J. Conover. *Practical Non-Parametric Statistics*, Wiley, 1999

[49] Ron Coleman. https://github.com/roncoleman125/Pretty, retrieved November 23, 2015

[50] Abram Hindle, Michael W. Godfrey, and Richard C. Holt. "Reading beside the lines: Indentation as a proxy for complexity," *Proc. 16th IEEE International Conference on Program Comprehension*, ICPC 2008, IEEE, p133-142

Comparing An Object Oriented Runtime Complexity Metric To Depth First Search Complexity with Mobile Agents in a Mobile Autonomous Environment

J. McKinney Young[1], L. Etzkorn[2]
[1,2]Department of Computer Science, University of Alabama in Huntsville, Huntsville, Alabama, USA
300 Technology Hall
University of Alabama at Huntsville
Huntsville, AL 35899
[1]julienmckinneyyoung@mac.com, [2]EtzkorL@uah.edu

Abstract - *Software complexity metrics provide a way to describe and predict the resources needed to maintain and update code. Complexity metrics derived statically from source code describe the complexity of the software's code. Complexity metrics derived at run-time describe the complexity of the software's behavior. In this paper, we use an object oriented runtime complexity metric that describes the group complexity of mobile agents working atop mobile autonomous platforms. The agent/platform pairs use the Depth First Search algorithm to walk a graph in order to search for colored balls. We extend the experiment by studying the mobile agents/mobile platform's behavior complexity in three scenarios of differing difficulty. We then compare the run-time derived complexity metrics with the algorithms theoretical worst-case complexity estimation.*

Keywords: Software Complexity, pathfinding, cellular automata

1. Introduction

Software complexity metrics traditionally have been computed statically using the software's source code. This provided a descriptive means of evaluating the entire collection of code. Recently, the capture of dynamic complexity metrics of code during execution has allowed study of software behavior at runtime [3,4,5,6,7,8,9]. This view into the actual behavior of the executing code allows the software professional to better understand and predict the software's actions. Runtime complexity metrics are especially useful with object oriented code because dynamic behavior such as inheritance and polymorphism can be hard to predict from a static source code-only review [9].

In this paper, we use an object oriented runtime complexity metric that measures the group complexity of mobile agents as they work atop mobile autonomous platforms to complete a basic walk and search task in a simulated environment. The compiled complexity data highlights the surprising ways in which actual complexity metrics gathered during execution in real world scenarios differs from theoretical static code complexity estimations.

2. Related Work

Software professionals have long noticed the relationship between a software applications code complexity and the resources required to adequately test and maintain it. The higher the complexity, the higher the level of resources required. T.J. McCabe addressed the issue in 1976 with his article, "A Complexity Measure." In this article, McCabe presents a way to measure code complexity statically by counting all possible paths of execution that could potentially be exercised. McCabe specifies that complexity depends not on the size of the program but only on the decision structure of a program. [10]

Another method of measuring code behavior was proposed by Chidamer and Kemerer in 1994 in [1]. They proposed that coupling between objects (CBO) was a useful object-oriented metric. Coupling is defined as the manner and degree of relationship between software modules - when methods in one class use methods or instance variables defined in another. They found that the higher the amount of coupling in the code, the higher the amount of complexity is present [1].

In 2005, Mitchell and Power extended Chidamer and Kemerer's metric, CBO in [11]. They applied it at run-time to examine coupling behavior of the actual running code. They believed that the static CBO measure did not provide an exact look at what really happened when the code actually executed. "...CBO cannot capture all the dimensions of object-oriented coupling because features of object-oriented programming such as poly-morphism, dynamic binding and inheritance render CBO imprecise in evaluating the run-time behavior of an application." [11] Their studies showed that objects from the same class can behave differently at runtime "from the point of view of coupling" than could be described from a static analysis of the source code [11].

Five years later, Mathur and Keen proposed a different metric to study run-time complexity. Just as Mitchell and Power extended Chidamer and Kemerer's CBO to the run-

time environment, Mathur and Keen extended McCabe's idea of cyclomatic complexity to the run-time boundary. They introduced a metric that is based on the number of decision points evaluated by the running code [9]. The authors make the distinction between "potential" complexity - the code at compile time - and "actual" complexity - the code that actually executed. They calculated the runtime complexity of objects by counting the number of decision points that were accessed at runtime. Selection structures like if...then, if..else..then, case statements and do...while, for...while structures added to the count as executed per object. Each decision point was counted once - iterative calls were not counted. The authors, like McCabe, write that the decision points alter the control flow of the program and can affect the complexity of the running code. The authors then compared the count-generated metric of complexity against the complexity ratings given by a panel of experienced programmers who analyzed the subject source code and static complexity measures derived from the source code. They, like Chidamer and Kemerer, found that the runtime metric measured a "different aspect of complexity" than did static complexity metrics [9].

Desouky, after Mathur and Keen, examined runtime complexity expressed by decision points. Desouky extended Mathur and Keen's decision-count metric to include iterative decision calls [5]. The author studied the metric using an open source application Rhino 1.7R4. The results were compared to bug reports because software quality is inversely related to the number of bugs found [5]. A code module that has a very low number of bugs found is considered to be of a higher quality and contain lower complexity than a module of that is found to contain a large number of bugs. The study found that the derived complexity values correlated with the number of bugs found in the code modules under test [5].

3. Background

In [6] and [7], Keen takes McCabe's idea of complexity as a function of decision structure and combines it with the idea that run-time metrics could give a more accurate view of complexity as represented by the running code. This represents a shift in focus away from static code complexity to dynamic behavior complexity. Keen introduced the metric Keenint^{RM} to describe the amount of complexity in a program as it executed. It is a descriptive metric that allows the software professional to compare different approaches and implementations for accomplishing the same task. The better approach is the approach that accomplishes the task with the least amount of complexity [6].

Keenint$_{RM}$ = 1/complexity for some particular task [6].

The metric is very similar to a runtime version of McCabe's cyclomatic complexity [6]. It is the sum of decision points encountered by the running code plus one is added at every method invocation even if no decision structures are encountered. This represents the idea that some level of complexity is represented by the activity of processing the method even without the greater work of processing a control structure.

In [6], Keen compares the behavior of two programming approaches by implementing the approaches as agents - static-vs-mobile - running on simple mobile systems fitted with infrared sensors. The mobile systems had a task to perform - to move about a grid in search of three colored balls in a particular order. The "agents" directed the mobile systems and processed the "visual" (infrared) data that the mobile systems perceived.

In the Three-Mobile System Scenario, which compared the activity of three mobile systems with static agents and then three mobile systems with mobile agents, starting at the same start points searching for the same balls in the same location, the data suggested that the mobile agent approach showed lower complexity than the static agent approach [6]. Keen's work also demonstrated that the mobile agent approach showed greater resiliency in the face of obstacles than the static agent approach. When one mobile system got stuck in a corner and couldn't turn enough to remove the wall/obstacle from its view, the mobile agent resident on the "stuck" mobile system was able to jump to another mobile system (that had already completed its agent's task) and continue searching for it's colored ball. In the static agent approach, both the "stuck" mobile system and its resident static agent where unable to complete their task.

We implemented Keen's experiment in a simulation. The simulation was validated against the original data. The mobile platforms use the DFS pathfinding algorithm to walk the graph. In this paper, we discuss our findings when we used the simulation to compare the run-time complexity of the mobile agents/mobile platform teams walking a graph with DFS's Order of Complexity. The comparison provides additional data to describe the aspects of complexity that the Keenint^{RM} metric captures.

4. Case Study

4.1. Simulation Description

The simulation is implemented in the form of a cellular automaton (CA) using Microsoft Access. The researcher may observe the progress of the running code (written in Visual Basic) through the graphical user interface (GUI). The persistent datafiles in which the rules, the geography, and the intermediate and final results are stored are used to derive the results of each run.

The formal definition of a CA is expressed by four things: the array dimensionality d, the set of states (the number of cells) S, the neighborhood vector (the definition of what is a neighborhood) N, and the local rule (how local neighbor states are evaluated) f [2].

CAi = (d, S, N, f).

At time quanta zero, an initial start state is assigned to each cell. The collection of all cells' states is called the

configuration of the cellular automaton at that time. The passing of each time quanta causes the states of all the cells to change (according to the cellular automaton's rule) and a new configuration describes the cellular automaton at that instant in time.

The CA expresses a 2-dimensional array(d) representing geographic direction (*x,y*). The simulation models mobile systems that are capable of moving only "wheels on the ground." There is no vertical component (*z*) to their movement. The cells (S) of the automaton comprise a 7x7 grid which has a total of 49 possible location states. The neighborhood vector (N) is made up of 4 possible cells at the North, East, South, West sides of the cell under analysis. The array of local rules are as follows:

Rules of CA behavior

1. There is a central clock.
2. Time only flows in one direction - forward.

Rules of Mobile System Behavior

1. Mobile Systems can only do one activity at each time tick:
   a. Start
   b. GoToSleep
   c. FindObject
   d. Walk
   e. Turn
   f. Lose Agent
   g. Receive Agent
2. Mobile Systems can only turn 90 degrees in one time tick.
3. A mobile system cannot revisit a node that it has already visited - unless the system is in "Fallback" mode and the already-visited node is the just-prior node.
4. Only one mobile system at a time may occupy a node.
5. A mobile system will go to "Fallback" state on the 4th time tick after it has turned a complete circle (4 turns in 4 consecutive time ticks) and has not been able to move out of its current node.
6. A mobile system will "go to sleep" on the 4th time tick after it has turned a complete circle in the "Fallback" mode (4 turns in 4 consecutive time ticks) and has not been able to move out of its current node to its "origin" node (the node it occupied right before it moved to its current node.)
7. Each Mobile System starts the simulation with one assigned Agent.
8. Mobile Systems possess the attribute of "handedness" (left or right) - when a mobile system turns, it will turn in the direction as specified by its "handedness" setting.
9. Mobile Systems "find" a ball by stepping into a cell that contains a ball.
10. Mobile Systems walk a self-selected path using the DFS algorithm.

Rules of Agent Behavior

1. Agents can only do one activity at each time tick:
   a. DoNothing
   b. GoToSleep
   c. FindObject
   d. Jump
2. When a mobile system dies so also does its agent.
3. An agent is assigned a particular colored ball (red, blue, green) to find.
4. It can only "find" that one ball.
5. The agent "checks" if the ball is in the cell during the same time tick that the mobile system has moved into a new cell.
6. After the agent finds its ball, it goes to sleep.
7. A different agent who is still active (has not found it's ball) can jump to a mobile system with a "sleeping" agent.
8. A different agent who is still active (has not found it's ball) can jump to a mobile system without an agent.
9. A "jumping" agent may only leave one mobile system and arrive at another mobile system in one time tick. (The mobile system will "wake up" in the next time tick.)

4.2. Depth First Search (DFS) Description

DFS is an algorithm for traversing a graph. It was described in the 19th century by Charles Pierre Trémaux, a French mathmematician [12]. In the algorithm, the search begins at some arbitrary node of the graph and "walks" or explores down as far as possible along every branch before backtracking. When an obstacle is encountered or the branch ends, one "falls back" to the next higher node and selects another node/branch down which to explore. In other words, the algorithm visits children nodes before it visits sibling nodes [12]. It's order of complexity is defined as $\Theta|V|+|E|)$, where V is the number of vertices (or nodes) in the graph, and E is the number of edges. As an actor "walks" a graph using the DFS algorithm, the DFS order of complexity represents the fact that the actor must make decisions at each node about where to go next.

4.3. Scenario Design Description

The mobile agent/mobile platform simulation is modeled after a real-world case study using mobile agents running atop real robots who walked a graph taped on a floor in a real building. The real-world scenario forced the agent/robot teams to interact with an uncertain environment that included each other, temperature and changing light conditions based on the position of the sun. The advantages of this approach are that the agent/robot pairs interacted with and potentially overcame unexpected conditions just like software and hardware must do once they are deployed on real-time, real-world systems. Metrics collected in this environment are more realistically descriptive to what would be experienced at actual deployment. The simulation includes the causal factors of the real-life case study. It allows expansion of the study to include questions that might prove too arduous and time-consuming to be studied with the real robots.

In this case study, we wanted to examine the Keenint^{RM} metric as it captured simple graph traversal activity in order to better understand the aspects of complexity that the metric describes. Since the agent/platform pairs represent actual entities who must move "wheels on the ground" in linear time, our hypothesis was that the agent/platform pairs would

experience more complexity than what a theoretical estimation would describe.

For that purpose, we compare the values of Keenint^{RM} collected for the group of agent/platform pairs while traversing a graph with the theoretical values proposed by the DFS algorithms order of complexity for the same traversed graph.

The three agent/platform pairs walk autonomously the graph searching for colored balls. The presence of three pairs interjects uncertainty with regards to path openness. Even though the mobile agents do check if the mobile platform has "found" the appropriate ball, that decision structure is not counted for this experiment. In the three mobile system/ mobile platform teams, the start states are described below.

Scenario 1 - Length: 120 time quanta
B1: red, at 2,2
B2: green, at 4,5
B3: blue, 1,6
R1, agent1 - read ball, 7,1, left-handed
R2, agent2 - green ball, 7,2, right-handed
R3, agent3 - blue ball, 1,7, right-handed

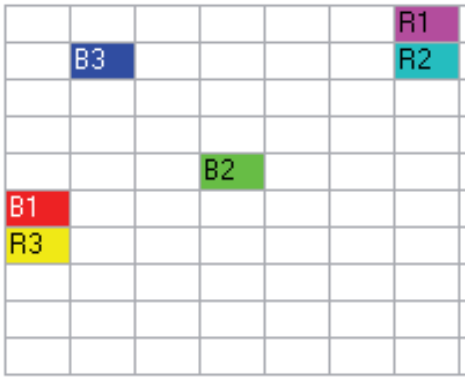


Figure 1. Scenario 1 Start

Scenario 2 - Length: 120 time quanta
B1: red, at 2,4
B2: green, at 5,3
B3: blue, 6,6
R1, agent1 - read ball, 7,1, left-handed
R2, agent2 - green ball, 7,2, right-handed
R3, agent3 - blue ball, 1,7, right-handed

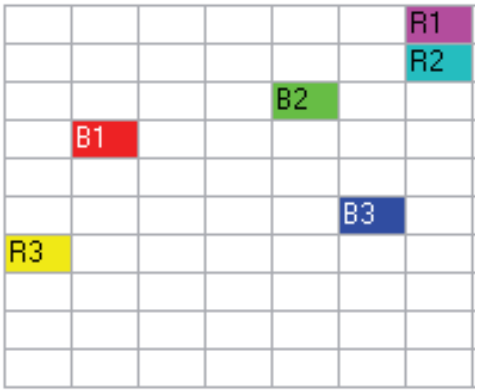


Figure 2. Scenario 2 Start

Scenario 3 - Length: 120 time quanta
B1: red, at 1,4
B2: green, at 1,6
B3: blue, 7,3
R1, agent1 - read ball, 7,1, left-handed
R2, agent2 - green ball, 7,2, right-handed
R3, agent3 - blue ball, 1,7, right-handed

Figure 3. Scenario 3 Start

Scen-ario	Θ|V| +|E|	Keenint^{RM}	Nodes Visited	Fall-back Nodes	Balls Found
1	161	433,476	81	9	3
2	129	280,562	65	3	3
3	61	34,063	31	1	3

Table 1: Keenint$_{RM}$ Aggregate Data (120 sec run) by Scenario

As Table 1 describes, the placement of the balls and the start location of the mobile platforms in Scenario 1 makes this scenario the most arduous for three agent/platform pairs. They visit 81 nodes in the 120 sec experiment. (And then revisit 9 of those nodes in "fallback" mode.) Comparing Scenario 2 data with Scenario 1 data, one sees a 24% increase in the number of nodes visited from Scenario 2's 65 nodes to Scenario 1's 81 nodes but there is a 54.5% increase in the Keenint^{RM} value. The difference between Scenario 2's activity and Scenario 3's activity is more dramatic. The number of nodes visited more than doubles from Scenario 2 to Scenario 3 but the Keenint^{RM} value increases by a factor of 7. In comparison, the Order of Complexity given by the DFS algorithm shows only an arithmetic increase between the three scenarios related to the number of nodes and edges visited.

5. Conclusion and Future Work

Implementing the DFS algorithm as a path finding strategy in a simulation where the moving objects are specifically designed to mimic real, autonomous, mobile platforms highlighted several critical differences between how the algorithm models movement and time and how real objects experience movement and time as described by Keenint^{RM}. The Order of Complexity given by the DFS algorithm shows only an arithmetic increase between the three scenarios related to the number of nodes and edges visited while the Keenint^{RM} values show a greater than arithmetic growth between the scenarios. In Scenario 1, the agent/platform pairs visit a total of 90 nodes in the 120 second experiment (81 nodes in normal mode and 9 in "fallback" mode.) Comparing Scenario 2 activity with Scenario 1 activity, there is is a 24% increase from Scenario 2's 65 nodes to Scenario 1's 81 nodes visited in normal mode. However, there is a 54.5% increase in the Keenint^{RM} value from Scenario 2 to Scenario 1. Comparing Scenario 3 with Scenario 1 shows that the agent/platform pairs visited almost 2 1/2 times more nodes in "normal" mode in Scenario 1 then they did in Scenario 3. Keenint^{RM} , however, shows an almost 12-fold increase between Scenario 3 and Scenario 1.

What is the difference between what the DFS order of complexity represents and what the Keenint^{RM} captures? First and foremost, in the DFS algorithm, an actor will "fallback" to its prior node instantaneously and effortlessly. Revisiting a "fallback" node does not add any complexity to the DFS algorithm's estimation. As the algorithm is written, an actor who is blocked from moving forward to the next unvisited node will drop out of the "next step" subroutine and return to the calling routine with all position data either updated or still available. DFS implemented recursively with a real mobile platform has many more actions to complete before it is back at it's prior location. First, the real object does not fly. It must move wheels on the ground. And it must be able to "see" where it's going. That means that platform must turn before it can "fallback" to its prior node. It can only turn 90 degrees in a time interval so it may take several time intervals before it is facing back the way it came. Then, because it takes the whole time interval to turn, it "walks" back to the prior node in the next time interval. Also, one must keep in mind that the platform has direction. It must then turn in order to evaluate possible "next steps" from the current-on-the-way-to-the-"fallback" node. At each time interval during the "fallback" exercise, the platform makes decisions on how to move (turn or walk) and what is the correct node to walk to (it can't walk to just any open node - it's in fallback mode, so it can walk only to its preceding node.) Once, the platform does determine that it is facing the proper fallback node, it then must decide if the node is empty. These determinations are represented by decision structures in the running code that are NOT part of the DFS algorithm. As the data in Table 1 shows, each node visited (or revisited), regardless of mode, adds to the complexity of the system.

Another logical difference between theoretical DFS and implemented DFS is that when the platform turns to "fallback", it still has the capability to evaluate other open nodes. What if the platform must turn 180 degrees in order to "fallback" to its prior node? That would take it two time ticks and two turns. Once the platform turns 90 degrees and is facing an adjacent node (that was evaluated as occupied several time ticks ago - and thus ineligible for moving to), what if that adjacent node (which is not the fallback node) is now open? In real life, if an actor were traversing a graph, it may be a more effective rule for the mobile platform to move into the now open, unvisited, adjacent node and skip the remainder of the "fallback" exercise. As DFS does not include interim steps between the decision to fallback and the actual fallback arrival at the calling routine, there is no mechanism to describe interim steps and how best to handle them in terms of the traversal exercise.

The complexity metric Keenint^{RM} in implemented DFS captures more decision making with regards to WHY a node may be ineligible to visit than the theoretical DFS algorithm does. In theoretical DFS, each node is checked once to see if it is open to be visited. In implemented DFS, however, an additional nested case statement is added to the algorithm to handle node evaluation when the platform is in "fallback" mode rather than "normal" mode. The additional condition checks represent decisions made by the platforms executing code and add to the complexity value. The case study shows that intermittently accessing different levels of nested control structures will affect the complexity metric in a way that is not represented by theoretical DFS's time complexity metric of Θ |V| + |E|). Theoretical DFS time complexity value should increase at an arithmetic rate with the addition of nodes and edges to the walked path. The Keenint^{RM} metric shows a greater than arithmetic increase in complexity with the addition of nodes and edges to the path as shown between Scenario 2 and Scenario 1.

Additional study could further discriminate between "normal" mode complexity values and "fallback" mode complexity values. An interesting question would be, is the greater than arithmetic growth in Keenint^{RM} due solely to "fallback" mode behavior? Or does the metric represent additional actual traversal exercise complexity that is not captured in the theoretical DFS estimation?

Another follow-on project to this paper may be to study how the complexity metric, Keenint^{RM}, compares to another pathfinding order of complexity such as A*. A* is commonly used in mobile platform pathfinding exercises. It may offer a better fit between the theoretical algorithm and the behavior of actual executed code for a real-life autonomous mobile platform.

6. References

[1] S. Chidamer, C. Kemerer, "A metrics suite for object oriented design,' IEEE Transactions on Software Engineering, Vol. 20, No. 6, June 1994, 476-493.

[2] Brooks, Richard R., Orr, Nathan, 2002. "A Model for Mobile Code Using Interacting Automata," *IEEE Transactions on Mobile Computing*, Vol. 1, No. 4, October-December 2002. pp. 313-325.

[3] A. Desouky, L.H. Etzkorn, "Object oriented cohesion metrics: a qualitative empirical analysis of runtime behavior," In Proceedings of the 52nd ACM Annual Southeast Regional Conference (ACM SE '14). ACM, New York, NY, USA, Article 58, 5 pages.

[4] A. Desouky, M. Beard, L.H. Etzkorn, A qualitative analysis of code clones and object oriented runtime complexity, 2014 IEEE International Conference for Convergence of Technology (I2CT 2014), Pune, India, April 6-8, 2014.

[5] A. Desouky, L.H. Etzkorn, An object oriented runtime complexity metric based on iterative decision points. The 2013 International Conference on Software Engineering Research and Practice (SERP'13), Las Vegas, NV, July 22-25, 2013. 468-472.

[6] K. Keen,. Measuring and Comparing Group Intelligence of Mobile and Intelligent Agents on a Mobile Robotics Platform.

[7] K. Keen, R. Mathur, L. H. Etzkorn, L., "Towards a measure of software intelligence employing a runtime complexity metric," Software Engineering and Applications, SEA 2009*,* November, 2009.

[8] R. Mathur, K. Keen. L.H. Etzkorn, "Towards a measure of object oriented runtime cohesion based on number of instance variable accesses," In Proceedings of the 49th Annual Southeast Regional Conference (ACM SE '11). ACM, New York, NY, USA, 255-257.

[9] R. Mathur, K. Keen. L.H. Etzkorn, "Towards an object oriented complexity metric at the runtime boundary based on decision points in code," In Proceedings of 48th Annual Southeast Regional Conference (ACM SE '10). ACM, New York, NY, USA, Article 77, 5 page.

[10] T.J. McCabe, "A complexity measure," IEEE Transactions on Software Engineering, Vol. SE-2, No. 4, December 1976, 308–320.

[11] A. Mitchell, J.F. Power, " Using object-level run-time metrics to study coupling between objects," In Proceedings of the 2005 ACM Symposium on Applied Computing (SAC'05), ACM, New York, NY, USA, 1456-1462.

[12] Wikipedia, https://en.wikipedia.org/wiki/Depth-first_search, last visited: March 14, 2016.

Principles of Continuous Integration (CI) in Practice

Kaushik Bhalerao and Ning Chen
Department of Computer Science
California State University, Fullerton
{kaushikbhalerao, nchen}@fullerton.edu

Abstract - Continuous Integration (CI), an extreme programing practice, is popular in the modern software development community. This practice promotes the continuous delivery of quality product, effective change management, early defect detection and correction that lead to cost reduction and promises on time delivery. Implementation of CI in the real world, however, is not free of hurdles. It is not unusual for smaller organizations to struggle with paying high CI maintenance and implementation cost. One strategy to make CI small-organization-friendlier is to fine-tune the original principles. In this work, we propose a set of improved CI principles that aim at low-cost CI maintainability.
Keywords: software engineering, continuous integration principles, open source tools, software configuration management, CI training

I. INTRODUCTION

Nowadays, almost all software projects become huge in terms of its size and complexity as a response to the market's demands on features and quality. As a result, the software industry turned to Agile software development methodologies that promise improved software quality, early defect detection and correction, and on time delivery, etc. Large software products have many components that need to be developed, tested and maintained by many teams and this kind of coordination cannot be achieved without the use of automation [1]. Continuous Integration (CI) is one of the key aspects of Agile software development methodologies that promotes "a fully automated and reproducible build, including testing, that runs many times a day. This allows each developer to integrate their work daily, thus reducing future integration problems [2]."

II. RELATED WORK

Although CI, in particular, refers to special CI tools, for example, CruiseControl, Hudson/Jenkins, BuildBot, that build and test pieces of software component in their integrated form, the term itself, in general, represents the whole automated software engineering process. This CI process involves other automated tools in addition to specific CI tools alone. The automated tools fall into three categories as follows:

1. Software configuration management (SCM), 2. Issue tracking, and 3. Continuous

integration. SCM tools are used to maintain software change history. Examples are Concurrent Versions System (CVS), Revision Control System (RCS), Subversion, Perforce and ClearCase. Issue tracking tools are software products that manage and maintain lists of issues. For example, Bugzilla is a web-based general-purpose bug tracking and testing tool originally developed and used by the Mozilla project [3]. CI tools promote the idea of integration early and often to avoid the so-called integration hell [4]. One popular open-source CI tool is Jenkins [5].

The CI process, sometimes called CI system, or simply CI, is adopted as part of extreme programing. In CI developers merge all their working copy to the centralized repository several times a day and expect system should automatically build and test the code. The original/traditional principles of CI are [6]:

1. Maintain a code repository
2. Automate the build
3. Make the build self-testing
4. Everyone commits to the baseline every day
5. Every commit (to baseline) should be built
6. Keep the build fast
7. Test in a clone of the production environment
8. Make it easy to get the latest deliverables
9. Everyone can see the results of the latest build
10. Automate deployment

III. ANALYSIS of CI PRINCIPLES

Although the traditional principles of CI are easy to comprehend, in a real-world practice, it is hard to adopt every principle correctly.

After studying every principle and analyzing the practices that enforce these principles, we first come up with the pros and cons observations and then provide our recommendations. All our recommendations aim to improve maintainability as well as to increase robustness of the CI infrastructure.

Principle 1: Maintain a code repository: This principle says that the build should be done using latest checkouts without dependencies. Avoid Excessive branching i.e. code should be in mainline trunk and considered as current version

Practice in reality: Even very big organizations with plenty of resources, maintain only one *repository* for automatic builds, it is difficult to maintain all changes in one branch. As a result, the organization ends up maintaining multiple branches in one repository, which, in turns, increases the complexity and maintainability.

Disadvantage: We can't proceed with red builds from even one branch. During the time one team is fixing its build, we cause delay for other teams. Even with green builds we may end up delivering Releases that are unacceptable to customers. Customers may demand old build to be fixed first and discard new builds even if it is a successful build.

Recommendation: Branching is inevitable therefor we need a tool to support this kind of management. The challenge for branching is "merge-hell" where development and release code try to synchronize. To alleviate the branching proliferation in one code repository, we recommend a short release cycle. It will decrease the wait time and reduce the chances of unacceptable release.

Principle 2: Automate Build and Deployment: This principle dictates that

single command should build the software and initiate next command to deploy on production environment or designated environment with creation of necessary artifacts (Documents and distribution media).

Practice Reality: Once a build is complete, system should deploy required artifacts to the designated network location. Nevertheless, when a change in the environment occurs, people need to get involved. Does this count as a clear automation? We have some tools (for example, Jenkins) which build software for different platforms but we need to manually place and deploy that software to designated platform. If we have different platforms and for that we have different build configurations, then this again leads to branching and the result is a complex infrastructure where errors are common and involves manual intervention. Deployment is not just placing files, but checking if the environments is ready for deployment or not.

Recommendation: Make sure the build system can handle all the variations for the same build processes and their shared components independent of whether the build system is aware of target machine requirements. Confirm that build system handles orchestration of deployment and testing in a visible and simple manner. Keeping things sorted and manageable is the key to smooth automation.

Principle 3: Make the build self-testing: This principle demands clear automation in testing. Once software is built, it should run all the tests and report the test result.

Practice Reality: Build runs at different stages and it is impractical to run all types of tests for every test. Many tests require deployment on production.

Disadvantage: Test results are not useful when we run all tests on one build. This build may respond differently depending on the production environment.

Recommendation: 1) Plan the prioritize testing and perform particular tests according to a particular environment. 2) Maintain different configurations for different tests (e.g. UI, Basic tests, etc.). The build engine should provide separate environments and allow corresponding management.

Principle 4: Fast build with most recent changes:

This principle covers the practice of CI. The developers should check in their changes daily. As a result, the build should run every day to root the defects early and one can solve the conflicts as soon as they are found. The changes should be transparent to the respective team. This principle revolves around the concept of rapid feedback.

Practice Reality:

We want to run every test on a production-like environment. This makes build-to-load time-consuming. The task of maintaining environment is complex when you have different entry attributes for running the build and test. We can run easy tests frequently and quickly. Nevertheless, tests that are useful or needed for teams to proceed are much harder since they need production-like environment.

Disadvantage: We need automation to achieve this principle and for that we need to invest in scripting which will increase the cost when the environment changes happen.

Recommendation: Always check whether the selected automated system is easy to setup and highly customizable for automation and it can quickly adapt to the changes in the infrastructure.

Principle 5: Testing in clone environment: This principle focuses on maintaining quality of the software from early stage of development. CI should able to test code in a pre-production environment which should be scalable.

Practice Reality: Most organizations maintain a pre-production environment and it adds some extra cost. Although we do not need exactly the same hardware or realistic load conditions to simulate a pre-production environment, maintaining a prep-production environment does require manual effort and resources.

Disadvantage: Cost and efforts with no 100% promise of success in production.

Recommendation: It is not possible to create an exact production environment for development and testing lifecycle. CI should able to support the deployment of complex environments easily. Make sure you are ready to create a VM close to production on the go without putting too much effort. Plan ahead and make this ability ready.

Principle 6: Make it easy to get the latest deliverable: Build should be easily and quickly available to stakeholders and QA teams. Early testing should be done on latest builds.

Practice Reality: In modern development this is done easily. Builds can be easily pushed to the artifact repository. Nonetheless, when a build fails, everyone stops and fixes that build. Red flags bring the entire development team to a screeching hold [7], [8].

Recommendation: When we encounter a build fail, there should be a synchronization to fix that build while continuing to development. Making deliverable easy to get will not solve the problem of "development pause." Instead, we need strong parallel development and testing strategy for higher productivity.

Principle 6: Build transparency: A build failed should be trackable enough so that we can point out where, who and when the change is made. Track should be accessible to the respective teams.

Practice Reality: Most complex and important tests are done manually. The manual tests make that build out of the automated feedback loop. The QA team responsible for testing, then need to notify the result to the development team

Recommendation: Try to collaborate separate testing with the CI loop, even manual testing should be reported back to CI regarding the result. This will allow the development team to get feedback easily from a single point of contact

IV. PROPOSED CI PRINCIPLE MODIFICATIONS

We summarize our proposed CI principle modifications in the following table.

Traditional Vs. Modified Principles

Traditional Principles	Modified Principles
1. Maintain a code repository	[RE1] maintain repository which supports the development strategy, for example, short release cycle.
2. Automate the build	[RE2] Automate build as well as deployment
3. Make the build self-testing	[RE3] Make the build self-testing and prioritized for different stages
4. Everyone commits to the baseline every day	[RE4] Recent changes should be built quickly and transparent
5. Every commit (to baseline) should be built	
6. Keep the build fast	
7. Test in a clone of the production environment	[RE5] Ready integration to production environment replica
8. Make it easy to get the latest deliverables	[RE6] Automation to get the deliverable at the proper location
9. Everyone can see the results of the latest build	(merged in other proposed principles)
10. Automate deployment	

V. LESSONS LEARNED AND DISCUSSIONS

A. Experiments

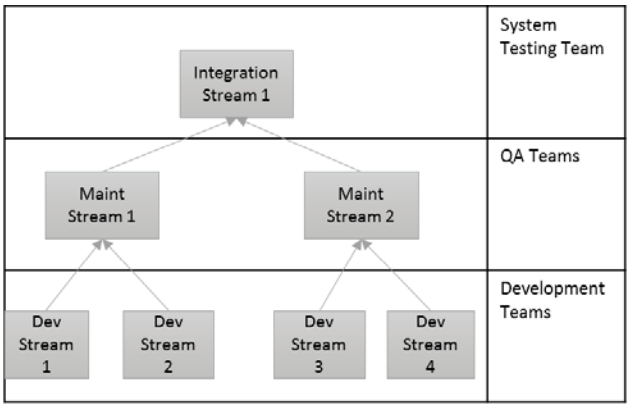
We have considered CI systems and their implementation from global service companies and several banking software departments. We also setup up and ran CI Lab for education and training purpose at California state university, Fullerton. Based on our direct observations on some global service organizations and our own CI Lab setup, we generalize common problems and made our recommendations. We also found that some non-ideal CI setups that follow our proposed recommendations, work satisfactorily for a given requirement.

B. Software configuration management (SCM) Strategy

We are in the process of implementing a complete CI solution where testing pause will not be an issue and it enforces the parallel and smooth development without stopping for one particular build [9]. We have implemented all our recommendations in our CI Lab.

We have implemented hierarchical structures of the code repository (Stream/ Branch / Trunk) where code is automatically transferred from development to maintenance to integration stream.

Here we build code separately and find the exact module for defect. The only responsible development team will work on correction and other teams can proceed forward. The goal is that we can eventually build corrected code and maintain versions at top level and consider only that build result for calculating build health.



For SCM strategy and CI jobs we need planning. This plan needs to be feasible and should support scalability for new streams and CI jobs.

It is always a good idea to maintain this hierarchy in such a way that it supports only one product line. The logic behind this recommendation focuses on keeping branching separate and achieving easy tracking and maintaining.

C. Tool Selections

Choosing a tool seems straightforward. We check a) compatibility with software technology we are building, b) which SCM system or tools we are using and c) platform support. In addition, one also needs to consider technical feasibility between the tools and our policies and environment. For example, we ask the following questions: Are we sure about the compatibility with the practice that we are following? Are we just adopting to the tool's policies and change our practice method accordingly? Have we decided on keeping the existing practice through tools? Which tool can enforce agile naturally? Which configuration speeds up our existing development environment? How much time do we need to train our stakeholders? Asking these questions before making your decision will surely help.

D. Rules to follow while using this system

- Set the priorities of the development environment's requirements
- Match Principles to the needs of the environment
- List high priority principles for your environment
- Try to achieve the suggested recommendation for high priority principles.
- These principles and recommendations are the guidelines and there is no need to force them to match the environment as we all know that it's an art of engineering.

E. Case studies

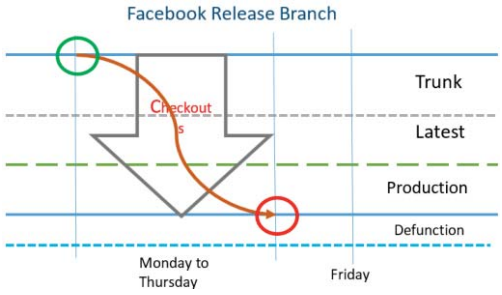
We have studied market's pioneers in this field such as Facebook release process, Google release cycle and Netflix API deployment. All pioneers are doing extremely well, i.e. development of quality product in a cost and time effective way. Our study showed that all cases are congruent to the recommendations proposed in this paper.

Facebook Release process

(The number in the brace [] is our proposed CI principle)

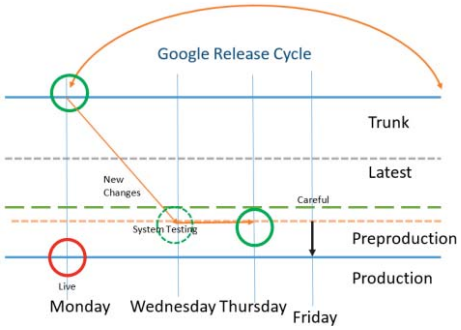
Strong culture of development and release process, e.g. "Friday Small, Daily Push" that fits into the proposed [RE 1]. Keep synch with other teams as you are not the only one who is pushing code live [RE 7]. Maintaining build transparency follows [RE 2, 3, 4]. One example is that 23 Facebook front-end teams know that where each team is working and which team is deploying what. Each developer is aware of the fact that his/her code is getting tested on a daily

basis. Well defined preproduction environment in which developers and testers can access and test upcoming flavor of Facebook [RE 5]. Testing of actual binary instead of relying on artifacts from the CI server [RE 6].



Continuous delivery at Google

Every day the team receives feedback from CI server. On Friday they don't release any new changes, but make sure whether all checkouts are proper or not. Every Monday they complete one release cycle. Google is doing its best in this practice due to a modular approach of development. Strong Integration infrastructure makes it possible for google to develop features separately. Each change is like plugin device one can add and remove without affecting the entire product. They maintain optimization separately and in parallel to the development, but behind one release cycle. So every week Googles product is getting better and updated. In a worst case, when Google wants to roll back, it is just a matter of few changes. A rollback simply means pointing server to the previous optimum step i.e. one release behind (five days back). In our opinion the above real-world case studies provide good indications that the proposed CI principles are on the right track.



REFERENCES

[1] T. Bruckhaus, N.H. Madhavii, I. Janssen, and J. Henshaw, "The Impact of Tools on Software Productivity", IEEE Software, vol. 13, no.5, Sep. 1996, pp. 29-38

[2] M. Fowler and M. Foemmel. Continuous Integration. http://www.thoughtworks.com/ContinuousIntegration.pdf, 2006.

[3]https://wiki.mozilla.org/Bugzilla:Home_Page

[4] M. Flower and M. Foemmel, "Continuous Integration", Online: http://martinfowler.com/articles/continuousIntegration.html.

[5]https://en.wikipedia.org/wiki/Jenkins_(software)

[6] W. Pedia, "Comparison of continuous integration software," Wiki Pedia, 2016. [Online]. Available: https://en.wikipedia.org/wiki/Comparison_of_continuous_integration_software. [Accessed 2016].

[7] P.M. Duvall, Continuous Integration: Improving Soft-ware Quality and Reducing Risk, Addpison-Wesley, Boston, USA, 2007.

[8] R. W. Ade Miller, "A Hundred Days of Continuous Integration," Agile Conference, Microsoft Corp, 2008.

[9] E. H. Kim, N. Corp., J. C. Na and S. M. Ryoo, "Test Automation Framework for Implementing Continuous Integration," IEEE, April 2009.

INTELLIGENT PLANT WATERING SYSTEM FOR RURAL FARMERS

John Samuel N.[1], Okonigene Robert E.[2], Samuel Peters C.[3], Okokpujie Kennedy[4]
samuel.john@covenantuniversity.edu.ng[1], robokonigene@aauekpoma.edu.ng[2],
sampet_halle09@yahoo.com[3], kennedy.okokpujie@covenantuniversity.edu.ng[4]

[1,3,4]Department of Electrical and Information Engineering, College of Engineering, Covenant University, Ota, Ogun State, Nigeria
[2]Department of Electrical and Electronics Engineering, Faculty of Engineering and Technology, Ambrose Alli University, Ekpoma, Edo State, Nigeria.

Abstract— This is an ongoing research study work. The objective of this study is to build an intelligent plant watering system for rural farmers. The study considered the availability of water supply in specific regions for five years. Also vital parameters statistics necessary for proper growth of each plant are stored in the system data base over the same period. Our study is primarily being guided by observations made in the rain fall pattern, different weather conditions, and environmental situation across the regions in the Northern and Southern parts of Nigeria. The target farmers are very poor. Therefore, our task is to produce a system that is affordable and reliable to these farmers. The complexity and stability of the system notwithstanding, overall, this study "Intelligent Plant Watering System for Rural Farmers" is being carried out to provide the rural farmers with a cheap, durable, power efficient, affordable, reliable, flexible, efficient and high performance intelligent plant watering system. Although this study is divided into three major groups, however, in this paper we try to present a subgroup that deals with soil moisture and fertility. The system based on its available statistics sets the various limits for the soil moisture, temperature and fertility. These features in the system ensure that water for irrigation is effectively managed and allowed to flow during specified temperature range. Also the soil fertility is properly regulated. This paper discursion focuses on the soil moisture and temperature.

Keyword: Irrigation, soil moisture, environmental temperature, microcontroller, artificial watering

I. INTRODUCTION

The required soil moisture is dependent on the type of plant. This is due to the fact that plants depend on water amongst others to survive. Water is extremely important to the existence of all living things. As simple as it is, the unique properties of water and its ability to appear in various forms makes it very essential in the many chemical reactions that take place, here on earth. One of these reactions is photosynthesis, the most important chemical reaction to physical life [1]. The basic process of photosynthesis can be represented as follows:

$$\underset{\text{Carbon Dioxide}}{CO_2} + \underset{\text{Water}}{H_2O} \xrightarrow{\text{light}} \underset{\text{Glucose (sugar)}}{C_6H_{12}O_6} + \underset{\text{Oxygen}}{O_2}$$

Plants need water, with other compounds and under certain circumstances, to produce energy. Almost every other living thing depends on this simple process of plant nutrition to survive [2].

To produce enough energy, plants synthesize the chemical compounds derived from the soil in the presence of Carbon dioxide, water and sunlight. It means that this process, photosynthesis, cannot take place without the presence of water. Thus, the soil around the plant should be wet regularly for the plant to produce energy regularly [3].

There are two basic methods of watering namely:

- The natural method
- The artificial method (Irrigation)

The main source of natural watering is the rainfall. During rainy seasons, plants rely comfortably on the availability of rainfall. But plants cannot solely rely on this system of watering as there is no rainfall every day throughout the year. Moreover, there are regions in Nigeria where there is very little rainfall throughout the year. Such regions are known as arid regions or zones [4].

It is important to note how climate has varied and changed in the past twenty years. An idea of the monthly mean historical rainfall and temperature data is necessary in order to understand the baseline climate and seasonality by month, for specific years, and for rainfall and temperature. In this study, the observation was that the mean historical monthly temperature for Nigeria during the time period 1995-2015 was lowest at 22^{o}C in January and highest at 30.6^{o}C in April. Equally, within this period the mean historical monthly rainfall was lowest at 7.1 mm in February and highest at 232mm in August [5 - 9].

A good way to provide adequate irrigation through artificial means is the automated irrigation system using a microcontroller to supervise the process. This system is also efficient in saving water, as the microcontroller ensures that the exact amount of water needed to saturate the soil when dry is provided. This is done by the help of a moisture sensor which ensures that water is supplied to the soil when the soil

moisture goes below a certain fixed level. The moisture sensor sends information about the soil moisture to the microcontroller. The microcontroller then receives the information, and then turns the solenoid valve on or off based on the information signal received.

The aspect of the study discussed the plant watering system that provides water in stipulated quantity when needed by the soil. With this system in place, the farmer, gardener or caretaker does not have to involve much effort in ensuring that the plant(s) are well watered. In the process it also determines the soil temperature. Temperature sensors are placed into the soil and configured with the microcontroller. The system during irrigation allows water flow only at low temperatures. This reduced the water loss due to evaporation. [10].

Irrigation is the artificial application of water to the land for various purposes which may include crop cultivation, re-vegetation of disturbed soils in dry areas and during periods of inadequate rainfall, and maintenance of landscapes.

With the invention of sensors and transducers, a great opportunity has been achieved for the application of electronics to solving physical day to day problems. Through the invention of soil moisture sensors and transducers, the real –time soil moisture status can be electronically monitored and same information can be used to determine the water requirement and through actuators, induce irrigation. Different approaches have been undertaken to manage irrigation using electronics. The task also includes the use of soil moisture sensor basic operating principles to produce a cheaper irrigation system for rural farmers.

II. PROCEDURE FOR DATA COLLATION

This part of the system is designed to manage irrigation based on response to the real-time status of the soil moisture. The system will cause the soil moisture to always be in a certain range suitable for proper crop development.

The underlying principle in this case is simple. It consists of the moisture and temperature sensors, the microcontroller, the Liquid Crystal Display, and the solenoid valve. The microcontroller converts the analog signals sent by the moisture sensor buried in the soil into digital values. It then compares this value with the accustomed value that represents the lowest allowable moisture content in the soil. For different plants the system reads the sensor value. Below the minimum set value, the microcontroller sends a "HIGH" signal to the solenoid valve to trigger it on. Also, when the sensor reads a value above the set value representing the maximum allowable moisture content in the soil, the microcontroller sends a "LOW" signal to the solenoid valve thereby causing the valve to be turned off. The temperature sensor ensures that the soil is watered when the temperature of the environment is below a preset value. All the reference information regarding the data that will be required by the microcontroller to make feature decisions about the soil moisture, temperature and fertility are stored in the database. The data are from the five years studies.

III. THE IRRIGATION CONTROL SYSTEM

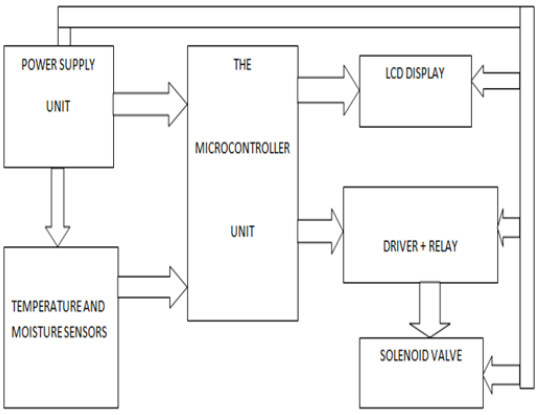


Figure 1: Block diagram description of the watering system

A. *The Sensing Unit*

Figure 1 is the block diagram of the circuit for irrigation system. The sensing units are responsible for the detection of the presence of the physical parameters and converting same to electrical form for processing. The physical parameters of interest are the soil moisture, and the ambient temperature.

B. *The Soil Moisture Sensor*

The soil moisture sensor utilized has its voltage output proportional to the quantity of water in the soil. Its specified supply voltage is from 3.3V to 5V and with this supply, it gives an output voltage of between 0V to 2.3V for the full range of complete dryness to submersion in water. Its rating for maximum operating current is 0.15A. Its output is fed into the analog-to-digital converter (ADC) input of the microcontroller.

C. *The Temperature Sensor*

The temperature sensor used in this circuit is the LM35 temperature transducer. It is a precision temperature transducer with a linear voltage output over the range of -55 ^{o}C to 150^{o}C [13]. Its favourable property is its linearity and step-wise sensitivity. It has a sensitivity of 0.01V/^{o}C starting from 0^{o}C hence its temperature can easily be calculated. The output of the transducer is applied to the ADC segment of the microcontroller for processing.

D. The Control Unit

This unit is basically the section that provides the control of the whole system. It consists of a microcontroller IC chip plus peripheral components and the control logic (firmware) which the chip functions with. The microcontroller chip is the central hardware component while the program/code written in Mikro-C language is the firmware component. The microcontroller used in this project is the PIC18F452 shown in Figure 2. The PIC18F452 is a 40-pin, 8-bit microcontroller [14]. The features of the PIC18F452 microcontroller make it a suitable choice for use in this automatic irrigation controller system.

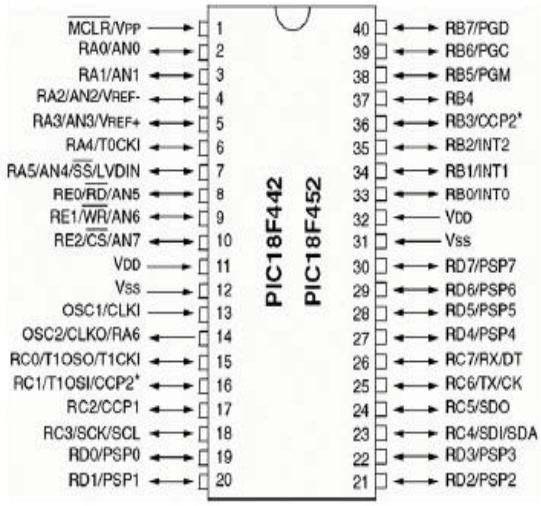


Figure 2: Pin arrangement of PIC 18F452 Micro-controller

E. The Display Unit

The display unit is simply an output unit used for the purpose of giving the user required information. The display unit is a simple 16x4 LCD module. The information displayed is the current soil moisture, the current temperature and the state of the system.

F. The Switching Unit

The switching unit consists of the BC108B transistor, the relay, and the solenoid valve.

The solenoid valve is a 12V DC solenoid valve. The solenoid valve is connected to the normally-open contact of the relay. When sufficient current enters the base of the transistor, current is allowed to flow through the coils of the relay (which is an electromagnetic switch). This sets up a voltage in the coils of the relay causing the normally-open contact to become a normally-closed contact [16]. When this occurs, current flows through the solenoid valve from the 12V DC supply and the solenoid valve is turned on.

IV. TEST RESULTS

Several tests and observations were made to ensure the proper functioning of the system. The physical model was tested and the overall response and performance of the system were checked. This covers the testing of various physical parameters of the system. Tests were done on the soil moisture sensor's natural response to pure water in order to ascertain its output voltage. Also, several soil samples were tested to determine the output voltage from the sensor at dry and saturated soil conditions. The sensor was connected to a 5V DC power supply. The ground terminal was grounded, and the output voltage was evaluated with the aid of a good voltmeter. The output of the sensor was connected to the micro-controller and the response of the micro-controller was observed. The sensor was placed in pure water giving the observed typical values shown in Table 1.

Table 1: Showing results obtained from moisture sensor test

INPUT INTO SENSOR	TEST CONDITION	OUTPUT FROM SENSOR	MICRO-CONTROLLER RESPONSE	REMARKS
5V	In Air	0V	-	Expected
5V	In Pure Water	2.3V	Gives no output, hence no watering is done.	Expected

The soil moisture sensor was placed into a sandy soil sample (due to its ease of availability). The output of the sensor was then determined in dry and wet soil conditions. The results obtained are shown in Table 2.

The temperature sensor used in this test circuit is LM35 temperature sensor. The temperature sensor was connected similar to the soil moisture sensor. The testing of the temperature sensor was done to determine the voltage output of the sensor to different changes in temperature. The sensor output was measured at room temperature (29^{o}C) and outside on a sunny afternoon (32^{o}C). The outputs of the sensor were 0.285V and 0.318V respectively. These measured values correspond with the calculated values of 0.29V and 0.32V.

Table 2: Table showing the output of the sensor with respect to different soil conditions

INPUT INTO SENSOR	SOIL CONDITION	OUTPUT FROM SENSOR	MCU RESPONSE	REMARKS
5V	Dry	0.8V	Gives a high output, hence watering should be done	Expected
5V	Wet	2.2V	Gives a low output to indicate that watering should stop	Expected

The Figure 3 and Figure 4 are the flowchart used for the control of the solenoid valve and the display unit respectively.

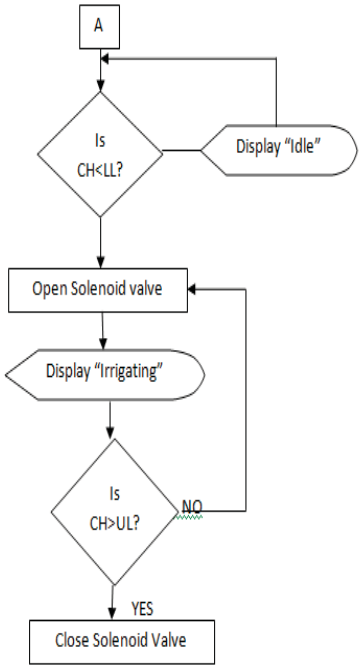


Figure 3: Flowchart of the solenoid valve control process

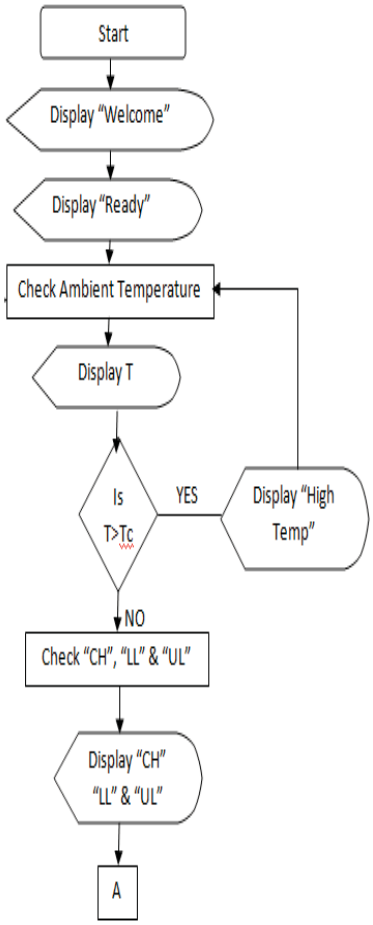


Figure 4: Flowchart representing the display unit of the system

MAJOR ACHIEVEMENTS

- A system was designed that measures the surrounding soil humidity and atmospheric temperature at a relatively very cheap cost. Most of the components are available local shops. To reduce the cost of production further we need to replace the soil moisture with very cheap models without losing the sensitivity and reliability of the overall system.
- The designed device reduced water used for irrigation by 62% as compared to what is currently being used by the rural farmers and accurately waters the plant at the right times.

The Figure 5 shows the tested circuit diagram for the automated irrigation system.

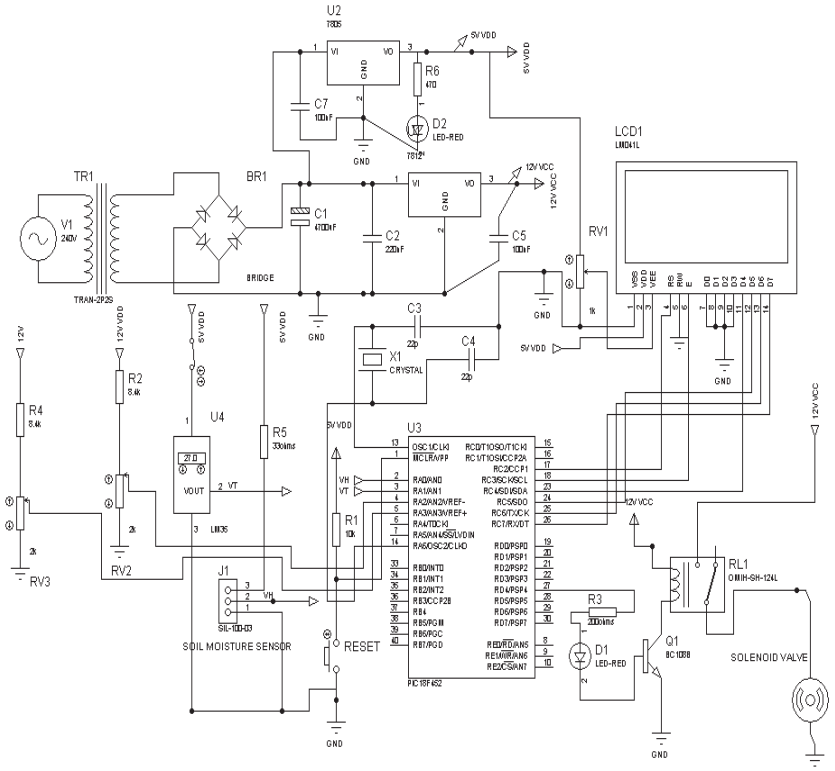


Fig 5 The circuit diagram used to test the automated irrigation system

V. CONCLUSION

The system was developed successfully, meeting the aims of workability low-cost design and simplicity of operation. Hence irrigation can be automatically controlled by rural farmers resulting to greater productivity of crops and efficiency of irrigation management. It is our desire that the entire final system will meet the set objectives for rural farmers use.

REFERENCES

[1] Michael McDarby, " Section 2 Chapter 1: Photosynthesis" http://faculty.fmcc.suny.edu/mcdarby/Animals&PlantsBook/Plants/01-Photosynthesis.htm, 2014.

[2] Shmoop Editorial Team. "Ecology: Organisms and Their Environments Terms." *Shmoop*. Shmoop University, Inc., 11 Nov. 2008. Web. 20 Mar. 2016...

[3] "10 most common herb garden mistakes" http://herbgardens.about.com, 2014.

[4] "Criteria and options for appropriate irrigation methods",

[5] Cardenas-Lailhacar, B. and M.D. Dukes. 2014. Effect of Temperature and Salinity on the Precision and Accuracy of Landscape Irrigation Soil Moisture Sensor Systems. *Journal of Irrigation and Drainage Engineering*, 0401407.

[6] Davis, S. and Dukes, M. (2015). "Methodologies for Successful Implementation of Smart Irrigation Controllers." *J. Irrig. Drain Eng.*, 141(3), 04014055.

[7] Cardenas-Lailhacar, B. and Dukes, M. (2015). Effect of Temperature and Salinity on the Precision and Accuracy of Landscape Irrigation Soil Moisture Sensor Systems. *J. Irrig. Drain Eng.*, 141(7), 04014076

[8] Dukes, M. D., B. Cardenas-Lailhacar, and G. L. Miller. 2005. Residential Irrigation Based on Soil Moisture. *Resource Magazine*, June/July. American Society of Agricultural and Biological Engineers, St. Joseph, MI.

[9] Haley, M. B., M. D. Dukes, and G. L. Miller. 2007. Residential irrigation water use in Central Florida. *Journal of Irrigation and Drainage Engineering* 133(5):427-434.

[10] "Water plants in the morning" http://www.goodhousekeeping.com/home/gardening/water-plants-morning, 2014.

[11] Elias Awili: "Design and Construction of an Automatic water sprinkler for small scale gardens," Covenant University, 2013.

[12] Purna Prakash Dondapat, K. Govinda Rajulu: "An Automated Multi Sensored Green House Management," International Journal of Technological Exploration and Learning (IJTEL), 2012

[13] G. K. Banerjee, Rahul Singhal: "Microcontroller Based Polyhouse Automation Controller", International Symposium on Electronic Systems Design (ISED), Bhubaneswar, 2010

[14] "Transformers - Faysal Electric Company" http://faysal-electric-co.com/transformer/

[15] John Bird: "Electrical Circuit Theory and Technology," Elsevier Ltd. Publishers, 2007.

[16] Illinois Capacitors Inc: "Filtering," Electronic Components Assemblies and Materials Association (ECA), 2012.

[17] "Voltage-regulator", http://www.britannica.com/EBchecked/topic/632467/voltage-regulator, 2014.

[18] Texas Instruments: "LM35 Precision Centigrade Temperature Sensors", TI publications 2013.

[19] Dogan Ibrahim: "Advanced PIC Microcontroller projects in C", Newnes publishers, 2008.

[20] "CrystalFontz America", http://www.crystalfontz.com/product/CFAH1604BNGHET, 2014.

[21] Madeline Bullock, " How Relays Works", http://electronics.howstuffworks.com/relay.htm, 1998.

[22] Otarelli, L., J. M. Scholberg, M. D. Dukes, and R. Muñoz-Carpena. 2008. Fertilizer residence time affects nitrogen uptake efficiency and growth of sweet corn. *Journal of Environmental Quality* 37(3):1271–1278.

ISBN: 1-60132-446-4, CSREA Press ©

SESSION

SOFTWARE SYSTEMS, TOOLS, FRAMEWORKS + NOVEL SOFTWARE APPLICATIONS

Chair(s)

TBA

ISBN: 1-60132-446-4, CSREA Press ©

ISBN: 1-60132-446-4, CSREA Press ©

Applying *EM³: Handover Framework* in a Project Parking Context

Ahmad Salman Khan [1], **Mira Kajko-Mattsson** [2]
[1]Department of Computer Science & IT, University of Lahore, Lahore, Pakistan
[2]School of ICT, KTH Royal Institute of Technology, Stockholm, Sweden

Abstract - *A well-defined handover process model is imperative and critical for succeeding with the transfer of a software system from one party to another. Despite this, there still do not exist any up-to-date handover process models. Recently, however, we have developed EM³: Handover Framework aiding organizations in constructing their own handover process models. In this paper, we evaluate it in one Swedish software organization via participatory observation. Our goal is to examine the framework's applicability and usefulness in a real-world industrial scenario. The handover process studied was of a self-to-self type and it was conducted in a project parking context. Our results show that our framework is fully applicable in an industrial setting.*

Keywords: self-to-self software transfer; participatory observation.

1 Introduction

A well-defined software system handover process model is imperative and critical for planning and managing a software handover and for alleviating many handover problems. Failing to transfer a system from developer to maintainer may lead to serious consequences such as loss of productivity, loss of maintainer credibility, loss of system and maintenance process quality, and sometimes, even loss of business. Despite this, there still do not exist any up-to-date handover process models that designate important process features that are necessary for conducting a systematic and disciplined software system transition. Regrettably, software handover is still an under-researched and neglected domain. The published handover models are either too old or they are defined on a very general level [1] [2] [3] [4] [5] [6].

Lack of appropriate software system handover process models leads to the fact that companies do not have any process models to follow while performing their handover, or if they do have them, then they still may feel insecure whether their models appropriately reflect the complexity of the handover process domain. One such a company is *E-Identity,* a company that has commissioned us to conduct software system handover. Although the company has developed its own handover process model, they still felt very insecure in conducting it in one of their very unique and intricate handover contexts, that is, in the project parking context.

In this paper, we report on the results of conducting a handover process at *E-Identity* using our recently developed handover process model – *EM³: Handover Framework. EM³* stands for *Evolution and Maintenance Management Model*. Our goal was to observe the implementation of our framework and examine its applicability and usefulness in an industrial setting. The handover process studied was of a self-to-self handover type, it was conducted in a project parking context and its evaluation was made via participatory observation [7].

Parking implies that the project gets deactivated for some known or unknown period of time, teams working on the project get dissolved, and probably with time, the project will get reactivated (resumed), however, with new team members. Parking is also a type of self-to-self handover. The company (the first self) transfers a partially developed system to itself (the second self).

The remainder of this paper is as follows. Section 2 briefly presents the company and its handover process. Section 3 describes our research process and Section 4 briefly describes *EM³: Handover Framework*. Section 5 reports on results of the framework's implementation within the company studied and Section 6 rounds up the paper.

2 Company Description

E-Identity is a Swedish company based in southern Sweden. It develops a product for digital identity authentication. It has encountered a financial crisis which did not allow it to continue developing its product. However, the company was strongly determined to continue with its development as soon as it recovered from the crisis. To be able to continue with the project, the company had to park it.

As shown in Figure 1, the company's system consisted of two parts. These were *API infrastructure* and *Digital Identity Authentication Product*. Different teams were responsible for these parts. The *API infrastructure development* team was responsible for developing the API infrastructure and for establishing a platform for the application development. The *application development* team then used the APIs to develop the digital identity authentication product for the end-user. Here, the *application development team* was an internal customer to the infrastructure development team.

Before the crisis, the company employed about 25 people. At the moment of writing this paper, the company had to dismiss about 15 people and dissolve the teams. Out of the ten

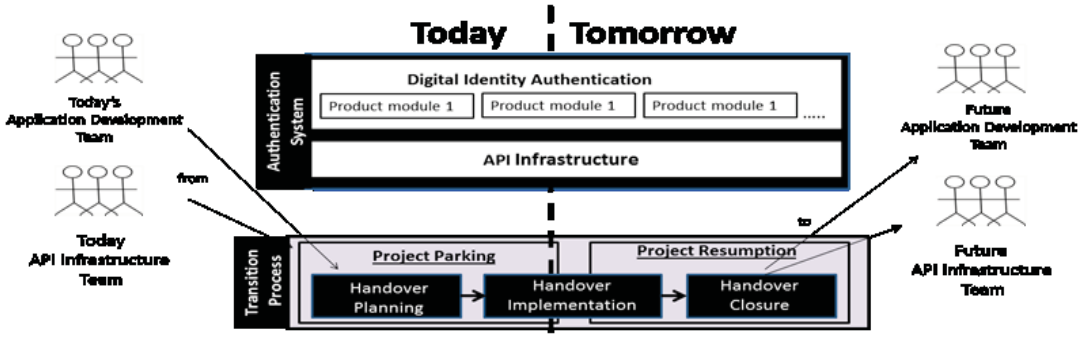


Figure 1. The handover context at E-Identity

people who stayed, four people were involved in the handover process. These were the following (1) *development team lead* responsible for documenting the system knowledge and generating a stable infrastructure development API release, (2) *project manager* responsible for managing the handover project, (3) *product owner* responsible for the product to be handed over, and finally, (4) *researcher* responsible for monitoring and supervising the handover process.

At *E-Identity* the handover process activities are classified under three categories. As shown at the bottom of Figure 1, these are *handover planning*, *handover implementation* and *handover closure*. *Project parking* stage mainly comprises activities dealing with *handover planning* and a few activities dealing with *handover implementation*. *Project resumption* comprises the *handover closure* activities and the rest of the *handover implementation* activities.

3 Research Process

We followed the participatory research where we played the role of an active participant [7]. This means that through participating in the process, we tried to understand the handover process studied by actively observing the process. We also provided support to the company while implementing *EM*^{3}. In this way, we gained a close familiarity with the process and the people performing the process.

To get as much intimacy with the handover process as possible, we used a wide range of data collection methods such as direct observation, active participation, collective discussions, brainstorming sessions, documentation study, and informal interviews. In this way, we could identify similarities and discrepancies between the *EM*^{3} practices and the handover process studied.

The *project parking* phase took three weeks to perform. During this time, we conducted four major steps that were typical of a participant observation method [7]. These were (1) *Establish Rapport*, (2) *Acting in the Field*, (3) *Recording Observations*, and (4) *Analyzing Data*.

The first phase, the *Establish Rapport* phase, lasted for only one day. We visited the company studied, we acquainted ourselves with the company's employees and acquired some introductory information about the company's situation.

In the *Acting in the Field* phase, we tried to act just as the company's "*local*" member with some minor exceptions [7]. We had to get a thorough understanding of the company, its product and processes. For this reason, we studied all the organizational documentation that was relevant and available. Just because not much documentation was in place, we continued our study via informal discussions.

The third phase, the *Recording Observations* phase, ran in parallel with the *Acting in the Field* phase. While doing our work, we matched it against *EM*^{3}, compared the framework's activities with the company's handover activities and evaluated their applicability. Wherever it was relevant, we suggested improvements. This helped the company to cover the gaps in their handover process and helped us gain feedback for improving our model.

Some *EM*^{3} activities could not be implemented in the process studied. Being such a case, we first asked whether the company conducted them in other handover contexts and inquired about their usefulness.

Finally, in the *Analyzing Data* phase, we studied each of the *EM*^{3} activities in order to find out whether it was fully or partially implemented and to find out reasons for their non-adherence to the executed handover process. It is these findings that constitute the contribution of this paper.

4 EM^3 : Handover Framework

EM^3*: Handover Framework* provides a skeletal structure of six different parts that are necessary for creating handover processes. It is a result of an explorative study made in 61 companies [8]. As shown in Figure 3, its central part is *EM*^{3}*: Handover Taxonomy* – a set of component practices including the activities that play a significant role in executing a handover process. The taxonomy activities may be used for orchestrating handover processes using the framework's other

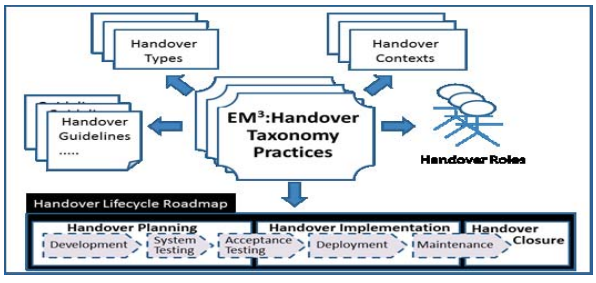


Figure 3. EM^3: Handover Framework

five parts such as (1) *Handover Types* designating types of software handover, (2) *Handover Contexts* placing handover within software lifecycle, (3) *Handover Roles* identifying the main responsibilities in the handover process, (4) *Handover Lifecycle Roadmap* designating time spaces in the handover lifecycle phases, and (5) *Handover Guidelines* providing support in the handover endeavors.

4.1 EM^3: Handover Taxonomy

EM^3: Handover Taxonomy comprises eight practices important for implementing software system handover. They constitute an improved version of the initial taxonomy of handover activities [11]. In this section, we briefly describe them and their constituent activities. To be able to follow our descriptions, we strongly advise our reader to follow the *EM^3* activities in Table 1.

4.1.1 Management and Administration

The *Management and Administration (MA)* practice includes the activities required for handling and controlling the handover process. The success of the overall process strongly depends on it. As shown in Table 1, the practice contains activities starting from planning a handover process, to managing it, to finally, evaluating it postmortem.

Before starting transition, organizations should identify its type and complexity. Transition might be self-to-self or it might be an external one where transitioners and transitionees are separate organizations. Transition might be of high complexity implying a handover of a large safety critical system among several parties or it might be as simple as a self-to-self handover of a system version.

As a next step, the transition team should create a transition plan and assure that important management plans are in place. Being guided by parameters such as, for instance, transition deadline, resource constraints and the like, the transition plan should define transition manpower resource requirements, budget and schedule. The management plans, on the other hand, should plan for the processes that interact with the transition process such as development, maintenance processes, to mention a few. A communication model should be in place for interacting and for transferring knowledge between different parties. Throughout the handover, the handover process should be continuously monitored and, at its end, it should be evaluated postmortem.

Determining the transition type and complexity is a prerequisite for defining a transition strategy, for establishing a transition team, for defining a transition process, and for designating a transitionee. A transition team should from now on manage and administer the transition process. It should enlist all its core activities, and the activities that are part of other processes, the processes that either impact or are impacted by the transition. Failing to identify them may jeopardize the whole transition. Finally, all the stakeholders involved, including the transitionees, should agree upon the design of the transition process to be executed.

4.1.2 Maintenance Environment

The transitionee has to have the environment that is right from the beginning. Hence, as shown in Table 1, the *Maintenance Environment* (ME) practice includes the activities that are required for determining the needs for hardware suites, software suites and maintenance support suites and activities required for their installation.

The needs should be determined in advance in cases one transfers a newly developed system. In other cases, the current suites should be assessed whether they still fulfill their function. Here, one should identify their potential adequacies and deficiencies and assure that they are compatible across all the environments, that is, the environments of the transitioners, transitionees and of the customers. If the suites are not determined or assessed in advance, then there is a risk that they will not be delivered on time, that the transitionees will not get enough time for learning them or that they may face compatibility problem.

4.1.3 Version and Configuration Management

The *Version and Configuration Management* (VCM) practice includes the activities required for keeping track of changes made to a software system before, during and after handover. This practice is critical for assuring that the system that has been handed over includes the right components. As shown in Table 1, it deals with placing the system under version and configuration management and baselines.

It goes without saying that it is significant to baseline the software system to be handed over. In the context of a system handover, at least two groups of baselines are relevant. These are test and postdelivery baselines. The test baselines are created before the system delivery during different testing phases. They constitute platforms for identifying and tracking all the changes made to the system and for making important decisions on handover. The postdelivery baselines, on the other hand, are created just after the system delivery. They constitute important platforms for synchronizing the changes across the development, maintenance and operational environments and for assuring that they have identical or as identical as possible system copies.

4.1.4 Training

People involved in handover must be trained so that they can work from the first day after handover. As shown in Table 1, the *Training* practice focuses on training planning, creating training material and on providing training. To ensure that the training is effective, the practice designates roles responsible for the training process.

4.1.5 Deployment

Deployment is a critical prerequisite for commencing software operation and maintenance. As shown in Table 1, the *Deployment* practice includes activities starting from defining a release scope and contents to preparing for installation, to

Table 1. EM3: Handover Taxonomy practices

+ stands for observed and performed, +(i) stands for inquired about and performed, -- stands for not performed, P stands for partially performed and NA stands for not applicable. Plan stands for handover planning, Impl stands for handover implementation and Clos stands for handover closure.

Activities	Status	Phase
MANAGEMENT AND ADMINISTRATION		
MA 1: Determine/redetermine type and complexity of transition	+	Plan
MA 2: Define a strategy for transition process	+	Plan
MA 3: Designate a transitionee	+	Plan
MA 4: Establish a transition team	+	Plan
MA 5: Define a transition process	+	Plan
MA 5.1: Identify core transition activities	+	Plan
MA 5.2: Identify activities of other processes that impact or are impacted by the transition	+	Plan
MA 6: Agree upon the executed transition process	+	Plan
MA 7: Create/adjust a transition plan	+	Plan
MA 7.1: Define/adjust parameters guiding the design of the transition plan	+	Plan
MA 7.2: Create the transition plan using the parameters	+	Plan
MA 7.3: Define transition resource requirements	+	Plan
MA 7.3.1: Define manpower requirements	+	Plan
MA 7.3.1.1: Define maintenance manpower requirements	+	Plan
MA 7.3.1.2: Define developer manpower resources	+	Plan
MA 7.3.1.3: Define transition team manpower resources, if any	+	Plan
MA 7.3.1.4: Define other manpower resources, if any	NA	Plan
MA 7.3.2: Define maintenance facility requirements	+	Plan
MA 7.4: Determine transition budget	+	Plan
MA 7.5: Create a transition schedule	+	Plan
MA 8: Develop management plans necessary for transition	+	Plan
MA 9: Determine a communication model to be used within transition	+	Plan
MA 10: Monitor the transition process	+	Impl
MA 11: Evaluate the transition process postmorten	+	Clousre

Activities	Sta.us	Phase
MAINTENANCE ENVIRONMENT		
ME 1. Manage hardware/software suite needs	+	Plan
ME 1.1: Determine hardware/software suite needs	+	Plan
ME 1.1.1: Determine hardware and software packages constituting the hardware/software suites	+	Plan
ME 1.1.2: Assure that hardware/software suite needs match the developer's hardware/software suites	+	Plan
ME 1.1.3: Assure that hardware/software suite needs match the customer's hardware/software suites	+	Plan
ME 1.2: Install hardware/software suite	+	Impl
ME 1.3: Grant the transitionee access permission to hardware/software suites	+(i)	Impl
ME 1.4: Assess current hardware/software suite, if any	+	Plan
ME 1.5: Remedy the deficiencies in hardware/software suite , if any	+	Impl
ME 2: Manage maintenance support suite	+	Plan
ME 2.1: Determine maintenance support suite	+	Plan
ME 2.1.1: Determine software packages constituting the maintenance support suite	+	Plan
ME 2.2: Install maintenance support suite	+	Impl
ME 2.3: Assess maintenance support suite	+	Plan
ME2.4: Remedy the deficiencies in maintenance support suite , if any	+	Impl
VERSION AND CONFIGURATION MANAGEMENT		
VC 1: Manage version and configuration	+	Impl
VC 1.1: Define rules to uniquely identify, name and label the configuration items and their relationships	+	Plan
VC 1.2: Define how the configuration items are to be selected, grouped and classified	+	Plan
VC 1.3: Decide on how to identify and track changes made to customizable configuration items during handover	NA	Plan
VC 1.4: Put software under configuration management	+	Impl
VC 1. 5: Place software under version control management	+	Impl

Activities	Status	Phase
VC 2: Manage baselines	+	Impl
VC 2.1: Establish test baselines (system test baseline, acceptance test baseline)	+	Impl
VC 2.1.1: Assist developers in attending to problem reports during acceptance testing	NR	Impl
VC 2.1.2: Identify and track customizable configuration items during handover	NA	Impl
VC 2.1.3: Keep track of the changes made to the baselines	+	Impl
VC 2.1.4: Notify all the stakeholders involved about the changes made to the system	+	Impl
VC 2.2: Establish post-delivery baselines (operational baseline and maintenance baseline)	+(i)	Closure
VC 2.2.1: Check whether the reported problems are not of critical nature	+(i)	Closure
VC 2.2.2: Synchronize system changes made during system handover in all the environments (operational, development and maintenance)	+(i)	Closure
VC 2.2.3: Assure that the identical copies (or as identical copies as it is possible) are installed in the operational, development and maintenance environments	+(i)	Closure
VC 2.2.4: Accept and approve the system for operation and maintenance	+(i)	Closure
TRAINING		
T 1: Designate the role responsible for managing the training process	+	Plan
T 2: Plan training	+	Plan
T 2.1: Identify training topics to be taught (e.g. system, maintenance process, support process, technology, legal aspects)	+	Plan
T 2.2: Identify the trainee groups	+	Plan
T 2.3: Determine training needs of each trainee group with respect to the training topics	+	Plan
T2.4: Define methods of training	+	Plan
T 3: Create/update training material	+	Plan
T 4: Identify the role responsible for providing the training	+	Plan
T 5: Prepare for training	+	Plan
T 5.1: Adapt the training material to the trainee group and its needs	+	Plan
T 5.2: Setup training environment, if required	+(i)	Plan
T 6: Provide training	+(i)	Impl
T 7: Involve maintainers in attending to modification requests	+(i)	Impl

Activities	Status	Phase
T 8: Involve maintainers in white box testing and debugging	+(i)	Impl
T 9: Provide onsite support, if needed	+(i)	Impl
T 10: Develop educational policies providing guidance for developing educational plans	+(i)	Plan
T 11: Develop project specific educational plan using policy guidelines	+(i)	Plan
DEPLOYMENT		
DP 1: Define/continuously re-define the scope and contents of the release	+	Plan
DP 2: Determine type of release (major/minor)	+	Plan
DP 3: Create a deployment team	+	Plan
DP 4: Develop installation procedures	+	Plan
DP 4.1: Develop rollback procedures	+	Plan
DP 4.2: Develop installation manuals	+	Plan
DP 4.3: List organizations and stakeholders affected by the new release	+	Plan
DP 4.4: Prepare release and build documentation	+	Plan
DP 4.5: Define/continuously update the access rights to release components	+	Plan
DP 5: Installation	+	Impl
DP 5.1: Take a backup of the system release to be de-installed	+	Impl
DP 5.2: Perform deployment readiness test	+	Impl
DP 5.3: Distribute and deliver the system and /or system components at a correct location and time	+	Impl
DP 5.4: Install the new system version	+	Impl
DP 5.5: Install operational data	+	Impl
DP 5.6: Record any incidents, unexpected events, issues or deviations from the release plan	+	Impl
DP 5.7: Perform deployment verification tests	+	Impl
DP 6: Deployment Closure	+	Clos
DP 6.1: Review the system deployment	+	Post
DP 6.2: Close the deployment	+	Post

Activities	Status	Phase
DP 6.2: Close the deployment	+	Post
DP 7: Planning for future releases	+(i)	Plan
DP 7.1: Plan updates of future releases	+(i)	Plan
DP 7.1.1: Identify features to be deployed in the next release	+(i)	Plan
DP 7.1.2: Determine the impact of the externally acquired components on the planned release and vice versa, if relevant	+(i)	Plan
DP 7.1.3: Estimate release size, effort, time and hardware/software infrastructure required	+(i)	Plan
DP 7.2: Determine the system distribution structure	+(i)	Plan
DP 7.3: Determine forms of deployment software	+(i)	Plan
DOCUMENTATION		
D 1: Establish a system documentation repository	+	Plan
D 2: Define services to be provided by the system documentation repository	+	Plan
D 2.1: Identify different types of services to be provided by the system documentation repository	+	Plan
D 2.2: Determine groups of access rights to the services	+	Plan
D 3: Subject system documentation repository to SCM	+	Impl
D 4: Establish documentation standards	+	Plan
D 4.1: Define organizational policies/rules/guidelines for developing documentation standards	P	Plan
D 4.2: Share documentation standards with the maintenance team during handover	+(i)	Impl
D 4.3: Develop templates for documentation according to the defined policies/rules/guidelines	+	Plan
D 4.4: Create rules for updating the system documentation repository	+	Plan
D 4.5: Create mechanisms for controlling the status of the system documentation repository	P	Plan
D 5: Transfer the documents from the documentation repository to maintainer	+(i)	Impl
MAINTAINABILITY MANAGEMENT		
MM 1: Assess system maintainability	P	Plan
MM 1.1: Define system maintainability attributes	P	Plan
MM 1.2: Define rules and guidelines for adhering to the system maintainability	P	Plan
MM 1.3: Identify milestones for assessing system maintainability	P	Plan
MM 1.4: Assess system maintainability using the system maintainability attributes	P	Impl

Activities	Status	Phase
MM 1.5: Assess procedures for managing and controlling system maintainability	P	Plan
MM 2: Assess data maintainability	--	Plan
MM 2.1: Define data maintainability attributes	--	Plan
MM 2.2: Define rules and guidelines for adhering to the data maintainability	--	Plan
MM 2.3: Identify milestones for assessing data maintainability	--	Plan
MM 2.4: Assess data maintainability using the data maintainability attributes	--	Impl
MM 2.5: Assess procedures for managing and controlling data maintainability	--	Plan
SOFTWARE SYSTEM TRANSFER		
ST1: Monitor status of software components	+	Impl
ST1.1: Identify stable software components ready to be used in the system to be handed over	+	Impl
ST1.2: Identify software components under testing stage	+	Impl
ST1.3: Identify software components under development stage	+	Impl
ST2: Make decision on the components to be handed over	+	Impl
ST3: Manage modification Requests	+	Impl
ST3.1: Create a template for managing information about modification requests and their management	+	Impl
ST3.2: Place modification requests in a modification request repository	+	Impl
ST3.3: Use modification requests to revise the handover decision	+	Impl
ST4: Transfer software system	+(i)	Impl
ST4.1: Transfer the agreed upon software components	+(i)	Impl
ST4.2: Transfer the replica of the operational data	+(i)	Impl
ST4.3: Transfer modification requests	+(i)	Impl
ST4.4: Monitor the system after handover	+(i)	Clos
ST4.5: Singoff the handover closuer	+(i)	Clos

installing and deploying the system, to finally, closing the deployment and planning for future releases.

4.1.6 Documenation

The *Documentation* practice focuses on establishing a system documentation repository and mechanisms for controlling its status. Both developers and maintainers need a central location for storing software system documentation and for assuring that nothing gets lost while handing over a software system. As shown in Table 1, the practice includes (1) activities for establishing a system documentation repository, (2) activities for subjecting the documentation repository to SCM, and (3) mechanisms for controlling the status of the system repository.

4.1.7 Maintainability Management

The *Maintainability* practice includes assessment of two types of maintainability: (1) *system maintainability* referring to the ease with which one changes the system, and (2) *data maintainability* referring to data integrity, correctness and consistency. If the system is not maintainable, then it becomes difficult for the maintenance team to understand, and thereby, difficult to evolve and change. If the data is defective, then the company may encounter the problem of a data loss.

Both maintainability types must be assessed before system handover. As shown in Table 1, one must define appropriate system and data maintainability attributes, define rules for adhering to them, identify milestones for assessing them, and finally, assess them. After finalizing the handover process, one should assess their procedures for managing and controlling data and system maintainability.

5 Status

In this section, we present the results of implementing *EM*$^{3:}$ *Handover Taxonomy* activities at *E-Identity*. Due to space restrictions, we cannot report on the implementation of all of them. We only report on the most important activities. For more information, interested readers are welcome to study [8]. Finally, while participating in the handover process, we observed that not all the *EM*^{3} activities were implementable in the project parking context. To evaluate them, we inquired about their applicability and usefulness in other handover contexts within *E-Identity*. To distinguish them from the observed ones in Table 1, we mark them with +(i) standing for "inquired about and performed".

5.1 Management and Administration

E-Identity has implemented almost all the activities listed in the *Management and Administration* practice. As shown in Table 1, we could observe that all except for two activities were implemented. At the moment of writing this paper, the company could not evaluate the transition process postmortem due to the fact that the transition project had not yet been finalized. Neither could it define any additional manpower resources required for the whole transition process. Due to financial reasons, their resources were restricted to simply what they had.

E-Identity experienced a self-to-self type of handover and, due to the unavailability of the transitionee, it deemed the transition process to be of a very complex nature. For this reason, their transition strategy focused on the following four strategies: (1) *Strategy 1* determining the future transitionees, (2) *Strategy 2* designating a future transition team, (3) *Strategy 3* designing the transition process, and (4) *Strategy 4* establishing ways of transferring knowledge.

Regarding *Strategy 3*, the company decided to structure handover into two phases: (1) the *project parking* phase and (2) the *project resumption* phase. At the moment of writing this paper, the *project parking* phase got finalized and the *resumption phase* had not yet started.

According to *Strategy 2*, not the whole transition team could be designated in advance. Right now, they had a team for conducting the *project parking* phase. This team will get dissolved. New team will be created in the *project resumption* phase. According to *Strategy 1*, the future transitionees will be consultants instead of fixed-term employees. This will substantially reduce project restart time and cost.

Regarding *Strategy 4,* concerning the transfer of knowledge between the transitioners and transitionees, the company was aware that the two teams would not be able to communicate with each other. For this reason, *Strategy 4* dealt with creating a documentation of the company's products, processes, and technology. The documentation would constitute the main channel of communication.

5.2 Maintenance Environment

E-Identity has implemented all but one activities listed in the *Maintenance Environment* practice. The activity of granting the transitionee permission to access hardware/software suites was not implemented. This is because the transitionee has not yet been designated.

The implementation of all the *Maintenance Environment* activities went very smoothly, mainly thanks to the fact that the transition took place within one and the same company. The hardware and software suites and maintenance support suites were already determined and installed. The company did not need to determine any new suites. Neither did they need to assure that the suites matched each other. They all did by default.

5.3 Version and Configuration Management

The company has implemented all except two *EM*^{3} activities for managing version and configurations. The two activities concerned the identification and tracking of the customizable configuration items. The reason for not

implementing them was that the company had only one customer. They did not experience any customization needs.

Regarding the activities that got implemented, we only had the opportunity to observe the accomplishment of their subset. As indicated in Table 1, we observed the complete accomplishment of the activities concerning the management of versions, configurations and baselines (Activities VC1 and VC 2.1). Due to the specific context of the handover process studied, we did not have however the opportunity to observe the establishment of post-delivery baselines (Activity VC2.2).

The company establishes four baselines: *developer test, system test, acceptance test,* and *deployment baselines*. While following the handover process at *E-identity*, we observed an additional baseline that we had not recognized in our model. It is a release baseline. It is a separate release branch that is created during deployment. It includes all the changes made to the software system during deployment.

5.4 Training

Training was regarded as one of the most important practices of the company's handover process. Hence, all the EM^3 training activities had been implemented. However, as shown in Table 1, at the moment of conducting this study, the company only implemented the training activities from T1 to T5.1, the activities focusing on the designation of roles. Regarding the remaining activities, the company will perform them in the *project resumption* phase. Its trainees are the transitionees and the planning for their training focused on creating a thorough system and process documentation on different granularity levels.

The transitionees will be highly responsible for self-educating themselves by studying the documentation that has been created during the handover process.

5.5 Deployment

All the deployment activities as defined in the *Deployment* practice have been implemented *E-identity*. The company has defined and planned the scope and type of the releases, defined installation procedures, installed the system, closed the deployment and planned for future releases. At the moment of our study, we did not have the opportunity to experience the full deployment process to an external customer. We only observed the internal deployment process.

The company had two types of deployment: (1) *internal deployment* of infrastructure API transferred from the *infrastructure development team* to the *application development team,* and (2) *external deployment* of a ready application from the *application development team* to its external end-user customers. The main reason for conducting the internal deployment during handover was to provide an updated and stable version of API to the *application development team* before freezing the system. The *application development team* would then continue their work on developing the application after project parking.

The steps in the internal deployment process studied were (1) establish a deployment branch for the release, (2) compile and verify the deployment branch by performing deployment readiness tests, (3) make changes to the deployment branch code to solve the problems encountered during testing, (4) integrate those changes in the main branch, (5) de-install the former system version, (6) install the new system version, and, (7) install the operational data. Finally, the company closed the deployment by reviewing the whole deployment process and by making sure that it ended in a correct manner.

5.6 Documentation Practice

The company had implemented all the activities in the *Documentation* practice. As indicated in Table 1, some of the activities were however partially accomplished. These concern defining organizational policies for developing documentation standards and creating mechanisms for controlling the quality of system documentation. The reason is that before handover the development team gave priority to meet the delivery deadlines, and hence, they put less emphasis on documentation quality. The documentation was of low quality before starting project parking. As a result, the company decided to develop the documentation standards to be used in the future.

Some other activities could not be observed while conducting our study. These concern sharing documentation standards and documentation repository with the transitionee. The reason is the fact that the transitionee has not been identified yet. However, in normal handover cases, the company shares the repository by default due to the fact that the transitioner is the same as the transitionee.

5.7 Maintainability Management

The company has not fully fulfilled the *Maintainability* practice. As indicated in Table 1, it has not defined any procedures for assessing data maintainability. They claim that the reason is that the system is not yet fully operationalizable. Hence, it does not have any operational data to consider.

The company has only partially defined procedures for assessing system maintainability. This means that it has defined various quality attributes concerning mainly architectural design and coding standards, however, it has not documented them.

Finally, at the moment of conducting our study, the company realized that one important maintainability attribute was missing. It concerned the traceability between the system documentation and code. The system documentation played the most important role in the company's handover process. It was a prerequisite for resuming system development and it was the only source of information for the *project resumption team*. For this reason and for the reason of attending to the traceability problem, the company revised all the documentation.

5.8 Software System Transfer

The *Software System Transfer* practice was added to *EM³: Handover Framework* during this study. Hence, its activities mirror the activities that were conducted at *E-Identify*. It is worth mentioning that the company distinguished between three types of system components. There were (1) stable components ready to be used , (2) components under testing, and (3) components under development.

6 Final remarks

In this paper, we have reported on the results of implementing the taxonomy activities inherent in *EM³: Handover Framework.* Our goal was to observe their implementation and examine its applicability and usefulness in a real-world industrial scenario. The handover was of a self-to-self handover type and it was conducted in a project parking context.

A quick scan through Table 1 shows that almost all of the EM³ activities have been implemented at *E-Identify*. Not all of them, however, were directly observable due to its specific handover case. The activities that could not be observed either concerned general prerequisite handover activities or the activities to be performed in the *project resumption* phase; the phase that the company has not performed yet. Out of the total of EM³'s activities, 66% could be directly observed and 21/% were inquired about. Only 6% were partially performed and as few as 4/% were not performed at all. Finally, 2% of the activities were not applicable and 1% of the activities was not relevant.

Except for a new component practice, *Software System Transfer*, and its activities, our study has not led to any additions of new activities. It has rather led to the confirmation that almost all the *EM³*'s activities were easily applicable in the handover context studied. It has also helped us identify a new context of a handover process where transitioners will never learn to know the transitionees. Finally, it has learned us the following lessons:

- In all transition contexts, one should designate a transition team including the representatives from the transitioners and transitionees. In the context when the transitionee is not yet known and the transition team only includes the transitioner representatives the only communication channel that is possible is a very detailed documentation of the company's products, processes, and technology.
- The specific context of the handover process studied forces the transitionee to be both the trainer and trainee. This means that the new hires will be responsible for self-educating themselves using the documentation created during the *project parking* phase.
- During handover, it is important to keep track of the software system and the health and progress of its components. For this reason, one needs to clearly distinguish between (1) stable components, (2) components under testing, and (3) components under development. Only then one may make decisions on their handover.

Even though *EM³: Handover Framework* has been originally explored within sixty one companies and has shown to be useful in this study, we strongly advise the software community to continue to explore the handover domain and evolve our framework. More handover contexts need be explored and more studies need be done to evaluate *EM³: Handover Framework*. We believe however, that this study has already provided evidence that *EM³: Handover Framework* is on the right path towards providing a fully-fledged support for creating handover process models.

7 References

[1] T. Pigoski. "Practicle Software Maintenance: Best Practices for Managing Your Software Investment", John Wiley & Sons, 1996.

[2] T. M. Pigoski och C. S. Looney. "Software Maintenance Training: Transition Experiences," Proceeding of Conference on Software Maintenance (CSM), 1993.

[3] T. M. Pigoski och J. Sexton, "Software Transition: A Casestudy," Proceedings of *International Conference on Software Maintenance ICSM*, 1990.

[4] I. O. Standardization. "ISO/ IEC 15288, Systems and software engineering- System life cycle processes," IEEE, 2008.

[5] I. O. Standardization, "ISO/IEC 14764:2006, Standard for Software Engineering- Software Life Cycle Processes-Maintenance," IEEE, 2006.

[6] T. Vollman. "Transitioning from development to maintenance", Proceedings of Conference on Software Maintenance, 1990.

[7] J. T. Howell. "Hard Living on Clay Street: Portraits of Blue Collar Families". Prospect Heights, Illinois, Waveland Press, Inc, ISBN 0881335266, 1972.

[8] A. S. Khan. "A Framework for Software System Handover", Stockholm: KTH, Software and Computer systems, SCS, ISBN:978-91-7501-739-6, 2013, http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-122270.

[9] JBoss. "JBoss Applicaiton server," JBoss, 2013. [Online]. Available: http://www.jboss.org/.

[10] G. Hub. "Git open source distributed version control system," Git, 2013. [Online]. Available: http://it-scm.com/.

[11] A. S. Khan, M. **Kajko-Mattsson.** "Taxonomy of Handover Activities", in Proceedings of the 11th International Conference on Product Focused Software, 2010.

Teaching Undergraduates Unix/Linux Shell Design and Implementation at Cameron University

M. Estep, C. Zhao, and J. Carroll
Computing and Technology Department, Cameron University, Lawton, OK, USA

Abstract – Operating Systems is a required core course in the Computer Science degree curriculum [1]. In this article, the authors discuss using a shell project to teach how operating systems interact with a user and execute user commands. The shell project contains four small projects: execute a single command with or without arguments, execute two commands separated by a pipe, execute a command with input file and/or output file, and run a command in the background. This practice provided students an opportunity using the stepwise technique to complete their shell project. Students learned some basic system techniques and skills, and therefore their learning outcomes were enhanced.

Keywords: Operating Systems, Shell, System Programming, C and C++, Java

1 Introduction

Operating Systems (OS) is a required core course in the Computer Science (CS) B.S. curriculum at Cameron University. This course is designed to provide CS students with an overview of hardware and OS, process management, processor management, interprocess communication, storage management, and auxiliary storage management. To improve the quality of an OS course, many different methods can be used in teaching practice at different universities [1, 2, 3]. At Cameron University, specific projects were used to help students understand basic OS concepts and principles, such as shell design and implementation, CPU Scheduler, interprocess communication, and parallel processing. In this article, the authors only focus on shell design and implementation. It is a basic function of an OS to offer the user an interface to communicate with a computer system. Through completing this project, students may have a better and deeper understanding on how an OS interacts with the user, how the user commands are executed, and why the shell is separated from the kernel.

2 Initial Methods

2.1 Project Arrangement

The shell project was divided into four subprojects: execute a single command with or without arguments, execute two commands separated by a pipe, execute a command with input file and/or output file, and run a command in the background. Each subproject is a continuation of prior subprojects except the first one. Each subproject was to be completed in one and half weeks.

2.2 Forming Teams

At the beginning of the semester, students were divided into teams. Each team consisted of a captain and one or two members. The captain was in charge of team activities, such as team meetings, programming assignments, and project integration and testing. Team members worked together to complete their project.

3 Procedures

3.1 Requirements

Each team was required to complete project documents and an executable file. The project documents had to include, at minimum, a readme file, analysis file, and design file. Implementation languages were chosen from C, C++, or Java, and the executable file had to run under a Unix/Linux environment.

3.2 Analysis and Design

3.2.1 Single Command

The resulting program accepts a single command with or without arguments. For example if the user types *ls* at prompt, then the program should list all the files except hidden files. If the user types *ls –a*, and then all files should listed. In either case, the program will come back to prompt after executing the command. An example of an analysis diagram is shown in Figure 1.

ISBN: 1-60132-446-4, CSREA Press ©

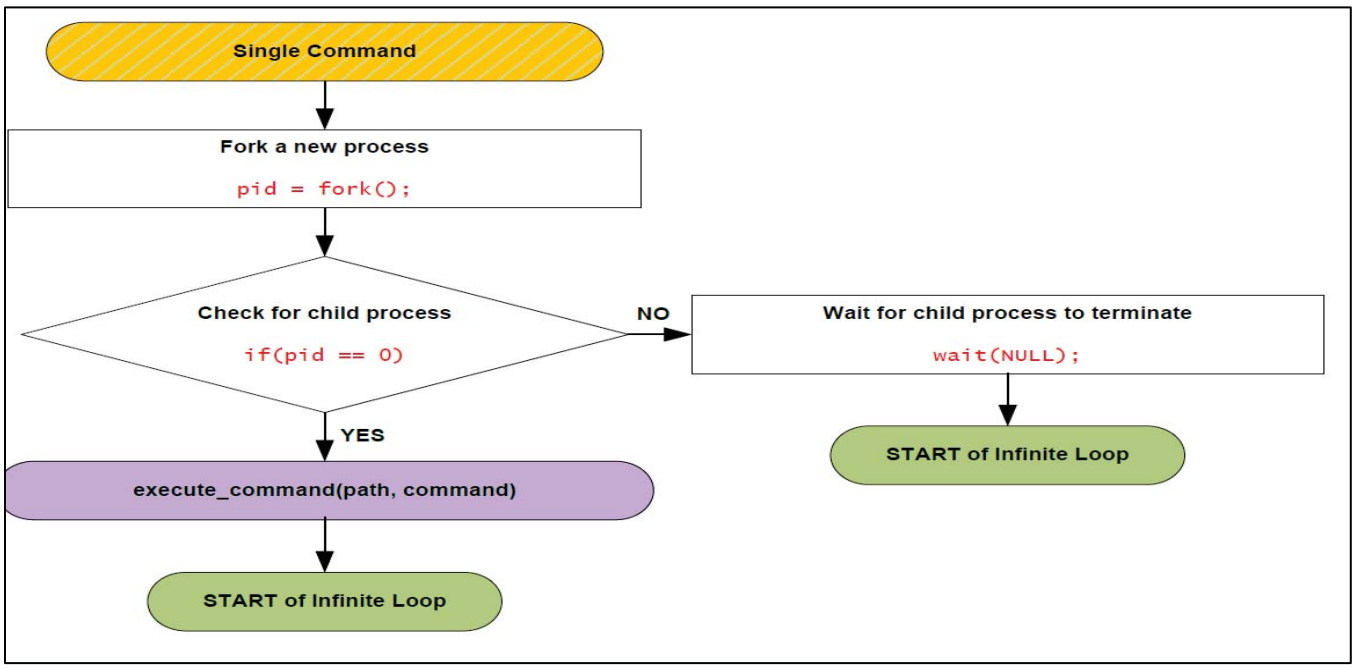


Figure 1. Analysis Diagram of Single Command

3.2.2 Two Commands Separated by a Pipe

The user types two commands separated by a pipe (Case 1). The project forks a child process to execute the first command, and sends the execution results to the pipe. Then the project forks again to execute the second command while reading the execution results from the pipe as input. For example, if the user types *ls –al | wc*, the first child process executes *ls –al* and stores the execution in the pipe. Then the second child process runs *wc* while taking the execution results from the pipe to output word count results. Figure 2 shows an analysis diagram of two commands with a pipe. The pipe redirection function flow chat is shown in Figure 3.

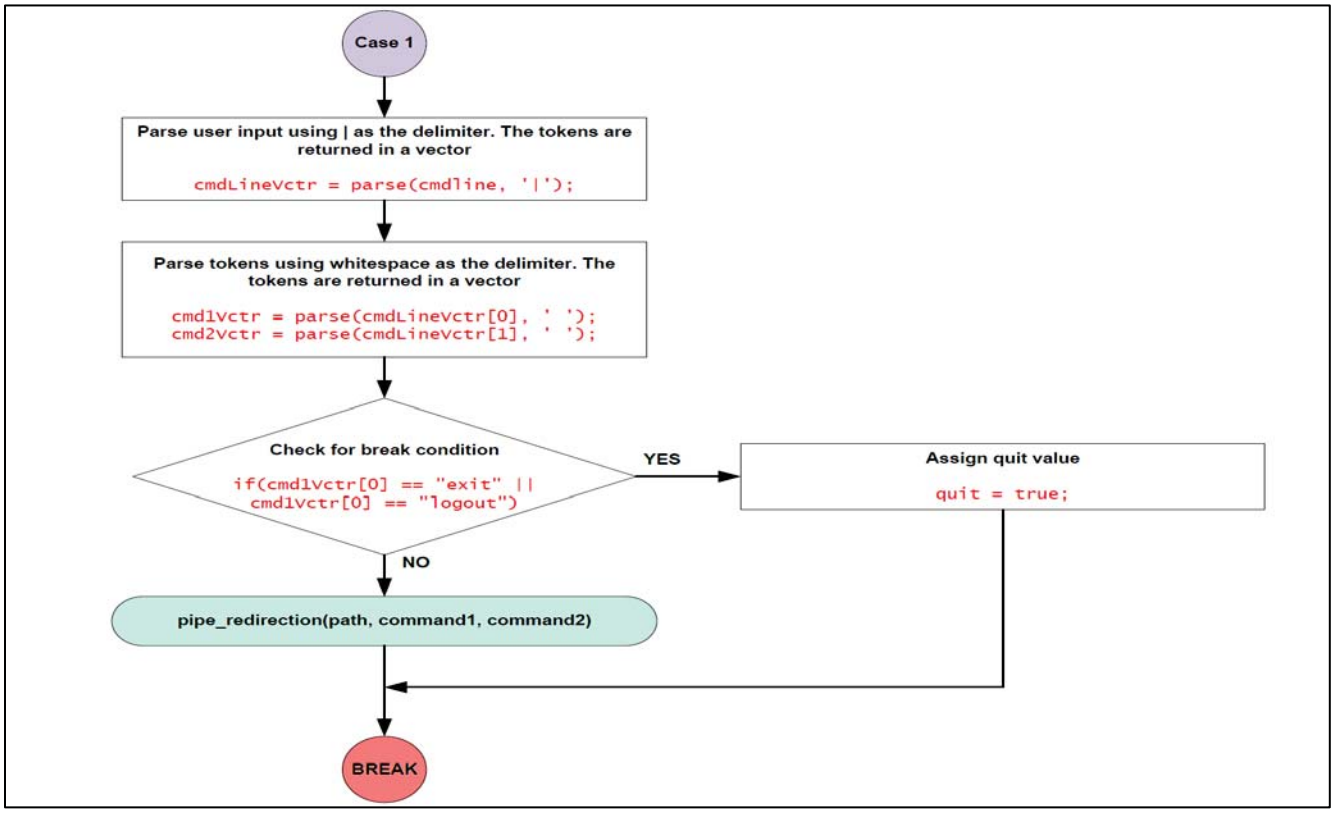


Figure 2. Analysis Diagram of Two Commands with a Pipe

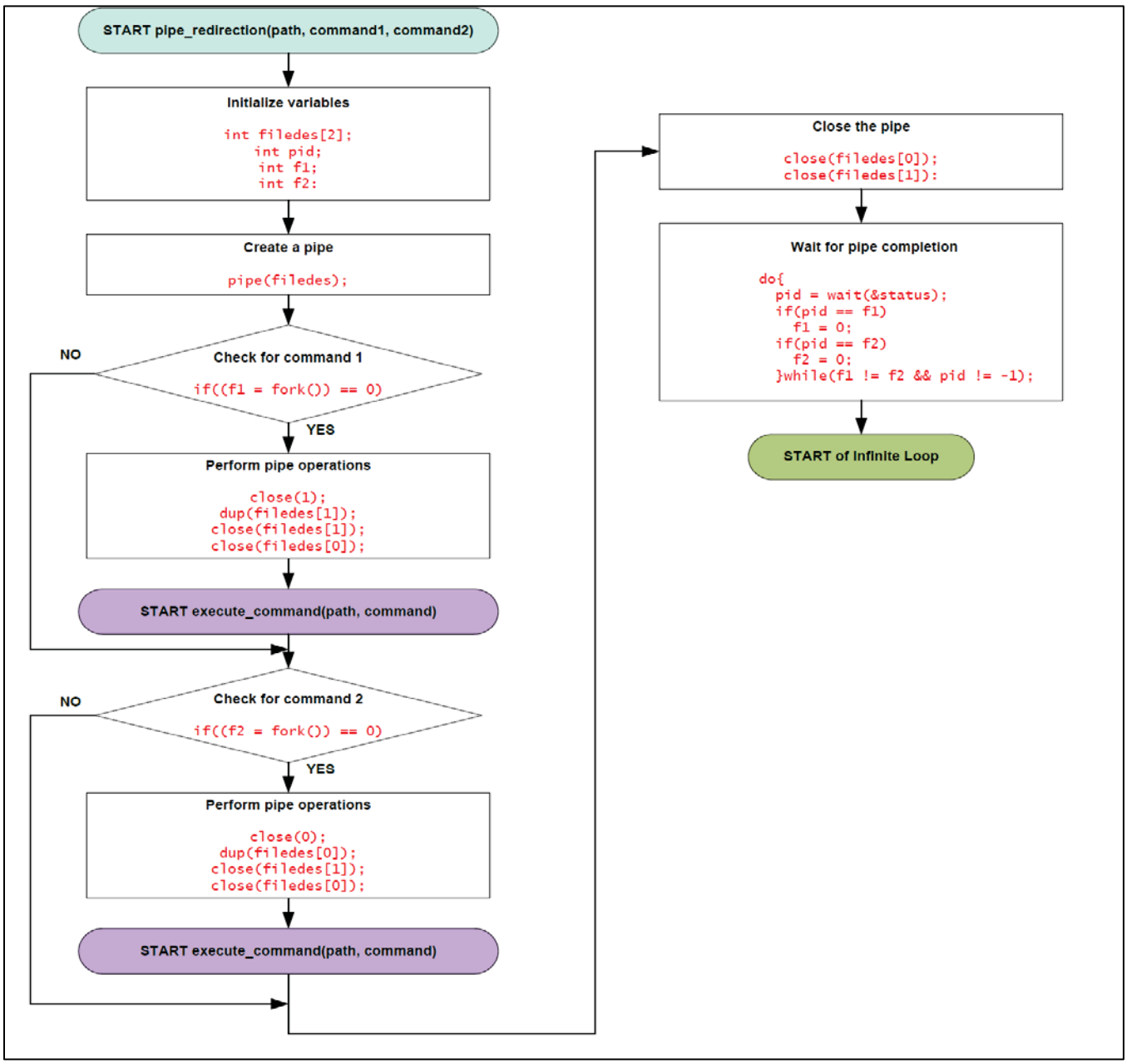


Figure 3. Analysis Diagram of Pipe Redirection Function

3.2.3 Executing a Command with Input File and/or Output File

If either or both redirection operators are used, the system parses the commands accordingly and places them into corresponding vector(s) to more easily facilitate executing the commands later. Redirection operations include two different functions, *in_file (Case 2)* and *out_file (Case 3)* as well as a *both_file (Case 4)* function which is used to handle commands where both redirection operators are being used. Figure 4 shows the analysis and design of input redirection.

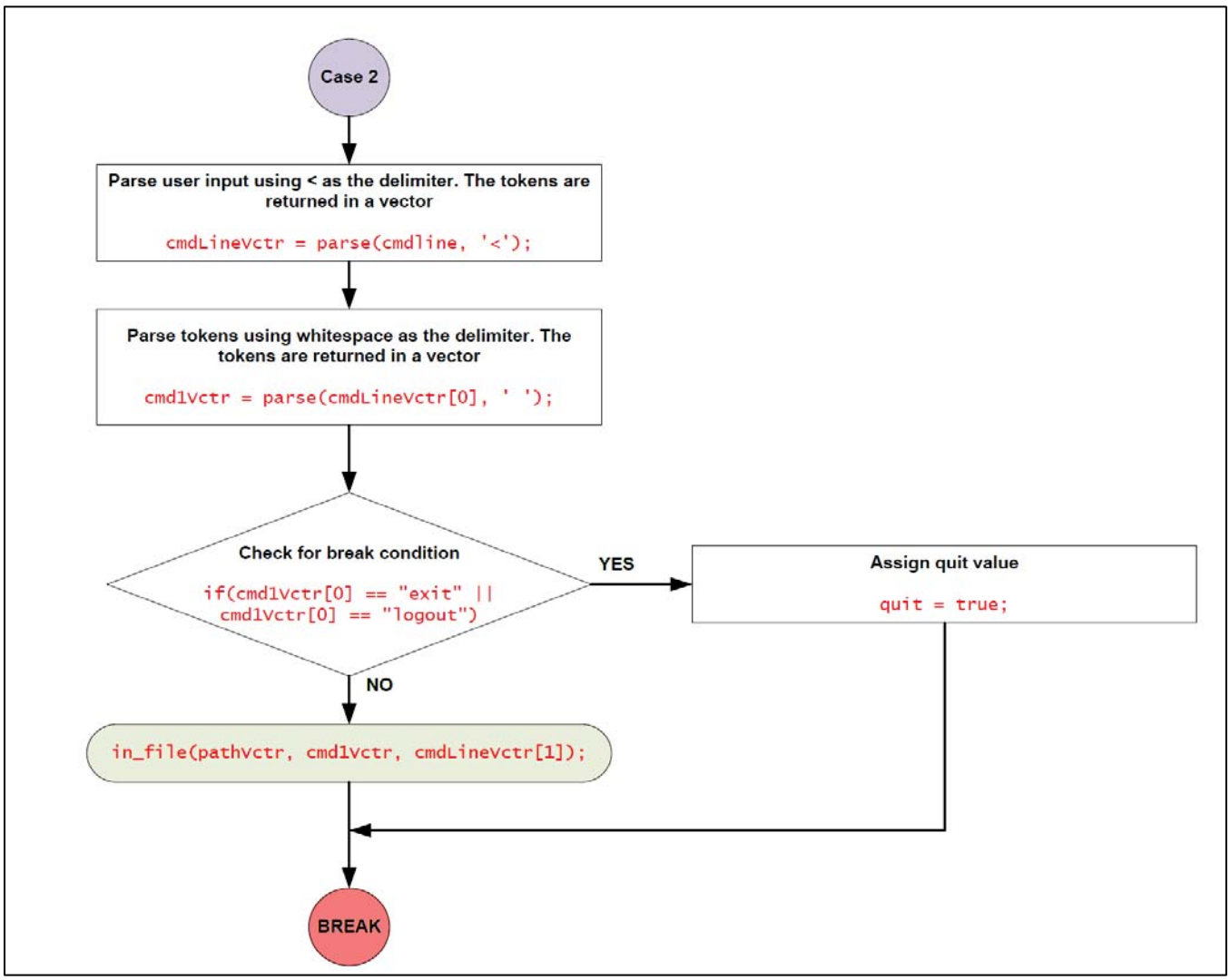


Figure 4. Analysis of Input Redirection

3.2.4 Running a Process in the Background (Case 5)

A background flag is set to true if parsing finds an & operator in the command string. If this flag is set to true, the simulated shell circumvents the *wait(NULL)* as part of the fork() process to enable the command to run in the background, otherwise the fork() proceeds as normal. Figure 5 shows analysis of running a process in the background.

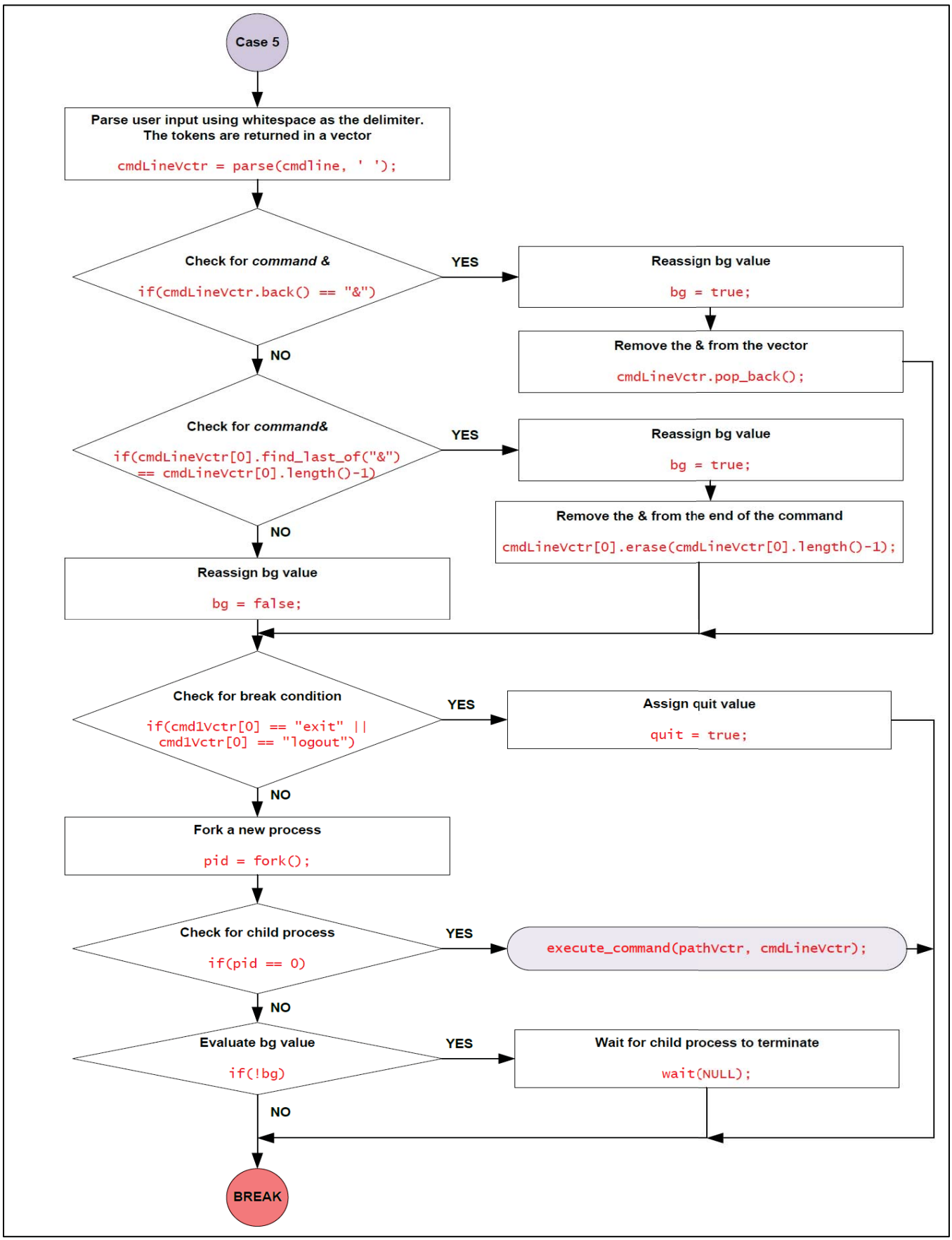


Figure 5. Analysis of Running a Process in the Background

3.3 Implementation

Once analysis and design were completed, implementation took place using C, C++, or Java. The key parts were parsing, executing commands, and piping. The code segment in C++ is shown in Figure 6.

```
void pipe_redirection(vector<string> path, vector<string> cmd1,
vector<string> cmd2)
{
  // Initialize variables
  int filedes[2], pid, f1, f2, status;

  // Create a pipe
  pipe(filedes);

  // First command
  if((f1 = fork()) == 0){
    close(1);
    dup(filedes[1]);
    close(filedes[1]);
    close(filedes[0]);

    // Attempt to execute the command
    execute_command(path, cmd1);

    _exit(0);
  }

  // Second command
  if((f2 = fork()) == 0){
    close(0);
    dup(filedes[0]);
    close(filedes[1]);
    close(filedes[0]);

    // Attempt to execute the command
    execute_command(path, cmd2);

    _exit(0);
  }

  close(filedes[0]);
  close(filedes[1]);

  do{ //waiting for two child processes'termination
    pid = wait(&status);
    if(pid == f1)
      f1 = 0;
    if(pid == f2)
      f2 = 0;
  }while(f1 != f2 && pid != -1);
}
```

Figure 6. Code Segment of Pipe Redirection

3.4 Testing

After completion of the shell project, the students were required to create a test plan and test their project to insure: (1) all functions were working properly, (2) after each execution, the program was back to the parent process prompt to enable the user to execute another command, and (3) fix all logical errors if any.

4 Discussion

- **Shell development is a software engineering process.** During developing software, the students practiced generally five work flows – requirement, analysis, design, implementation, and testing [5]. However, one work flow may be dominant over others in a special software development phase. Seeing this in the shell project can help students truly understand the complexity of the software development process. Meanwhile this project also offered an opportunity to develop and manage a relevant complex project.

- **Improving Communication skills.** To complete the shell project, much oral and written communication has to take place between the instructor and student development team and within a student team. This offers students a chance to improve and enhance their professional communication skills.

- **Enhancing Student learning.** Student learning is one of the core values at Cameron University. The students worked together as a team and not only learned operating system concepts [4] and programming skills and techniques from each other, but also learned how to work with others, which will benefit them in their future profession.

5 Conclusion

The shell project provides the students with an opportunity to design and implement the interface part of a Unix/Linux Operating System, thereby allowing the students to understand how the OS interacts with users and why the OS always forks a child process to execute a user command(s). It also creates an attractive learning environment that motivates the students to go further and dig deeper in the OS field step-by-step. Furthermore this practice can improve the quality of OS and student learning outcomes.

6 References

[1] **Computer Science Curricula 2013 Curriculum Guidelines for Undergraduate Degree Programs in Computer Science** December 20, 2013. The Joint Task Force on Computing Curricula Association for Computing Machinery (ACM) IEEE Computer Society Computer Science Curricula 2013.

[2] ***Teaching Operating Systems Using Code Review***, Christoffer Dall, Jason Nieh Proceedings of the 45th ACM Technical Symposium on Computer Science Education, March 2014

[3] ***Teaching Operating Systems Using Android, Jeremy Andrus***, Jason Nieh Proceedings of the 43rd ACM Technical Symposium on Computer Science Education (SIGCSE 2012), February 2012

[4] ***Teaching Operating Systems as How Computers Work***, Peter Desnoyers Northeastern University, May 12, 2011. http://www.ccs.neu.edu/home/pjd/papers/fp144b-desnoyers.pdf

[5] Operating System Concepts, 8th Edition, Silberschatz, Galvin, and Gagne. John Wiley & Son INC. 2009.

[6] ***Object-Oriented and Classical Software Engineering***, 8th Edition, Stephen R. Schach, McGraw Hill, 2011.

A Framework Based on Image Processing Techniques for Providing Synchronization in Smart TV

Cédrick Bamba Nsimba[1] **and Alexandre luis Magalhães Levada**[2]
[1]Department of Computer Science, Federal University of São Carlos, São Carlos, São Paulo, Brazil
[2]Department of Computer Science, Federal University of São Carlos, São Carlos, São Paulo, Brazil

Abstract—*In smart TV, we can naturally observe a lack of connection between applications and the content of the tuned programming on TV set and TV broadcasters have fully control over the transmitted content. Based on the fact that computer vision and machine learning tools can provide that synchronized information by, respectively, processing the image frame of TV content and then classifying them, new opportunities will be opened up for developing a bunch of new interesting applications aiming to promote the user interaction level in TV. Moreover, with the extensive use of mobile devices and computers in today's life, it will have new user interaction possibilities and a new business model will be emerged. In this paper, the smart TV framework is presented and evaluated. The objective of this study is to facilitate the developers to implement applications in this area without being concerned about low-level implementation details.*

Keywordst: TV channels monitoring, image classification, multimedia synchronization, detection of TV channel logos, smart TV

Type of the submission: Regular Research Paper

1 Introduction

Smart TV is changing the way that people watch television and providing to the user a possibility to interact with TV contents. However, unlike this type of interaction, most of the time, the audiovisual content of programs in the smart TV does not communicate with the native TV applications and other TV platform features. The only basic interaction is the possibility to switch between the things are displaying on the TV screen either program or TV applications.

Bachelet [1] conducted a research in five European and North American countries in May 2013 and found out that only less than the half of 6115 smart TV owners who were interviewed, connected their TVs to the Internet. According to his research, two elements are the reasons that why the smart TV has not yet become popular and been widely accepted by consumers as much as smartphones: (a) The lack of content and interesting applications: although the majority of smart TVs offer a wide range of content and applications, most of them are irrelevant and are not interesting to users; and (b) The Poor User Interface: a lack of rich user interface that can integrate TV applications with its audiovisual contents.

Schofield [12] mentioned, "If TVs are going to be truly smart they must do more than offer a wide variety of online video services. Instead they must add advanced functionality including voice control, motion control, advanced advertising, attractive user interfaces and two-way communications with other smart devices –so-called 'second screens'– allowing these devices both to send video to the TV and know what is being watched. Manufacturers should focus less on adding more content and more on improving how users can interact with that content".

This suggests that new synchronization mechanisms, also interaction with TV environment can help to improve the user experience. Information related to the channel being watched by the user can be retrieved automatically through a clear approach using TV channel logo detection, for example. Then, this information can be used to notify all connected devices, in the environment of smart TV, about the channel being watched by the user. Channel logos are visual objects (name, symbol or trademark) designed with the purpose of easy recognition and are important to assess the identification of a TV channel.

One of the finding mentions that innovative interactivity has not been promoted and used adequately in smart TV platforms. In addition, they show that the lack of facilities that allow the integration of Smart TV applications with their own audiovisual content is a major reason for its limited use. Consequently, the current proposed paper uses image processing to promote the development of smart TV applications synchronized with TV program contents.

2 Related Work

Several authors reported some results of building frameworks designed to support TV applications development. However just few of them focus on reuse in smart TV applications, specifically those ones synchronized with television programming.

Group Share-TV [6] proposes a framework called share-TV used for the development of converged applications centered on TV for GoogleTV and Ginga-J platforms. The share-TV allows the development of TV applications that include a generic mobile application. Ever since this generic application get downloaded from a given available TV IP,

installed and run on mobile device, the communication with the TV convergent application will be initiated automatically in order to register and receive shared objects. While the objects are being received, the device show them on the screen allowing interaction on them. Compared to the work reported in this paper, share-TV also provides communication services and reuse, however, this framework is limited to applications based on Google TV and Ginga-J platforms. The main difference is that share-TV is used to develop TV convergent applications while the work reported in this paper focuses on building smart TV applications synchronized with the content of TV programs.

Samsung Smart TV [11] presents a framework called AppsFramework. This framework encapsulates reusable modules for scene management, video playback / music, and so on. This makes it easier for the developer of smart TV applications to avoid performing complicated sequences of calls to the operating system in order to manage scenes (focusing, showing and hiding events) of an application, for instance. Some of these modules were used in the framework proposed in this paper.

Freitas and Teixeira [5] proposed an architecture for supporting the development of ubiquitous applications in home networks focusing on Digital TV. The proposed architecture consists of communication interface with home devices, a protocol layer for automatic service discovery, and so on. Although the architecture is designed to be implemented in iDTV middleware, some of its reusable artifacts such as the aforementioned ones were used and implemented in the framework proposed in this paper.

The framework for building synchronized smart TV applications with TV programs presented in this paper is different with all the aforementioned works from the scene that it uses image processing for synchronization purpose and it is a reference framework that allows more efficiency and less cost in building multi-platform applications in this field.

3 Synchronization

Teixeira et al. [13] studied the synchronization in the context of multimedia applications and considered that synchronization is a mechanism to guarantee that actions can happen according to the defined time reference set by a clock or established by the occurrence of events. It considers the tolerances that vary according to the type of application. In case of TV program, the characteristic of event is to be considered as a reference that is one of the important aspects of synchronization in the context of this study.

In order to provide synchronized applications with television programming, it is essential to know which channel the viewer is watching and what is being presented to the audience every moment. Then this information can be published to stakeholders and used to promote synchronization between program and application through notification services implemented in this work. Notifications addressed in this work have various types such as channel identification, start and end of trading blocs, start, pause and end of the TV programs, sex scenes, violence, crime and some notifications generated by the events triggered by the user.

The synchronization module of this framework have API's that allow applications to access notifications related to events occurring in TV programs. All these synchronization signals are generated by image and audio processing techniques. In this paper, only that synchronization mechanism is discussed which implemented by SURF, K-NN, K-Means and template matching techniques for channel identification.

3.1 Synchronization based on SURF

The main idea of this part is to, automatically, figure out what is the content of extracted frame from smart TV video. Although this problem of object detection and recognition in videos are seen as the frames sequence analysis, it can be simplified to single image analysis. Consequently, the main issue related to this task is how to compare two images (video frame and TV channel logo). We need to know if the logo image is present in the video frame. To do so, and after a research in the literature, we decide to use the SURF [7, 8] (*Speed-Up Robust Features*), an image features detector and descriptor inspired by SIFT [4] (*Scale-Invariant Feature Transform*). This decision took in consideration a precision and fast computation speed of SURF achieved by: (a) use of integral images for image convolutions; and (b) use of hessian matrix-based measure for detector and a distribution-based descriptor. Given an input image (video frame) and its extracted SURF features and corresponding descriptors, we need to match its descriptors with the logo images in the collection. With the huge number of logo descriptors in the collection, we need to avoid comparing with entire collection. For this, we use some segmentation techniques to divide all the features in sub-groups by common properties.

The process entails two sub-tasks: (1) data collection (Fig. 1(c)), a set of TV channel logos pick up from LyngSat [9] logo collection where for each logo class we defined the number of logo images and frames for training and testing phase; and (2) the real application (Fig. 1(e)). The first part contains the following tasks: (a) *feature extraction:* key-points are extracted from all logos (and all test frames) using SURF. These key-points are used for comparison; and (b) *feature segmentation:* receives large set of key-points as data features, extracted from TV channel logo collection, and segments all of them to obtain groups of common features.

The second part (Fig. 1(e)) is retrieving the logo (s) from an input image. This procedure has the following steps:
(a) *features extraction:* extracting the key-points of a video frame using SURF; (b) *classifying the features:* with all the features of a frame, we need to find which logo segment

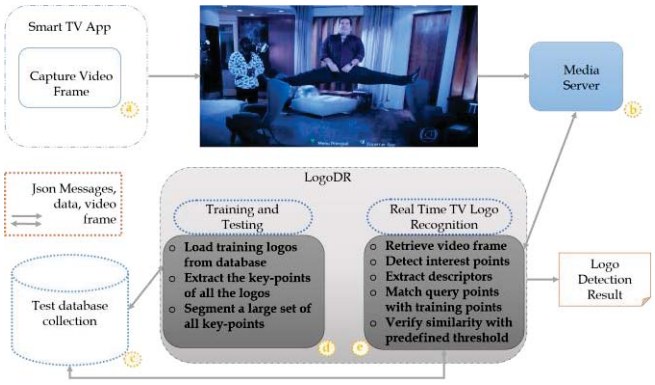


Fig. 1. Real Time TV Channel Logo Recognition

matches the key-point better. All the logos point in those segments are possible candidates; (c) *ranking the candidates:* with all the candidates presenting some common key-points with the video frame, we need to rank these key-points by a determined threshold and keep only the similar ones; and finally (d) *selecting the logo:* after ranking the candidates, we have all the information to decide whether there is a logo inside the video frame or not and which one is contained.

3.1.1 Features Extraction

To extract features from each video frame, we used SURF algorithm implemented with OpenCV [3]. SURF is sufficiently fast for real time object recognition. For feature vector matching purpose, as showed in Fig. 3, we used K-Nearest Neighbors, Fast Library for Approximated Nearest Neighbors, distance ratio rule for finding all good matches and a threshold used to verify if the number of good matches found is enough to recognize a presence of the logo inside the frame. Some results of our experiments are shown in Fig. 4.

3.1.2 Collection Segmentation

After applying SURF algorithm and collecting a large dataset composed with TV logo vectors, we used K-Means clustering method to split this dataset in sub-groups containing common properties. K-means is a clustering method, widely used for partition problem, where given a dataset of n points, the objective is to divide this dataset in k groups finding k centroids. In this work, we partitioned the logo dataset in k vectors called centers or centroids. To obtain these centroids, we calculated the distance of all logo vectors to each k initial centroid during each iteration. Then, each logo vector belongs to its nearest centroid. We repeat this operation for a number of iterations where in each loop the groups are more balanced. By achieving the convergence, the process finishes. By convergence, we mean having no more change in the distribution of each group.

Fig. 2. Example of TV logo detection with τ (distance ratio) parameter equal to 0.95

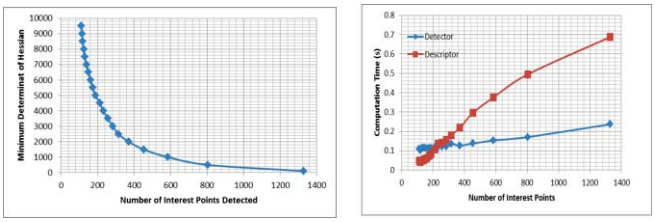


Fig. 3. Different minimum determinant of Hessian values with corresponding number of detected interest points and corresponding computational time of feature detector and descriptor tasks are shown. Settings used are det(H) = 2000, τ = 0.95 and good_matches = 6.

After computing all the centroids, the TV logo collection is segmented in a way that for each SURF vector there is a corresponding centroid, which can be accessed directly through the indexing mechanism shown in Fig. 5. For training the model, we used the data in Fig. 2 and more data encountered in LyngSat logo collection.

3.1.3 Classification and Ranking

The objective of this section is to use the K-Means and the relative segmented logos collection to classify the input video frame. To do so, we extracted the SURF vector from the frame and defined which the best group that contains elements more similar to the input SURF vector.

After applying the K-Means technique in the segmentation step, the result was a list of centroids that are, simply, the vectors which represent each group. Then, as it is shown in Fig. 6, we matched each frame SURF vector to all centroids in the K-Means model and we only kept the most similar one with the highest value of match.

Once we have these classes, all the candidates are ranked using the distance metric between the SURF points.

3.2 Synchronization based on template-matching technique

Multimedia synchronization, especially in smart TV environment, is tightly related to a synchronization of TV program video with smart TV applications. Through the image processing, anchors – associated with image transmitted by TV or parts of it - linked to the content can be defined. This technique allows realizing an image

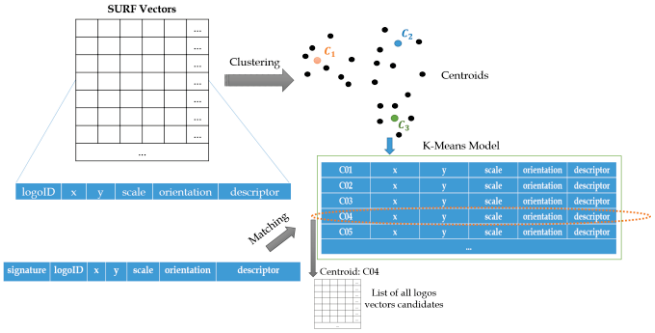


Fig. 4. K-Means segmentation and classification processes

processing, trying to find a similarity between images. Consequently, in the context of this study, it was possible to perform a more optimized search, trying to find points of synchronization inside the video frame, for example a TV channel logo, in our case.

With the extensive research on new and innovative techniques for multimedia indexing, actually we can find an OpenCV library, widely used in computer vision applications, which implements some of these algorithms. In this work, we used the CV_TM_CCORR_NORMED OpenCV method to compare smart TV video frame (query image) with the TV channel logo (reference image). This process is shown in Fig. 5.

4 Framework Architecture

Bosch et al. [2] report that framework development is different from a common application development. This is because of that framework's design needs to cover all relevant features of a particular domain and not just those ones of specific application. This is why it is important to consider the following when developing framework for smart TV integrated applications. Generally, there are six issues to consider. First, How to synchronize the TV content with Smart TV applications (Fig. 6(1)), Second, how mobile devices (smartphones, tablets, and others), which are in the same space with TV can interact with it (Fig. 6(2)). Third, how connected mobile devices in the TV environment can detect the presence of available TV service for use (Fig. 6(3)). Forth, how Smart TV applications can act over the TV controls such as changing channels, controlling volume and so on (Fig. 6(4)). Fifth, how ticker applications must share the remote control with the TV (Fig. 6(5)). Finally how a Smart TV application can identify the channel, which is being watched by the user (Fig. 6(6)). This last module was implemented using the TV channel logo detection presented in the section 3.

5 Framework Validation

To validate the framework proposed here we first developed a set of tools to implement test with developers and then used the experimental method proposed by Wohlin et al. [14]. The set is composed of two main parts: the Back-

Fig. 5. Template-matching steps

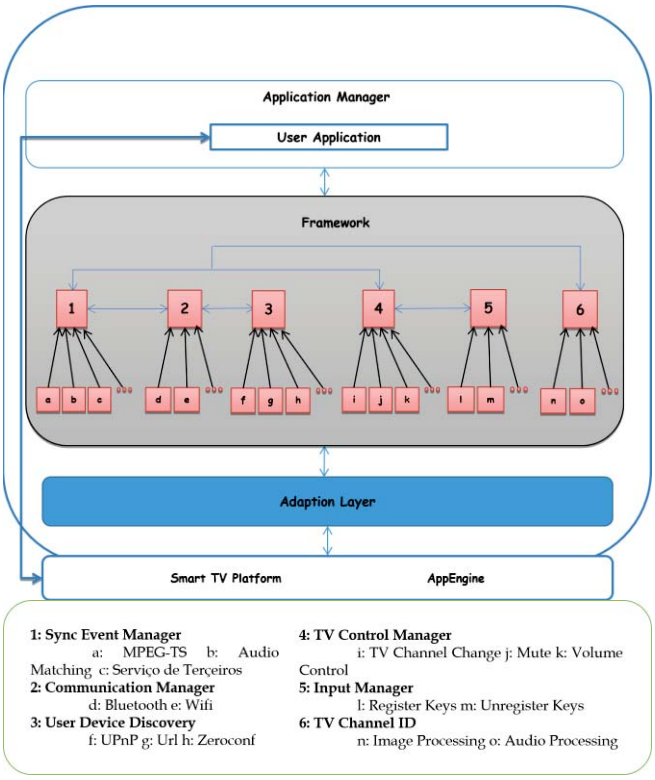


Fig. 6. Framework Architecture

end (Fig. 7(a, b, c)) used to store the information of users and devices and to allow communication and sharing of media content among different components of the set; and the Front-end (Fig. 7(d, e)), which contains applications executed on mobile devices and Smart TV platforms.

The *Back-end* of this work provides web services for smart TV discovery services and for managing the users of social TV systems. The aforementioned services were developed using the Grails framework (Fig. 7(a)). Moreover, the *Back-end* allows communication and sharing the media content among the different applications

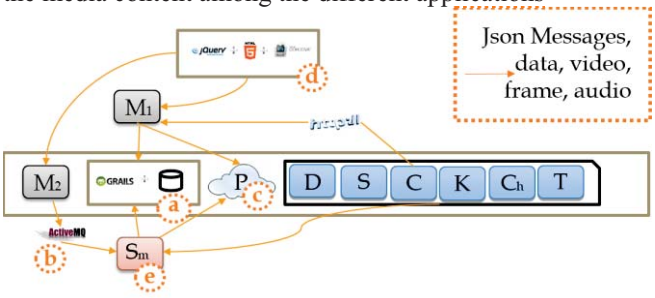


Fig. 7. Framework Instantiation Architecture where M1: Mobile Application 1; M2: Mobile Application 2; Sm: Smart TV Application; P: PHP Server; D: Discovery; S: Synchronization; C: Communication; K: Keymanagement; Ch: Channel identification and T: TV.

respectively using Apache ActiveMQ message broker (Fig. 7(b)) and PHP server (Fig. 7(c)).

The *Front-end* is based on client-side (PC, smart TV, Tablet, smart Phone, etc.) that was developed using Apache Cordova + HTML5 + JQuery (Fig. 7(d)) and JavaScript-based smart TV framework API's.

The objective of our experiment was:

- **To analyse** the use of the proposed framework in the construction of smart TV applications synchronized with television programing;
- **With the purpose** of evaluation
- **Regarding** the efficiency in terms of time spent and productivity;
- **From a point of view** of software developers;
- **In the context** of undergraduate and graduated in computer science and computer engineering. It is important to point out that in this experiment twelve (12) developers and one object were considered (*TVMonitor: standard synchronized smart TV application*).

The experiment consisted of a comparative study of the development processes of two versions of a standard synchronized smart TV application, one built with the reuse approach using the proposed framework and other built without this approach. We formulated three hypotheses and considered some metrics during the experiment.

For the formulation of the three hypotheses, the following metrics were considered:

τ - Total time spent by the team for developing smart TV application synchronized with the TV program;

Þ - Team Productivity in terms of produced lines of code (LOC) per unit time (Þ = LOC / τ);

μ_τ - Average of the spent time by the teams for developing smart TV application synchronized with the TV program;

$\mu_Þ$ - Average productiveness of the teams in the development of smart TV application synchronized with the TV program.

We have a null hypothesis and its two corresponding alternatives:

- ***Null Hypothesis (H_0):*** There is no difference between teams who used the proposed framework and teams that did not use while developing *TVMonitor* application regarding the efficiency (ε) of the team.
  H_0: $\varepsilon_{framework} = \varepsilon_{withoutframework} \Rightarrow \mu_{\tau framework} = \mu_{\tau withoutframework}$ e $\mu_{Þframework} = \mu_{Þwithoutframework}$
- ***Alternative Hypothesis (H_1):*** Teams who use the proposed framework for building *TVMonitor* application are generally more efficient than those ones who developed without the use of framework.
  H_1: $\varepsilon_{framework} > \varepsilon_{withoutframework} \Rightarrow \mu_{\tau framework} < \mu_{\tau withoutframework}$ e $\mu_{Þframework} > \mu_{Þwithoutframework}$
- ***Alternative Hypothesis (H_2):*** Teams using the approach "*without framework*" for building **TVMonitor** application are generally more efficient than those developing with the use of framework.
  H_2: $\varepsilon_{framework} < \varepsilon_{withoutframework} \Rightarrow \mu_{\tau framework} > \mu_{\tau withoutframework}$ e $\mu_{Þframework} < \mu_{Þwithoutframework}$.

For conducting our experiment, we prepared effectively the material needed to support the process, that means, the set of objects manipulated during the experimentation and some documents that allowed the experimenter to exchange information with the participants.

The organization of the data collected during the experiment in Fig. 8 is done according to the two development approaches used in this experiment: development with and without the use of the framework reported in this paper.

An initial analysis was done on the data collected in Fig. 8. It is important to note that the distribution efforts of the groups in the design and test phases of *TVMonitor* development was constant. However, there is a great discrepancy of development efforts for the groups that used the proposed framework during the implementation of *TVMonitor*. While groups which have not used the proposed framework, in average they spent 4 hours and 59 minutes but those who used the framework spent 2 hours and 08 minutes (a decrease of 57, 2%).

Finally, we tested our hypotheses using the t-test [10] which aims to verify that a variable differs between two independente samples, based on the arithmetic average and considering variability of its data items. Then with some degree of significance (α), reject the null hypothesis (H_0) and choose one of the alternative hypothesis (H_1 or H_2). The t-test formula is given by equation (1), where Sx^2 and Sy^2 are the variances of each sample; *Sp* is the dispersion; and *n* and *m* are the numbers of data items that each sample contains. In equation (2), n+m-2, typically noted by gl is called the degree of test's freedom.

$$t_0 = \frac{\bar{x}-\bar{y}}{Sp\sqrt{\frac{1}{n}+\frac{1}{m}}} \qquad (1)$$

a)

$$Sp = \sqrt{\frac{(n-1)Sx^2+(m-1)Sy^2}{n+m-2}} \qquad (2)$$

Once you have calculated t_o, α and gl, you can check the value of the standard t in t-test distribution to see if t_0 is so significant.

If $|t_0| >$ standard $t = t_{\alpha/2\,gl} \rightarrow$ **REJECT H_0**,
Otherwise, $\rightarrow$ H_0 **NOT REJECTED** and no conclusion is drawn from the experiment.

As the dependent variable of the experiment (efficiency of teams) has two treatments (total time (τ) and productivity

	Group	StartTPro	FinTPro	StartTImp	FinTImp	StartTTest	FinTTest	LCG Auto	LCG Man	TLOC	TT(τ)	TPrd(p)
With Framework	G1	11:20	11:46	12:40	15:21	15:27	16:18	3270	52	2532	3:58	838
	G2	11:56	12:13	12:47	15:19	15:25	16:04	2826	61	2541	3:28	833
	G3	14:30	14:47	15:00	16:10	16:20	16:30	2480	56	2536	1:37	1569
	Average	00:20		2:08		00:33					3:01	1080
Without Framework	G4	14:20	14:51	15:23	18:05	18:10	18:25	1882	361	2243	4:08	543
	G5	8:30	9:10	9:20	15:17	15:25	16:12	1882	427	2309	7:24	312
	G6	8:30	9:02	9:10	14:48	15:00	15:33	1882	392	2274	6:43	339
	Average	00:34		4:59		00:32					6:05	398

StartTPro: The Start Time of the Project;
FinTPro: The Finish Time of the Project;
StartTImp: The Start Time of the Implementation;
FinTImp: The Finish Time of the Implementation;
StartTTest: The Start Time of the Test;
FinTTest: The Finish Time of the Test;

LCGAuto: Lines of Code Generated Automatically;
LCGMan: Lines of Code Generate Manually;
TLOC: Total Lines Of Code;
TT: Total Time;
TPrd: Total Productivity.

Fig. 8: Data Collected

(Þ)), the application of t-test was performed in two steps too. During this process, we calculated the variance using the following equation:

$$Sx^2 = \frac{\sum_{i=1}^{n}(x-\bar{x})^2}{n-1} \quad (3), \text{ with } n = 3$$

Step 1: t-test (Total Time)

After calculating the variance of each group, we have:

Sx^2 (withoutframework)= 2,97723 Sx^2 (withframework)= 1,5325
α= 0,2 Sp = 1,501620791012165
$t_{\alpha/2,\,gl} = t_{0,1000,\,4} = 2,1318$ t_0 = -2,498498775014366

Then we have $|t_0| > t_{0,1000,\,4}$ **REJECT the null hypothesis H_0 with 20% of significance.**

Step 2: t-test (Total Productivity)

After calculating the variance of each group, we have:

Sx^2 (withoutframework)= 15951 Sx^2 (withframework)= 179347
α= 0,02 Sp = 130,1691207621838
$t_{\alpha/2,\,gl} = t_{0,01,\,4} = 4,6041$ t_0 = 5,475964737075994

Then we have $|t_0| > t_{0,01,\,4}$ **REJECT the null hypothesis H_0 with 2% of significance.**

6 Final Remarks

The proposed framework can be offered to developers in form of a semi-complete source code skeleton that integrates synchronization, notification and TV controls functions. A set of tools that were developed in section 5 can provide to a developer more facilities in the process of building Smart TV synchronized applications. In addition, the experiment conducted in this paper could prove statistically that the proposed framework can be considered as an important tool to support the developers of applications in this field. In addition, all functional requirements that were established while planning the development of this framework were implemented and used during the instantiation of the proposed framework.

Regarding future work, we plan to: (a) add mechanisms of synchronization through local audio/video processing of TV content to framework; (b) explore and add adjustment mechanisms of synchronization with the purpose of minimizing the delay difference that exists among various forms of television content transmission (radio broadcasting, cable, satellite, etc.); and (c) offer more notification API's for supporting the development of social TV systems with the purpose of improving the user experience quality in Smart TV environment.

7 References

[1] Bachelet, C. Most smart-TV owners do not connect their TVs to the Internet: manufacturers must respond. Analysys Mason 2013. Available at: <http://www.analysysmason.com/About-Us/News/Insight/smart-TV-May2013/>. Accessed on: march. 2014.

[2] Bosch, J; Molin, P; Mattsson, M; Bengtsson, P; Fayad, M. Framework Problems and Experiences. In: Fayad, M.; Johnson, R.; Schmidt D. Building Application Frameworks: Object-Oriented Foundations of FrameworkDesign.Nova Iorque : John Willey and Sons, p. 55-82, 1999.

[3] Bradski, G.; Kaehler, A. Learning OpenCV: Computer vision with OpenCV library. 1. Ed: O'Reilly Media, 2008.

[4] D.G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal on Computer Vision*.2004.

[5] Freitas, G; Teixeira, C. Uma arquitetura de serviços para aplicações ubíquas em redes domésticas centrada em TV digital. In: *XVI Simpósio Brasileiro de Sistemas Multimídia e Web (Webmedia 2010)*, 2010, Belo Horizonte - MG. Anais do XVI

Simpósio Brasileiro de Sistemas Multimídia e Web (Webmedia 2010). Porto Alegre: SBC, 2010.

[6] Group Share-Tv. Share-TV: Um framework para desenvolvimento de aplicativos convergentes centrados na TV para as plataformas GoogleTV e Ginga-J. Webmedia '12. São Paulo, 2012.

[7] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)", *Comput. Vis. Image Underst.*, vol. 110, no. 3, pp. 346-359, 2008.

[8] H. Bay, T. Tuytelaars, and L. J. V. Gool, "Surf: Speeded up robust features.," in ECCV (1) (A. Leonardis, H. Bischof, and A. Pinz, eds.), vol. 3951 of *Lecture Notes in Computer Science,* pp. 404-417, Springer, 2006.

[9] LyngSat. TV Logotypes by Country. Available at: <http://www.lyngsat-logo.com/tvcountry/tvcountry.html >. Accessed on: may. 2015.

[10] Montgomery, D.C. Design and Analysis of Experiments. 5 ed., Wiley, 2000.

[11] Samsung Smart Tv. AppsFramework. Available at: <http://www.samsungdforum.com/Guide/art00017/index.html>. Accessed on: may. 2014.

[12] Schofield, J. Smart TVs may be taking off, but they're still not smart enough. ZDNet, 2012. Available at: <http://www.zdnet.com/smart-tvs-may-be-taking-off-but-theyre-still-not-smart-enough-7000008042/ >. Accessed on: march. 2014.

[13] Teixeira, C.A.C.; Cédrick, B.N; Santos, C.A.S, Melo, E.L. Mechanisms of synchronization for multimedia applications. 2015.

[14] Wohlin, C; Runeson, P; Host, M; Ohlsson, M.C; Regnell, B; Wesslén, A. Experimentation in Software Engineering: an introduction. Kluwer Publishers, 2000.

A Meeting-Oriented Process for Streamlining Business Collaboration: A Conceptual Example of Software Development Case

Chung-Yang Chen [1], **Kuo-Wei Wu**[2], **Jung-Chieh Lee**[1]
[1] Department of Information Management, National Central University, Tao-Yuan, Taiwan, ROC
[2] Department of Industrial Engineering, National Taiwan University, Taipei, Taiwan, ROC

Abstract *- Human beings play a critical role in any collaborative business activities, including software development. People and people-derived issues such as communication and teamwork efficiency and effectiveness are critical to their success. However, managing teamwork and stakeholder' involvement is a challenging work, especially when we often see that a technical job may be easily done, it requires substantial efforts for relevant stakeholders on brainstorming with people, reviewing the work, explaining and convincing the work to clients/reviewers, etc. To help resolve this human-side collaboration and communication issue, this paper focuses on software development project as an example and draws an attention on project meetings, and illustrates a meetings-flow approach. In this preliminary study, we would show you the innovative definition on meetings in order to help model and streamline the collaborative proceeding of software development.*

Keywords: business collaboration, project meetings, software development

1. The Human Side of Software Development

Contemporary software development (SD) heavily requires the participation of various stakeholders and parties in accomplishing ad-hoc project tasks. Project activities such as feasibility analysis, presentation rehearsal, requirements exploration, critical artifact review and acceptance, project monitoring and control, change control, conflict resolution, etc. are performed in the form of group discussion and social presence to announce, brainstorm, negotiate, reach consensus, leverage peer pressure, and present or report works under public scrutiny. Stakeholders' involvement in software development, though not technical, contributes to the success of a project.

The importance of stakeholder involvement has been evidenced by many studies, e.g. (Faraj and Sambamurthy, 2006; Natale and Ricci, 2006; Marchewka, 2010; Standish Group, 2007). Unfortunately, managing people is not easy, and it gets more difficult in software development that particularly requires teamwork among the stakeholders (Crocitto and Youssef, 2003; Dennis and Garfieldll 2003; Faraj and Sambamurthy, 2006; Hong et al., 2004; Natale and Ricci, 2006; Probert, 1997). According to Standish Group's chronicle reports, the stakeholder involvement problem continues to majorly cause software projects to fail (Marchewka, 2010; The Standish Group, 2007). These human-side issues of stakeholder involvement in collaborative software development should be emphasized and further integrated into mainstream methods and tools (FinstAM, 2003).

In a collaborative project that develops integrated products and processes, since it involves complex people configuration and participation, the stakeholder participation become the critical path (Chen, 2011; Roberts et al., 2002). From the process aspect, known project management and software process standards, such as the Project Integrated Management in the PMBOK and the Integrated Project Management process area in CMMI, suggest this kind of people issue be handled by proper planning and institutionalizing various group involvements throughout the development of a project (SEI, 2010; PMI, 2008). Hence, software projects ought to have a communicative venue for people to effectively distribute information and collaborate, and should sustain the communication venue throughout the development. This "people" refers to project stakeholders of the project

team, the software organization, the suppliers, and the project customers and users.

2. Managing the Human-centered SD: A Focus on Meetings

Meetings are conceivable in serving as this communicative venue. In the integrated and cooperative software development environment, stakeholders' participation and communication are usually done in the form of meetings (Verner and Evanco, 2005; Gallivan and Keil, 2003; Teasley et al., 2002; Rising and Janoff, 2000; Davision, 1999). According to many reports, meetings enable and facilitate participation in sharing inspiration, leveraging expertise and consolidating information (Hass, 2006; Newell, 2004; Gorse and Emmitt, 2007; Hass, 2006; Wenger et al., 2002). Meetings are also helpful in codifying and preserving substantial group or team actions (Orlikowski and Yates, 1994). The codification (i.e. meeting minutes and subsequent supporting information and documents) becomes the group memories that support the collaborative development of a project.

The study of meetings has been a major topic in project management or information technology related literature. Much literature focuses on the subject of joint application development, group support systems (GSS/GDSS). They are mostly administrative and internal behavioral studies of effectively operating group action inside a meeting, or building computerized tools for running a meeting. In other words, although many studies and tools promote the contribution of meetings to ad-hoc group actions; these isolated meetings are at the micro level in meeting management. A more holistic and collective perspective with regard to the interconnectedness of previous, current, and future meetings may be further needed in order for forming the "group flow (Csíkszentmihályi and Csíkszentmihályi, 1992; Martin, 2010)" that streamlines the collaborative development. In this regard, some researchers take a macro approach with an innovative treatment on project meetings for managing stakeholder involvement. Such an approach suggests treating meetings as mutual interdependent entities, and interconnecting them to form meetings flow, a macro group process that represents the collaborative proceeding of software development.

3. The Meetings-flow Approach

The meetings-flow (abbreviated as MF) study is an emerging research. The concept was seen in conducting students' software capstone projects and engineering projects (Chen and Teng, 2011; Chen and Chong, 2011; Chen, 2009). According to these studies, the serialized manner of meetings forms the temporal meetings-flows, indicating how the collaborative development proceeds; the interdependent information sets among the meeting entities form the contextual flows, indicating the evolving of group knowledge.

Chen and Chong (2011) pointed out that the Meeting-Flow Approach had made a step further and extended the application fields into the software engineering (SE) education field. Besides, it had successfully implemented a senior PIMIS (Project Issues Monitoring Information System) project from the CEUIM (Computing–Engineering Undergraduate Program) at NCU (National Central University) in Taiwan. In this project, an external party ESNE, a CMMI-based company, sponsored the project and played the external stake-holder's role as well. This project implemented the MF in PPQA, VER and PMC process areas of CMMI and examined how MF encouraged team work. It formalized and streamlined stake-holder participation, and show how to monitor students' work as well as sustain their desired collaborative effort throughout the development. MFA have also shown the technical benefits of monitoring product quality and students' work.

Furthermore, Chen et al., (2014) implemented MFA in an undergraduate science, technology, engineering, and mathematics (STEM) project, which emphasized team and project-based learning. The results of this study revealed that the MFA had significantly improved the team communication, coordination and balanced the contribution of the members through giving mutual support and efforts to each other. Its impact is relatively small to the student team cohesion.

In addition, Chen (2011) reported the usage of the MF in managing a multi-party large-scale engineering project. In the report, the project client used the MF to monitor and participate in the development. The MF synergizes the interconnectivity of project

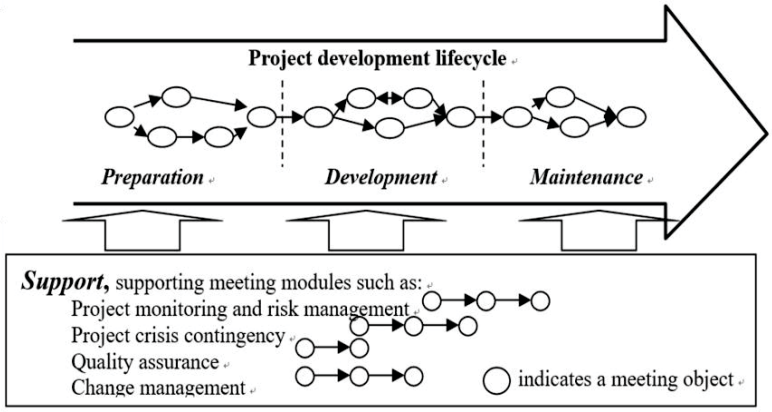


Figure 1: Managing software team processes by managing meetings and their flows

functional meetings and institutionalizes a continuous method and a more natural way of intervention. Such a meeting-oriented process was also recognized in the study as a new type of project's critical path—a path of showing the people and communication bottleneck of the project. This communication critical path differs from the traditional CPM (Critical Path Methodology) that solely focuses on the technical work path. According to Chen's argument (p.12) in the report, while a technical job may be easily done, it often requires substantial efforts to brainstorm the ideas prior to the work, taking considerable time for reviewing the work, and spending much effort on convincing the clients to accept the work.

Due to there are gaps between the students' projects and the business projects in the real-world, such as, students lack experience in, and knowledge of the complete development of long term projects. (Hassan, 2008; Chamillard and Braun, 2002). Students may not be as fully committed to the project or assume as much liability as do those in industry (Sancho-Thomas et al., 2009). Chen, et al., (2013) had out-reached the MFA and implied in a contract-based outsourced engineering projects to see how were these processes shaped in contract-driven projects, and if there was an alternative approach that could improve inter-organizational control of coordination processes (CPs) from the client perspective. The survey results showed that the CP requirements by the client to enable the integration and institutionalization of the venue through the effective management of different organizations related to the project activities. CP is the shape of the client and the contractor, and they can be improved and maintained through MFA.

Based on the aforementioned review on project meetings and current development of the MFA, we explore the applicability of the approach in software project development. Specifically, in this paper we preliminarily introduce how the approach is used in a software project and the benefits of the approach may contribute in streamlining the collaborative development of the project.

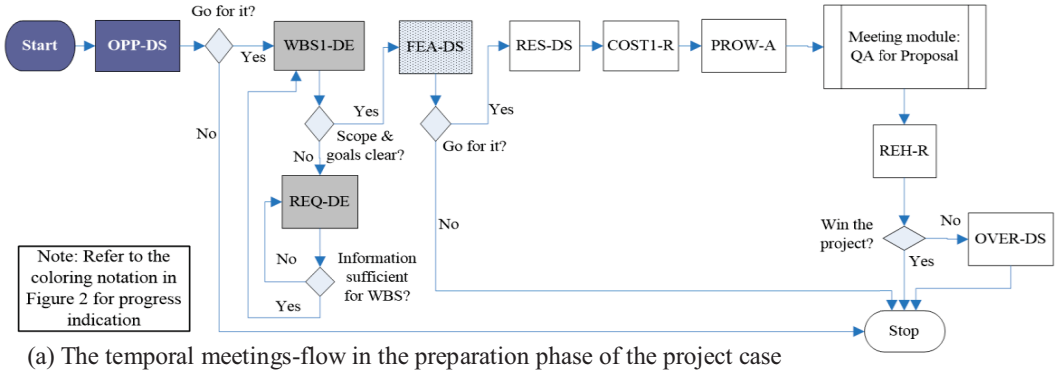
4. A Software Company Case

Introduction

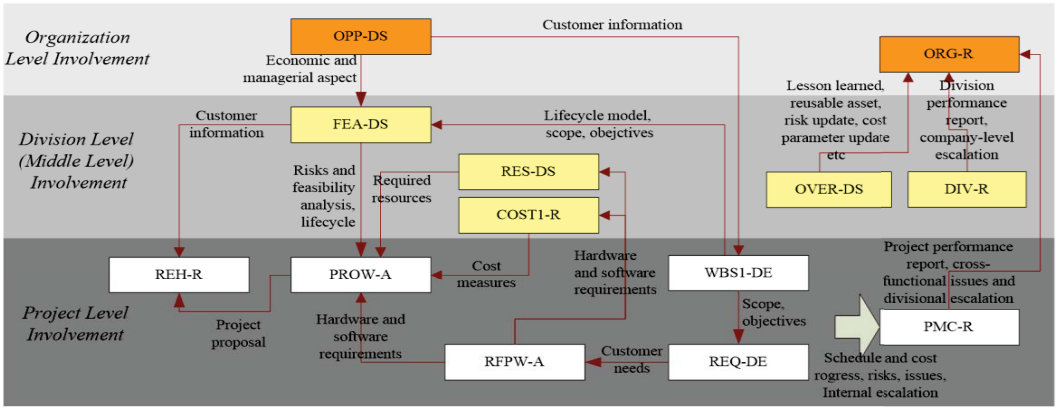
Founded in 1985, Environmental Science and Engineering Inc. (abbreviated as ESNE hereafter) is a system integration company in New Taipei City, Taiwan. ESNE develops ad-hoc meteorological software systems for its clients such as Central Weather Bureau, Taiwan Air Force, and other government agencies. In ESNE, the development of meteorological system projects requires various stakeholders or parties to participate in continuous and intensive validation and verification. Therefore, starting in 2008, the company conducted a research project (Liu, 2009) of using the meetings-flow approach for managing and streamlining project's critical collaboration and group communication path.

ESNE's projects are managed in waterfall-like phases: presale (i.e. preparation), development, and transition and maintenance. Therefore, this paper summarizes the company's MF framework as Figure 1. For simplifying the presentation due to the page limitation, in the following MF report we use only a phase (the "presale" phase) of a project example.

In TKE, meeting entities are identified from any work item on a project's WBS that requires group participation. TKE further



(a) The temporal meetings-flow in the preparation phase of the project case



(b) A DFD like contextual presentation of the meetings-flow model

Figure 2: The meeting-oriented group processes in the case company

defines the participation as group behaviors of brainstorming, review, announcement, reporting, presentation, and negotiation. The meetings are then characterized into various meeting types (classes) with the generic agenda, attending roles, participating roles, degrees of participation, etc. Once identified, the meeting classes are further linked up, according to their corresponding positions on the WBS. The upper part in Figure 2 below demonstrates such a temporal flow.

As the lower part of Figure 2 shows, the project uses DSM (design structure matrix) and a simple DFD-like diagramming tool to frame the information context for the planned meeting entities. The underlying multi-layered communication channel, visualizes the relationship and linkage between meetings (thus participants too) and levels of management. Such meeting-oriented collaborative proceeding, in both the temporal and contextual representations, is regarded as a macro-level group process of the project's development, and the generic content of the meeting types in the flow becomes a reusable group process model for similar projects to follow.

5. Discussion

Streamlining the collaborative software development: Due to a functional organization structure, employees of ESNE work on technical tasks of individual domains. Previously they argued about the lack of a whole picture regarding the shared vision of the project. The meetings-flow was found to fit in this gap. In a follow-up interview, participants replied that the project's meetings-flow enabled a shared track for people to join in together, streamlining the collaborative development by bringing the right information to the right people at the right time and venue.

One practical issue pertinent to such a collective planning manner was raised in the case company. Participants recalled that in the beginning, the planning of meetings-flow was challenging, because they tended to regard meetings as unpredictable and nondeterministic events, e.g. how can the MF determine all the unexpected meetings and handle the change of previously concluded agenda in a meeting? But

later they realized that the MF was not to cover all the meetings that happen or pop up during the development of a project. In TKE, meetings were defined based on the project's WBS, highlighting a collaborative path through the development. They are different from the communication events on demand.

Increasing meeting effectiveness: According to the company, project members felt that, by modeling collaboration into meetings, the MFA helped reduce the number of meetings overall. We further questioned whether project's communication was hindered due to the reduction of meetings. The responses were positive in two. The first benefit referred to the increased control inside a meeting. Because of the understanding of the contextual relationships (Figure (b)), participants in current meeting were able to track the information from previous meetings. Secondly, they became more careful in producing the meeting's outputs, which would be fed to other meetings.

Handling the dynamics of software development: a future study: As far as this paper presents, the MF approach was used to model the critical group path of the reported project. However, software development is inherently unique due to the characteristics e.g. different lifecycle modes, durations, participants, etc. The MF model in Figure 2 may not entirely fit into other projects. Moreover, due to the dynamic nature and the people factor, collaboration in software development may not go as planned. Although the members in the reported project expressed positive responses as mentioned above, they also concerned the effort devoted to follow the planned flow model as the project deployed. In this regard, the planning of project's meetings-flow should also consider the flexibility of dynamically adjust the flow to meet a project's specific dynamic needs. This echoes to the software process tailoring needs recommended in CMMI (SEI, 2010). This becomes one of our studies in the future development of the MF approach.

6. Conclusion

This paper is preliminary and conceptual in nature; it introduces an innovative treatment on project meetings and describes the concept of the MF approach to address the stakeholder involvement issue in software development. While the existing PM methods and software tools mostly present a discrete way to manage project development process, here suggests a new concept, a focus on meetings and meeting flows, to manage and streamline the stakeholder involvement in collaborative software development. In the future, the MF would be continually introduced to the society. This would include the methodological development of applying the MF in software development. Specifically, we would focus on how to align and tailor the flow model to meet individual project's needs.

Acknowledgement

We thank the Taiwan National Science Council for financially supporting this research (NSC-94-2213-E-182-004). We thank Environmental Science & Engineering Corporation for providing the needed operating environments of this research.

References

[1] Artail, H. (2008). A methodology for combining development and research in teaching undergraduate software engineering. International Journal of Engineering Education, 24(3), 567-580.

[2] Bailetti, A.J., Callahan, J.R., and DiPietro, P. (1994) A coordination structure approach to the management of projects, IEEE Transactions on Engineering Management, 41(4), pp.394-403.

[3] Crocitto, M. and Youssef, M. The human side of organizational agility, Industrial Management & Data Systems 103, 6 (2003), 388-397.

[4] Ceschi, M., Sillitti, A., Succi, G., and DePanfilis, S. (2005 May/June) Project management in plan-based and agile companies, IEEE Software, 22(3), pp21-27.

[5] Chamillard, A. T., & Braun, K. A. (2002). The software engineering capstone: structure and tradeoffs. ACM SIGCSE Bulletin, 34(1), 227-231.

[6] Chen, C. Y., & Chong, P. P. (2011). Software engineering education: A study on conducting collaborative senior project development. Journal of systems and Software, 84(3), 479-491.

[7] Chen, C.Y. and Teng C.K. (2011) The Design and Development of a Computerized Tool Support for Conducting Senior Projects in Software Engineering Education, Computers & Education, 56(3), pp.802-817.

[8] Chen, C.Y. (2011). Managing projects from a client perspective: the concept of the meetings-flow approach, International Journal of Project Management, 29(6), pp.671-686.
[9] Csíkszentmihályi, M. and Csíkszentmihályi, I.S. (Eds.) (1992) Optimal experience: Psychological studies of flow in consciousness, The University of Cambridge Press.
[10] Damian, D., Izquierdo, L., Singer, J. and Kwan, I. (2007) Awareness in the wild: why communication breakdowns occur, The 2007 International Conference on Global Software Engineering (ICGSE).
[11] Dennis, A.R. and Garfield, M.J. (2003) The adoption and use of GSS in project teams, toward more participative process and outcomes, MIS Quarterly, 27(2), pp.289-323.
[12] Faraj, S. and Sambamurthy, V. (2006) Leadership of information systems development projects, IEEE Transactions on Engineering Management, 53(2), pp.238-249.
[13] FinstAM, T.D. (2003) Project management, an integrated approach, The British Journal of Administrative Management, 34(January), pp.24-25.
[14] Gallivan, M., and Keil, M. (2003) The user-developer communication process: a critical case study, Information System Journal, 13(1): pp.37-68.
[15] Hass, M.R. (2006) Knowledge gathering, team capabilities, and project performance in challenging work environment, Management Science, 52(8), pp.1170-1184.
[16] Heller, T. (2000) If only we'd known sooner: developing knowledge of organizational changes earlier in the product development process, IEEE Transactions on Engineering Management, 47(3), pp.335-344.
[17] Hong, P., Nahm, A.Y. and Doll, W.J. (2004) The role of project target clarity in an uncertain project environment, International Journal of Operations & Production Management, 24(12), pp.1269-1291.
[18] Hudson, V.F. (2007) The human side, Industrial Engineer, 39(9) pp.40-44.
[19] Humphrey, W.S. (2007) Managing the Software Process, Addison-Wesley.
[20] Liu, H. (2009) A study of the meeting-flow through the software project development in ESNE. Master Thesis, National Central University, Taiwan
[21] Marchewka, J.T. (2010). Information Technology Project Management, John Wiley and Sons Inc., New Jersey, USA.
[22] Martin, P. (2010) Better Business, Pearson Publishing Inc., Upper Saddle River: NJ, USA.
[23] Natale, S. and Ricci, F (2006) Critical thinking in organizations, Team Performance Management, 12(8), pp.272-277.
[24] Probert, G. (1997) Projects, people, and practices, Engineering Management Journal, June 1997 pp141-146.
[25] Project Management Institute (PMI) (2008) Project management body of knowledge (PMBOK)
[26] Rising, L. and Janoff, N.S. (2000) The scrum software development process for small teams, IEEE Software.
[27] Roberts, T.L., Cheney, P.H. and Sweeney, P.D. (2002) Project Characteristics and group communication: an investigation, IEEE Transactions on Professional Communication, 45(2), pp.84-98.
[28] Sancho-Thomas, P., Fuentes-Fernández, R., & Fernández-Manjón, B. (2009). Learning teamwork skills in university programming courses. Computers & Education, 53(2), 517-531.
[29] Schwartzman, H. B. (1989). The meeting. In The Meeting (pp. 309-314). Springer US.
[30] Software Engineering Institute (SEI) (2010) Capability Maturity Model® Integration (CMMISM) Version 1.3 for DEV, CMU Press, Pittsburgh: PA.
[31] Steele-Johnson, D. (2000) Goal orientation and task demand effects on motivation, affect, and performance, The Journal of Applied Psychology, 85(5), pp724-7238
[32] The Standish Group (2007). CHAOS Summary for 2006, West Yarmouth, MA, USA, http://www.standishgroup.com/press/article.php?id=2
[33] Teasley, S.D.; Covi, L.A.; Krishnan, M.S.; Olson, J.S (2002) Rapid software development through team collocation, IEEE Transactions on Software Engineering, 28(7), pp671 – 683.
[34] Verner, J.M., and Evanco, W.M (2005 Jan/Feb) In-house software development: what project management practices lead to success? IEEE Software, 22(1), pp86-93.

Development of "Multiple Sightseeing Spots Scheduling System" and Comparison with The Existing Sightseeing Methods

Kazuya Murata[1], **Takayuki Fujimoto**[1]
[1]Graduate School of Engineering, Toyo University, Saitama, Japan

Abstract - *Recently, foreign tourists and travelers coming to Japan is increasing explosively. In 2004, the number of the foreign visitors was approximately 6.1 million people, and it became approximately 20 million people in 2015. One of the backgrounds includes various Japanese sightseeing policies such as "Visit Japan Project" and "Cool Japan". As a result, Japan's travel and tourism competitiveness of 2015 was the ninth place, a high rank in the world. However, a problem is still left in the current Japanese sightseeing situation. For example, there are not so many stores with foreign language correspondence. And also there are not so many places where they can pay with a credit card. Furthermore, the traffic such as the subway or the buses is complicated. In current Japan, those kinds of problems are left. But, it is difficult for foreign tourists and travelers to accept them without stress. We saw if there was any way to solve the problems and uneasiness. In this research, we have developed totally a new form of sightseeing application "Multiple sightseeing spots scheduling system" as a solution for these problems and uneasiness. In this paper, we developed the application prototype, and performed the comparison experiment with the existing sightseeing methods and verified superiority of this application.*

Keywords: Sightseeing, Tourist, Application, Sightseeing Information

1 Introduction

T Recently, foreign tourists and travelers coming to Japan is increasing explosively. In 2004, the number of the foreign visitors was approximately 6.1 million people, and it became approximately 20 million people in 2015, 10 years later [1]. The background includes recent Japanese sightseeing policies. For example, they are "Visit Japan Project", "Cool Japan Policy" and "Relaxation of the visa acquisition for Middle Eastern countries".

Japan Tourism Agency plans "Visit Japan Project" and Japan National Tourism Organization performs promotes it. "Visit Japan Project" is one of "Strategy to increase the number of foreign tourists visiting Japan". In "Visit Japan Project", Japan establishes 20 important point markets abroad to attract foreign tourists. And promotion of Japan in those countries and areas are performed. This promotion includes two methods of "Public appeal in overseas markets" and "Promotional programs for travel agents". In "Public appeal in overseas markets", activities such as "Transmitting charms of the Japan sightseeing by newspapers, magazines and websites, etc." and "Inviting the local media to Japan and encouraging them to deliver charms of Japanese sightseeing" are carried out. In "Promotional programs for travel agents", an activity mixing "Promotion of Japan sightseeing by Japan National Tourism Organization" with "The advertisement of Japan sightseeing product of the travel agency" is held. Like these, Japan sightseeing is promoted by various methods.

"Cool Japan Policy" is a policy of Ministry of Economy, Trade and Industry. Current Japan has the famous industries: the car industry; the electric appliances industry and etc. In addition, Japan has special culture including contents such as "Anime", "Manga", fashions and Japanese food. The foreigner appreciates them as so-called "Cool Japan". These kind of Japanese unique culture can be expanded to the business deployment properly and it enables people to be interested in Japan more. At the same time, Japan can get foreign demand. Attracting foreign tourists is the activity which can be connected with economic growth of Japan. Examples are "Effective transmitting of Japanese charms", "Platform construction to make money locally", and "Bringing bark consumption to Japan".

In "Effective transmitting of Japanese charm" is an activity to raise interest in Japan. For Example, for the overseas development promotion of "Anime" and "Manga", there is the localization such as subtitles and dubbing. In addition, there is the promotion activity to an international trade fair and running the advertisement. In "Platform construction to make money locally", There is an activity such as the securing of the channel for exclusive use of Japanese contents and the allied product sale in commercial facilities. Through this activity, the matching support is practiced for local company and Japanese company with materials based on unique Japanese technique and culture. In 2014, it carries out eight times of trade fair exhibition and business matching were held in various countries. In "Bringing bark consumption to Japan", Japanese attractions are dispatched as "Cool Japan" to acquire share in overseas markets. It also promotes "Visit Japan Project". For example, an activity to attract foreign people to Japan is practiced by

spreading information on Japanese traditional craft arts abroad and also casting the tours for foreigners to experience Japanese traditional craft arts at the same time. In this way, the policies targets to deliver Japanese various contents to the foreign countries and encourage foreign people to come to Japan.

In "Relaxation of the visa acquisition for Middle Eastern countries", relaxation of the acquisition of the visa is carried out for Middle Eastern countries such as Indonesia, Philippines, Vietnam and China.

By these sightseeing policies, the current Japanese sightseeing competitiveness becomes the high rank of the ninth in the world [2]. In 2020, "Tokyo Olympics" will be held. From these, the sudden increase in number of foreign tourists and travelers are expected in the future.

2 Purpose

Current Japanese travel and tourism competitiveness is a high rank of the ninth in the world. However, a problem is still left in the Japanese sightseeing situation. It is a problem with the information environment on sightseeing spots and the stores in Japan. For example, there are not so many places with the foreign language correspondence. Of course, there are a lot of stores with the foreign language correspondence, but the stores with no foreign language correspondence still exist. In addition, there are few places where the credit card payment is possible, and the credit card payment is a basic means of payment in the foreign counties. Furthermore, the traffic such as the subway or the buses in Tokyo is complicated and hard to understand. Regarding sightseeing of Japan, these problems are left. However, it is difficult for the foreign tourists and travelers who don't know much about Japan to accept those problems without stress. Of course, it is difficult for them to find the places which are other than famous sightseeing spots and stores on the guidebook.

Thesedays, most of foreign tourists and travelers are interested in Japanese local spots more than Japanese famous sightseeing spots. However, it is not easy to find those places. The tourists and travelers can use websites for Japanese sightseeing information, but they are just digitized version of the guidebooks basically. To increase the number of the tourists and travelers to Japan more, visits to the places other than sightseeing spots on the guidebooks are indispensable.

Therefore we thought if there was any way to solve this kind of problem by a casual tool, which is other than Japanese policies, but enables foreign tourists and travelers to come to Japan more. And we devised "Multiple sightseeing spots scheduling system based on inversed operation method" that enabled a guidance of new sightseeing to suggest in this research.

3 Purpose

In this research, we have proposed and created the prototype of "Multiple sightseeing spots scheduling system based on inversed operation method" enabling guidance for new type of sightseeing. This application consists of the following five components.

- Settings for the "current place" and the "place to return"
- Settings for time to return
- Selection from "sightseeing categories"
- A list of sightseeing information
- Use or disuse of the credit card payment

In this chapter these five components are described in detail.

3.1 Setting for the "Current Place" and the "Place to Return"

This application requires the user to set the current place and the place to return, unlike common sightseeing applications. From the current place and the place to return, the application determines some range of the tourist route or sightseeing plan to suggest.

3.2 Setting for Time to Return

The common sightseeing applications display time required to the destination on the search results. But, for the application of this research, the user needs to set "time to return". Using the settings, the system suggests possible tourist route and sightseeing plan by calculating time allowance available for sightseeing from current time to "time to return".

3.3 Selection from "Sightseeing Categories"

For the common sightseeing applications, the user inputs a destination and searches routes to the destination. But for this application, the user decides a sightseeing place to go by selecting "sightseeing categories" without setting the destination. In this application, we set "sightseeing categories" such as table 1

Table 1. List of "Sightseeing Categories"

Tourist spot	Cafe	Temples Shrines
Museum	Japanese sweet	History
Aquarium	Japanese food	Souvenir
Archives Center	Western food	Garden
Art Museum	Chinese food	Park
Memorial Hall	Traditional Craft	

In this application, using this "sightseeing categories", the system shows appropriate sightseeing spots for "sightseeing categories" from the area determined by calculation based on the current place, the place to return, and time to return.

3.4 List of Sightseeing Information

This application displays information on the suggested sightseeing spots as well as the suggestions of the tourist route and the sightseeing plan. For this application, we made a list of sightseeing information to be viewed by the user. In that list of sightseeing information, we list information on table 2.

Table 2. List of Sightseeing Information

Sightseeing Categories	Store Name	Address	Opening Hours	TEL
Regular Closing Day	Need Time	Need Money	Use or Disuse Credit Card payment	

The users choose sightseeing spots using "sightseeing categories" from this sightseeing information. By the search results of this application, the use of the system can read sightseeing information from the search results.

3.5 "Use or Disuse of the Credit Card Payment"

One of the target of development of the application in this research aims to enable foreign tourists and travelers to do sightseeing without stress. To solve the problem, the users can choose use/disuse of "Credit card payment" when searching a tourist route. The system excludes the sightseeing spots and stores where the credit card payment is unavailable form the results and it can suggest a more comfortable tourist route and sightseeing plan.

With these five components, first, the system "calculates a range from the current place and the place to return". Second, it "calculates time allowance for sightseeing by back calculation from time to return". At last, it "searches appropriate sightseeing spots for the specified sightseeing categories". That is how the system suggests possible tourist route and sightseeing plan in specified time.

4 Example of Operation Method "Multiple Sightseeing Spots Scheduling System"

In this chapter, an operation method of "Multiple sightseeing spots scheduling system" is described.

4.1 Setting for the Current Place, the Place to Return, and Time to Return

In this application, firstly, the users input a current place and a place to return. User input a current place as a starting point and a place to return into "current place" and "place to return" columns of the input form in figure 1. In the former prototype system, "current place" was specified only by the station. However, the current system enables guidance using GPS by the "Current place" input in the "Present place" column. Also, the users are required to input "Time to return" into the "Time to return" column. For example, in the case of a plan to return to Tokyo Station at 18:00, the user inputs "1800" in the column. From this, the application calculates time form "present time" to "time to return" by back calculation and suggests a tourist route and sightseeing plan in specified time.

An example of these operations is indicated in figure 1.

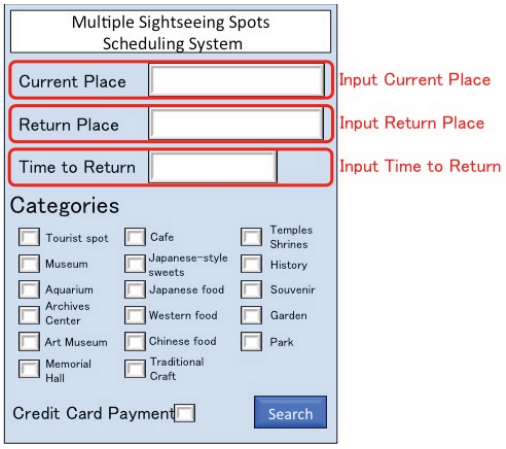


Fig 1. Settings for Current Place, Return Place, and Time to Return

4.2 Selection from "Sightseeing Categories"

In this application, the user sets "sightseeing categories" that is connected with "list of sightseeing information". The system suggests appropriate sightseeing spots for "sightseeing categories" by this selection. Also the user can set plural "sightseeing categories" and can suggest a tourist route and a sightseeing plan to the multiple sightseeing spots.

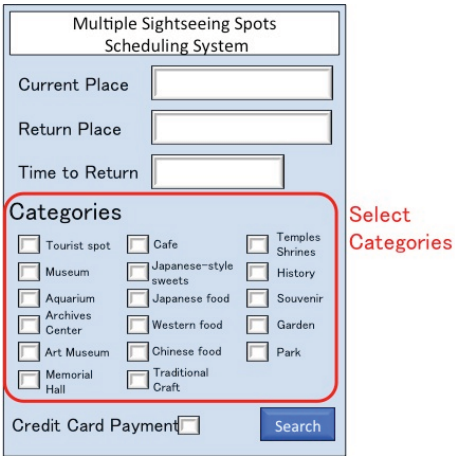


Fig 2. Selection from "Sightseeing Categories"

4.3 Use or Disuse of the Credit Card Payment

With this application, we aim to suggest more comfortable tourist route and sightseeing plan s for a foreign tourists and travelers. Therefore in this application, at the time of tourist route search, the user choose s "use or disuse of the credit card payment".

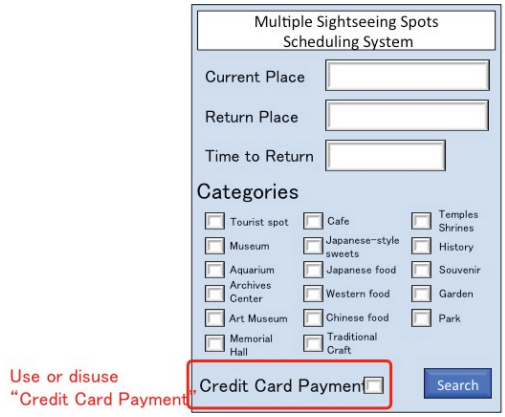


Fig 3. Setting for "Use or Disuse of the Credit Card Payment"

4.4 Search Results

When the user sets five components: "current place", "place to return", "time to return", "sightseeing categories", and "use or disuse of the credit card payment", and then carries out a search, the screen image in figure 4 will be displayed. In the search results, the application displays the tourist route and a sightseeing plan with the map in the upper part. At the same time, it displays a tourist route and a sightseeing plan with the text. In a sightseeing plan with the text, names of sightseeing spots and stores are displayed. The user can view sightseeing information by tapping those names of sightseeing spots and stores. This sightseeing information will be displayed using the information list that we made in "list of the sightseeing information".

Fig 4. Search Results Example

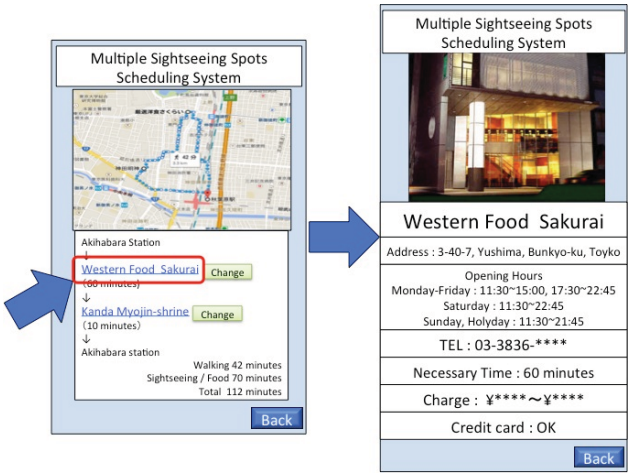


Fig 5. Example of the Store Information

In addition, this application has "Change button" beside names of a sightseeing spots or stores. The user can change the route to the different sightseeing spot or store in the same "sightseeing categories" by tapping "Change button". The operation example is indicated in figure 6. By this system, the user thinks about one's favorite sightseeing and can make the user's original sightseeing route or the sightseeing plan. This application enables guidance of the sightseeing that is very high diversity.

Fig 6. Example of the Change of Sightseeing Spot

5 Comparison Experiment between Existing Sightseeing And Application of This Research

In this research, we are developing a new application to guide sightseeing by a method totally different from the existing guidebooks and applications. As a beginning of the comparison experiment, we actually saw the sights by three methods: "Sightseeing using a guidebook"; "Sightseeing using a smart phone": "Sightseeing using the application of this research", and carried out a comparison experiment.

In this chapter, we will review an experiment method and experiment results.

5.1 Experiment Method

In this research, we carried out the comparison experiment for the three sightseeing methods: "Sightseeing using a guidebook"; "Sightseeing using a smart phone"; "Sightseeing using the application of this research". We set almost same conditions and carried out sightseeing by three methods. And we compared each sightseeing method from experiment results and considered superiority and a refinement of this application.

5.2 Setting of the Sightseeing Condition

Sightseeing conditions are indicated in table 3.

Table 3. Sightseeing conditions in this experiment

Condition 1	Sightseeing area form Ikebukuro to Tokyo Station
Condition 2	Sightseeing time from 13:00 to 17:00
Condition 3	Add lunch (Japanese food) in the sightseeing route

We set those conditions and saw the sights by each method, and considered the experiment results.

5.3 Experiment Results

Experiment results on "Sightseeing using a guidebook", "Sightseeing using a smart phone", and "Sightseeing using the application of this research", are indicated in table 4, table 5, and table 6.

Table 4. Sightseeing using guidebook

Order	Tourist route	Area
1	Sunshine city aquarium (aquarium)	Ikebukuro
2	Unagi Kappo IZUEI (Japanese food)	Ueno
3	Ameya-yokocho market (sightseeing spot)	Ueno
4	Tokyo Sweet Land (sightseeing spot)	Tokyo
5	Tokyo Station (Souvenir)	Tokyo

Table 5. Sightseeing using smart phone

Order	Sightseeing route	Area
1	Sushi Mamire（Japanese food）	Ikebukuro
2	Namco Namja town （Sightseeing spot）	Ikebukuro
3	Gokoku-jinja Shrine (Shrine)	Ikebukuro
4	Senso-ji (Tenmple)	Ueno
5	Nakamise Shopping Street (Sightseeing spot)	Ueno
6	Tokyo Station (Souvenir)	Tokyo

Table 6. Sightseeing using application of this research

Order	Sightseeing route	Area
1	Manmaru (Japanese food)	Ikebukuro
2	Ancient Orient Museum (Museum)	Ikebukuro
3	Nakamise Shopping Street (Sightseeing spot)	Ueno
4	Radio Center (Souvenir)	Akihabara
5	Kanda Shrine (Shrine)	Akihabara
6	Red-brick Tokyo Station (Sightseeing spot)	Tokyo

Through this experiment, the remarkable difference that we felt was "Difference of the required time before deciding a sightseeing spot". For "Sightseeing using a guidebook" and "Sightseeing using a smart phone", at first the user has to decide where oneself goes to. In this experiment, we spent considerable time to decide the first sightseeing spot. On the other hand, regarding "Sightseeing using the application of this research", at first we targeted the sightseeing spots that one wants to go in "sightseeing categories" and then a rough plan of the tourist route was displayed. In regards to the application of this research, the user could change the sightseeing spot that oneself wants to go to from the first tourist route for the decision. Therefore it did not take time before deciding a tourist route.

There was also a problem about "required time for sightseeing". Regarding "Sightseeing using the application of

this research", transferring time and required time for sightseeing are displayed for search results. With the system, we saw the sights checking the time. As a result, we finished planned sightseeing smoothly and arrived back at Tokyo Station on time. But, we need to decide the tourist route by ourselves, in regards to "Sightseeing using a guidebook". Therefore we had to think about required time for the sightseeing route and had to practice the plan. Also, we had to transfer while always watching the guidebook. it was difficult to do sightseeing while being conscious of time. As a result, we were just able to see the five spots without enough time though planned to go to six spots. On the other hand, in regards to "Sightseeing using smart phone", transferring time to a sightseeing spot was displayed, but required time for sightseeing was not displayed. Time lag occurred and it was not sightseeing on schedule.

From these results, the application of this research has higher performance for time comparing with a common sightseeing methods, and it is thought that there is superiority.

Regarding "Sightseeing using a guidebook", we visited a sightseeing spots and stores listed in the guidebook basically. Therefore the sightseeing information is limited. However, this application can display local sightseeing spots and stores, which are not on the guidebook.

And also, in regards to "Sightseeing using a smart phone", famous stores are displayed as search results mainly. With the smart phone, we could research the local sightseeing spots, but it took much time.

5.4 Problem point of this application

From these experiment results, "Multiple sightseeing spots scheduling system" being developed in this research has high cost-performance for the time in particular. On the other hand, we discovered the problems with this application.

"Change button" for the sightseeing spots or stores that we implemented newly has the problem. In "sightseeing using the application of this research" by this experiment, we intended to change the sightseeing spot from a suggested sightseeing spot displayed initially. But, it was hard to understand where a displayed sightseeing spot was at all when we tapped the "change button" and the alternative was displayed. As a result, a sightseeing route was displayed in a strange way when we changed it into one in far-off area. Also, all sightseeing spots in the specified "sightseeing categories" was displayed. As a result, it was very difficult to change a sightseeing spot.

It is necessary for us to revise a system to change the sightseeing spots. As the solution, we think about a system possessing "priority ranking" in "sightseeing categories". In the current system, "sightseeing categories" require the user to choose all the categories that one wants to go for sightseeing. We also think about letting this system display only an appropriate sightseeing spots by possessing priority ranking such as " The spots that user wants to go most" or "The category that user want to go second most".

6 Conclusion and Future Research

In this research, we are developing "Multiple sightseeing spots scheduling system" enabling guidance of new sightseeing. In this paper, we have carried out the comparison experiment for three sightseeing methods: "Sightseeing using a guidebook"; "Sightseeing using a smart phone": "Sightseeing using the application of this research". From the experiment results, we considered superiority of this application, discovered problems with this application and aims at future improvement. We carried out the experiment only by the authors without enough time. The next experiment would be the examinee test in which more people do sightseeing using this application actually. We will verify the difference from other styles of sightseeing in more detailed way and discover problems for further improvement of this application.

7 References

[1] Japan National Tourism Organization, "Trend of the number of the visit to Japan foreign visitors", http://www.ww.jnto.go.jp/jpn/reference/tourism_data/visitor_trends/

[2] The World Economic Forum "The Travel & Tourism Competitiveness Report 2015", http://www3.weforum.org/docs/TT15/WEF_Global_Travel&Tourism_R eport_2015.pdf

[3] Japan Tourism Agency, "Inbound Travel Promotion Project (Visit Japan Project)", http://www.mlit.go.jp/kankocho/en/shisaku/kokusai/vjc.html

[4] Ministry of Economy, Trade and Industry, "Cool Japan / Creative Industries Policy", http://www.meti.go.jp/english/policy/mono_info_service/creative_indust ries/creative_industries.html

[5] K. OGAWA, Y. SUGIMOTO, K. NAITO, T. HISHIDA, T, MIZUNO, "Basic design of a sightseeing recommendation system using Characteristic Words", IPSJ SIG technical reports 2014-MBL-71(14), 1- 6, 2014-05-08

[6] M. URATA, S. NAGAO, F. KATO, M. ENDO, T. YASUDA, "Photo Rally System to Support Tourists in Tourism Areas [In Japanese]" Journal of the Japan Information-culture Society 21(2), 11-18, 2014-12-25

[7] Jalan, "Introduction of the tour guide application -Jalan net-", http://www.jalan.net/jalan/doc/howto/iphone_kankou.html

[8] Shigeo Kuroda, "TABIMARU Tokyo", Shobunsya Publications, Inc. , August 2015

A Conceptual Data Model for Health Information Systems

André Magno Costa de Araújo[1], **Valéria Cesário Times**[1], **Sérgio Castelo Branco Soares**[1]
[1] Center for Informatics, Federal University of Pernambuco, Recife, Pernambuco, Brazil

Abstract - *The development of Health Information Systems based on dual models allows modifications to be conducted in the layer of archetypes, reducing dependencies on software developers. However, we identified a lack of conceptual models to represent two-level database entities. This paper proposes a novel conceptual data model, called ArcheER, which is a dual modeling approach and aims to reduce redundant entities and guarantee the creation of unique electronic health records. ArcheER is an extension of the Entity-Relationship model and is based on archetypes. A CASE modeling tool based on ArcheER is outlined. Finally, to illustrate the key features of the proposed model, an ArcheER conceptual schema built for a legacy system is discussed, and results collected from a test with 18 human subjects are reported. Results indicated a reduction of 83,35% in the representation of redundant entities and a gain of 78,9% concerning the modeling of entities characterizing knowledge.*

Keywords: Novel Software Tools, Conceptual Data modeling, Health Information Systems, Archetypes.

1 Introduction

Conceptual modeling is an important activity for designing a database. The conceptual scheme is a concise description of data requirements specified by the application designer, including detailed descriptions about types of entities, relationships and constraints [1]. Thus, the artifacts generated from the conceptual data modeling are important elements in building database systems. Currently, most Health Information Systems (HIS) are built using traditional database modeling technologies [2], in which both information and knowledge concepts are represented in single level computer systems using conventional data models. However, HIS must handle a large number of concepts that often change or are specialized after a short period of time and, consequently, HISs based on such models are expensive to maintain.

Several research projects and many applications have been developed from the specifications of the openEHR system architecture and the concept of archetypes [3-8]. The Open Electronic Health Record (openEHR) software architecture for HIS is aimed at developing an open, interoperable and computational platform for the Health domain [9]. This architecture separates generic information that represents the structures of the Electronic Health Records (EHR) and demographic characteristics of the patients of a reference model, from the constraints and standards associated with the clinical data of a given specific domain, which composes the knowledge model. An archetype consists of a computational expression that is based on the reference model and is represented by domain constraints and terminologies [3] (e.g. data attributes of a blood test), while templates are structures used to group archetypes for allowing their use in a particular context of application, and are often associated with a graphical user interface. On the other hand, some authors have already proposed extensions of traditional conceptual modeling techniques to represent HIS applications. However, these extensions do not model EHR, do not provide dual modeling constructors and are not based on archetypes. In fact, little attention has been devoted to the investigation of the following issue: which conceptual constructors are needed to model the two-level database entities of HIS applications?

This paper proposes a novel conceptual two-level data model, named ArcheER, for helping database designers with the modeling of HIS applications. ArcheER is an extension of the Entity-Relationship (ER) model [1] and is based on the openEHR definitions [3]. It also comprises a set of modeling constructors with graphical representations for building health information conceptual schemas and a set of knowledge-level constructors that are based on archetypes. The ER model was chosen because it is simple and widely used in both academia an organizations for the development of DB applications, and because it is capable of providing an abstraction of implementation details as well as being easily mapped to DBMS logical data models. Another contribution of this paper is related to the development of a modeling tool for ArcheER. The main goal of such tool is to provide application designers with computer support to assist in the database modeling activities of healthcare applications.

The dual modeling approach has been used by several researchers and is not unique to archetypes [10]. However, in this paper the focus lies on the use of dual modeling based on the concept of archetypes, since [8],[11],[12] reported the use of this approach as essential to achieve interoperability and standardization of EHR.

2 Related work and motivation

Späth and Grimson [8] used the openEHR specification to map the structure of an EHR into a proprietary database system. They examined the reuse of archetypes available in the repository of openEHR by specializing some of them and then proposing a new set of archetypes to support biomedical knowledge discovery. To achieve this, they studied the database schema to reorganize it according to the concept of archetypes, by mapping each field of the database to an archetyped element. Some difficulties were reported while doing so, including a lack of consolidated modeling tools and lack of mechanisms to determine overlapping archetypes as

well as solve the semantic conflicts that may appear when archetypes are mapped to the chosen DBMS.

Bernstein et. al [6] conducted a study in Denmark about the patterns of the development of healthcare computer systems. This research indicated that the Danish healthcare systems were based on several information models and heterogeneous technology platforms, developed by different software vendors. Besides, it showed the need for replacing traditional standards of software development in the Health domain, and reported the importance of the openEHR architecture as a new pattern for the development of computer systems for healthcare.

Despite the development of HIS based on the openEHR specifications being a multidisciplinary research area, (there are already varied studies published by the scientific community [13-15]) there is a consensus that the openEHR architecture definitions have to evolve and address some open problems [8]. This paper points out that the difficulty in applying the openEHR concepts to a given problem domain for enabling the two-level data modeling is due to the lack of a methodology to express which are the data requirements requested by users and how these might be modeled.

This paper goes one step beyond previous works by specifying an ER-based conceptual data model enabling the definition of which archetypes, patient demographic properties, hospital administrative information and clinical data are important and should be taken into account during the conceptual modeling of a healthcare database application. The main concepts of the ArcheER modeling proposal are detailed in the following section.

3 The ArcheER conceptual model

ArcheER is a conceptual data model that aims to allow the specification of a health application domain using the concepts of dual modeling. The proposed set of ArcheER modeling constructors extends the basic ER modeling elements with a set of archetyped components listed in Figure 1. ArcheER represents entity types alongside their relationships and properties, and a conceptual schema is composed of hospital administrative data and archetyped information. An archetyped entity denotes a set of entities that must have a set of generic data structures. Each of those structures is defined as an attribute of those entities, and organizes data through data structuring elements that are neither dependent of the DBMS storage format nor of the application development technology.

In order to model relationships between archetyped entities, and a relationship between a conventional entity and an archetyped entity, ArcheER proposes a new relationship type, called *Party Relationship*. It is worth noting that an administrative entity may not be modeled as an archetyped entity, i.e. it may not be represented as an entity type with a set of generic data structures, therefore not being able to have party relationship associations with other entity types. However, this must not be mandatory and an administrative entity may benefit from the use of generic data structures as well.

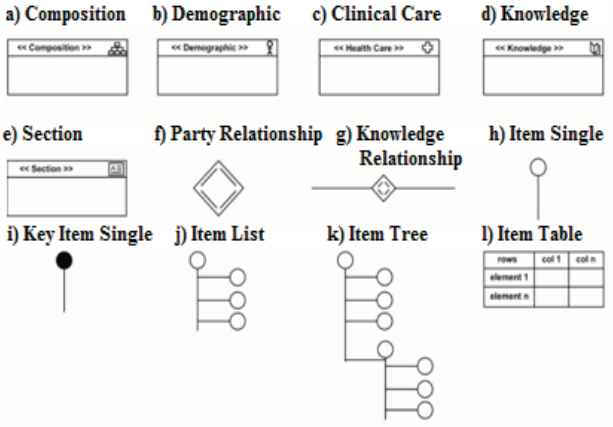


Figure 1. The Main Modeling Components of ArcheER

One of the advantages of ArcheER is the elimination of data redundancy by defining a uniqueness constraint based on the concept of roles played by the actors being modeled. According to this constraint, every instance of a relationship involving the demographic information of an actor and an entity of the type *clinical care* or *administrative* must be modeled as a relationship between a role played by the actor and the entity type *clinical care* or *administrative*. Observe that the use of openEHR definitions requires an understanding of which actors should be considered while modeling an application domain, how they relate to each other, how they play their roles and which capabilities they have. This understanding is important and must not be neglected. However, this cannot be enforced automatically by the DBMS nor can it be seen as a data model constraint to guarantee a unique EHR.

The ArcheER constructors inherited from ER are mostly used for modeling the operational aspects of a hospital organization (i.e. entity type *Administrative*), while the archetyped entity types are mainly concerned with the representation of (i) metadata and the context of the application being modeled (i.e. entity type *Structuring*); (ii) patient's demographic information (i.e. entity type *Demographic*.); (iii) clinical data (i.e. entity type *Clinical Care*) and (iv) constraints, terminologies of health area, internal coding of vocabulary and textual information given by a domain specialist (i.e. entity type *Knowledge*). While the first three entity types represent the information level of the dual modeling approach, the last type of entity and its specializations compose the second level and are useful for generating knowledge at runtime. The definition of each type of constructor is given below.

3.1 Structuring constructors

The ArcheER data model provides the following modeling constructors for structuring health care information: *Composition*, whose attributes represent the metadata of an ArcheER conceptual schema; and *Section,* which organizes the remaining modeling constructors of ArcheER into themes

or subjects that represent the context of the application being modeled.

3.2 Demographic constructors

The modeling of demographic information requires the identification of actors who compose the hospital application domain, and the definition of their roles and capabilities in the health area. For modeling actors, ArcheER specifies the following set of constructors of demographic entities that represent the specialization of an actor in a Health domain. The definition of each type of demographic entity is given below:

- *Agent*: Expresses a software agent or any device that communicates with the healthcare application.
- *Person*: Corresponds to an entity type that represents a generic description of people who are part of the context of the application being modeled.
- *Group*: Models parts of the real world that interact with each other and are grouped to represent the purpose of being together.
- *Organization*: Denotes an abstraction of all companies involved in a health application domain.
- *Role*: Represents a generic description of a role played by a given actor.
- *Capability*: Models the qualification of an actor to play a certain role in a healthcare domain.
- *Party Identity*: Indicates how an actor is identified in a healthcare application, and allows an actor to be identified in several ways.
- *Contact*: Expresses the possible ways of contacting an actor.
- *Address*: Indicates how the contact information of actors is formatted.

3.3 Health care constructors

The ArcheER modeling constructors that represent health care information are in charge of defining all the semantics of EHR - hence, the information they model represent the main target to be archetyped. For the modeling of clinical care information, ArcheER proposes the following entity types:

- *Admin Entry*: Expresses all the administrative information of patients in the modeling of EHR. Note that this entity type concerns the modeling requirements of the patient´s administrative information that compose the EHR of the patient and does not refer to administrative aspects of a service provider organization in health. In this work, for the modeling of these administrative issues of a health service organization, we assume that the use of traditional ER constructors will suffice, thus, in fact, only clinical care and demographic information are modeled using archetypes.
- *Observation*: Represents any event or clinical status associated with the patient.
- *Instruction*: Expresses all future actions to be administered to the patient.
- *Activity*: Specifies the activities of an instruction.
- *Action*: Specifies the actions of an instruction.
- *Evaluation*: Represents general information about the clinical care of patients, based on diagnosis, assumptions, risk assessments and observations.

3.4 Knowledge constructors

The entity type *Knowledge* expresses the terminology and constraints related to attributes, also called generic data structures. This entity allows the second level of dual modeling to be displayed in an ArcheER schema. Furthermore, ArcheER adds a new constructor of relationships, called *Knowledge Relationship,* to express associations between generic data structures and instances of the entity type *Knowledge.*

ArcheER extends the definition of ER relationship types to enable the creation of direct relationships between the generic structure attributes of archetyped entities and the entity type *Knowledge*. The following relationship cardinalities are considered by our ArcheER proposal: 1:1, 1: N and M: N.

The entity type *Knowledge* is specialized in the following entity types: *Free Text, Internal Code and Terminology*, which are directly related to the generic data structures through the *Knowledge*-type relationship. These specializations model knowledge of a given Health domain – in other words, they represent the second level of ArcheER dual modeling. The entity type *Free Text* represents free text information given by the domain specialist, while the entity type *Internal Code* denotes codes of a health vocabulary (e.g. procedures, billing tables, international classification of diseases) used for the exchange of information between EHR applications. Lastly, the entity type *Terminology* represents terms and concepts designed to standardize, promote and disseminate health knowledge.

3.5 Data entry constructors

The ArcheER modeling constructors used to define attributes are called data entry constructors, since such attributes comprehend entries with any kind of data that are represented by generic data structures. Hence, generic data structures are defined as attributes of archetyped entities of ArcheER. For each element of these data structures, a data type must be specified. An entry may have a single clinical statement (e.g. a short description about the history of the current illness), or otherwise contain a large amount of data (e.g. the list of values of a laboratory test, tabular data reporting a hospital infection, a hierarchical structure containing all procedures, materials and medications of a patient's hospital bill, an entire microbiology result or a psychiatric examination note). An entry defines the semantics of multiple formats of data which are properties of the archetyped entities of ArcheER.

For modeling generic data structures, the ArcheER model provides the following types of attributes: *ITEM_SINGLE*: represents a data structure with a single element; *ITEM_LIST*: represents a list of data items or values, where each element of this list may assume a value or not, may be referenced by a name and may have an index to indicate its position within the list; *ITEM_TREE*: models a data structure that is logically represented as a tree; and *ITEM_TABLE*: defines a data structure with lines and columns, where the line represents the specification of an element, and the column the information value.

3.6 ArcheER constraints

A set of constraints aiming at ensuring the uniqueness of the EHR is specified in Object Constraint Language (OCL) [16] notation. Thus, the relationship between demographic and clinical care entities and the relationship between demographic and administrative entities of the patient are restricted by two constraints, respectively: *Context Clinical Care inv: Health->forAll (oclType=Role) and Context AdminEntry inv: Health->forAll (oclType=Role or OclType=Administrative)*. Consequently, each instance of entities *CareEntry* and *AdminEntry* is related with demographic information of patients only by means of the roles played by the actors. The benefit of defining constraints over these relationships using the concept of roles is that, while actors of a Health domain are modeled as generic entities, their specific characteristics are represented as roles. This ensures the conceptual modeling of the uniqueness of demographic information, since new instances of a given actor are created only through the roles played by him.

In order to model actors, roles and capabilities, we propose four constraints, which are aimed at enforcing the uniqueness of EHR and explained as follows. Constraint *Context Actor inv: Actor.allInstances->forAll (ar | self.Actor < > ar.Actor implies self < > ar)* specifies the actors' uniqueness constraint and enforces that each instance of an actor entity of ArcheER is unique. The constraint *Context Actor inv: self.Actor_Role->notEmpty() implies self.Actor_Role->forAll (r1 | self.Role < > r1.Role implies self < > r1)* indicates that each actor is not allowed to have two instances with the same role, while constraint *Context Role inv: self.Actor_Role-> includes (self.Actor)* guarantees that in order to create a new instance of a given role, a corresponding instance of an actor must exist. In addition, the constraint *Context Capability inv: self.Role_Cap-> includes (self.Role)* defines that, for each instance of the entity *Capability*, a corresponding instance of the entity *Role* must exist. Entity *Address* models the details of each instance of the entity *Contact*; thus, an instance of entity *Address* can exist only if there is a corresponding instance of entity *Contact*. This is enforced by the constraint *Context Address inv: self.address-> includes (self.Contact)*.

The entity *Administrative* may be related to demographic information (i.e. to instances of the entity *Demographic*) and to clinical concepts as well (i.e. to instances of the entity *ClinicalCare*). For relationships with demographic information, a constraint is specified to ensure that this relationship is always established through instances of the entity *Role*, while for the relationship with a clinical care entity, there must be an entity *AdminEntry*. The constraint *Context Administrative inv: demographic-> forAll (oclType=Role or oclType=AdminEntry)* guarantees this.

3.7 The ArcheER case tool

ArcheERCASE is a computational modeling tool that builds conceptual data schemas based on ArcheER. It is a graphic design software, not a technology-oriented tool, since both the data schema elaborated using this tool and the configuration metadata of this tool are stored in XML format.

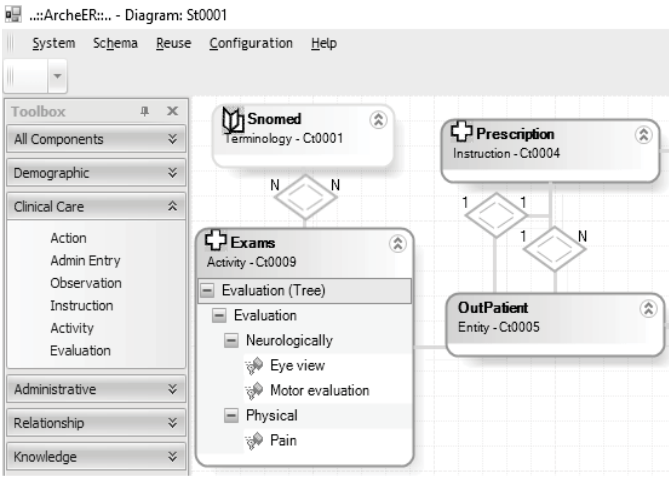


Figure 2. The ArcheERCASE Tool

The main goal of ArcheERCASE is to provide application designers with computer support to assist in the database modeling activities of healthcare applications. Details about ArcheERCASE, including the system prototype architecture, the ArcheERCASE Data Dictionary and the Graphic Module of ArcheERCASE can be found at www.r2asistemas.com.br/ArcheER. Figure 2 depicts the graphic environment of this tool alongside graphic notations.

4 Results

4.1 Experimental design

To validate ArcheER, we conducted two data modeling experiments with two distinct set of human subjects. In both experiments, nine Brazilian professionals with at least two-year experience in conceptual modeling and database design were asked to build two conceptual schemas to model a problem domain. The experiment is based on a hospital scenario located in Northern Brazil, for urgency care. The full description of the problem domain is available at www.r2asistemas.com.br/archeER.

The goal of this research is not to determine the best conceptual data model for HIS applications, but to better understand some important differences between ArcheER and ER conceptual models.

To accomplish that, we computed the time each participant took to complete a given modeling task. Also, for each conceptual data schema generated, we observed whether the uniqueness of EHR was represented, and whether terminologies used in health standards were identified and modeled. Observing the software artifact produced by the ArcheER approach (i.e. each ArcheER conceptual scheme), our experiments measure the difference with respect to the ER model in the following aspects: (i) elapsed time for building a conceptual data scheme; (ii) number of redundant entities produced by each conceptual data schema; and (iii) number of entities that represent terminologies and standards in health of each conceptual data schema produced.

To perform our experiments, each selected participant received the following support instruments: a) instruction about the ER model, b) instruction about the ArcheER approach, c) record sheet, and d) description of the problem domain. In our experiments, the following hypotheses were considered: Hypothesis 1 (H1): The use of the ArcheER approach reduced the time needed to build a conceptual data scheme of a problem domain. Hypothesis 2 (H2): The use of ArcheER warrants the uniqueness of EHR. Hypothesis 3 (H3): ArcheER allows the identification of terminologies and health standards used in a given problem domain.

The variables considered in our experiments are: F1 – Conceptual Modeling Technique for building data schemes; Level of factor T1: Conceptual scheme designed with the ER model (F1→T1); Level of factor T2: Conceptual scheme designed with the ArcheER model (F1→T2). The metrics collected in our experiments are TSB – Time spent for building the conceptual data scheme, QRE – Quantity of redundant entities and QEK – Quantity of entities characterizing knowledge (terminologies and standards). The subjects selected to take part in this study were divided into two working groups (i.e. G1 and G2), chosen by lottery. To eliminate the influence of previous experience of the selected subjects, we used the design of Latin Square experiment 2x2. Considering that Exp1 and Exp2 correspond to the experimental objects that were randomly attributed by lottery to the variables, the experiment design is described in Figure 3a.

Design		
	Exp1	Exp2
G1	F→T1	F→T2
G2	F→T2	F→T1

a) Latin Square

	Group 1			Group 2		
	TSB	QRE	QEK	TSB	QRE	QEK
Stat t	4.73	12.57	10.66	0.40	7.44	4.35
t-student	1.79	1.81	1.75	1.76	1.83	1.81

b) Statistical results of each metric

Figure 3. Design and statistical results

For interpreting the raised hypothesis, we have used the *t* distribution test. This test is often chosen when the average population is less than 30 and there is a normal (or approximately normal) distribution. For this work, the distribution of *t* sampling with n-1 degrees of freedom was adopted. Figure 3b has the values of each metric computed after the application of statistical tests.

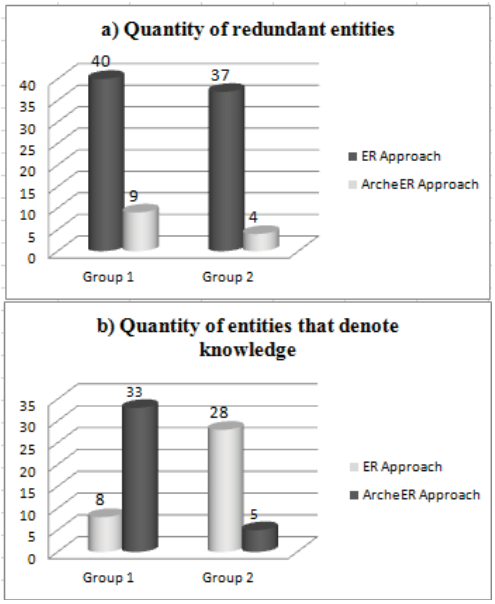


Figure 4. Results of experimental design

Results indicated that the time for building a conceptual data scheme is similar for both modeling approaches. However, for the other two metrics, it is possible to say that, as shown in Figure 4a, the quantity of redundant entities in conceptual schemes designed by the ER Model is greater than the respective number of redundant entities in schemes designed using ArcheER. Moreover, the quantity of redundant entities was reduced in 77.5 % for group 1 and in 89.2 % for group 2 with the adoption of ArcheER approach. Actually, in conventional modeling, for each new role an actor plays in a health domain, new instances are created to represent it, which possibly generates data redundancy in the DBMS – i.e., if a doctor needs to be represented as a patient, a new instance of patient is created by storing information about this person redundantly in the EHR.

Regarding the quantity of entities that denote knowledge, the ArcheER approach identified more entities than the ER approach, as shown in Figure 4b. This increase represents a gain of 75.7% for group1 and 82.1% for group 2. As the use of health terminologies and standards is common in the Health domain, the previous identification of terminologies and standards during the conceptual modeling phase can provide a better understanding of which archetypes are needed for an application to generate knowledge during runtime.

4.2 Modeling an outpatient emergency with ArcheER

In this section, we describe the main difficulties encountered in modeling HIS using traditional approaches, and later we comment on the advantages of modeling HIS using ArcheER. For the sake of didactics, we present in Figure 5 a data schema extracted from a HIS produced by manufacturers of a Health Software in Brazil. This HIS concerns an ambulatory emergency that is performed daily at a Hospital located in Northern Brazil.

Observing the data schema, it is possible to see that the initial difficulty is due to the variety of roles played by the actors in a Health domain, such as workers of a hospital, physicians responsible for patient care, nurses, and other health professionals that sometimes act as health care providers, but occasionally might be seen as a patient who receives care themselves. Besides, the current approaches for database modeling do not provide any constraints to limit this redundancy. Actually, in conventional modeling, for each role played by an actor in a Health domain, new instances are created to represent it, and thus data redundancy may be added to the DBMS.

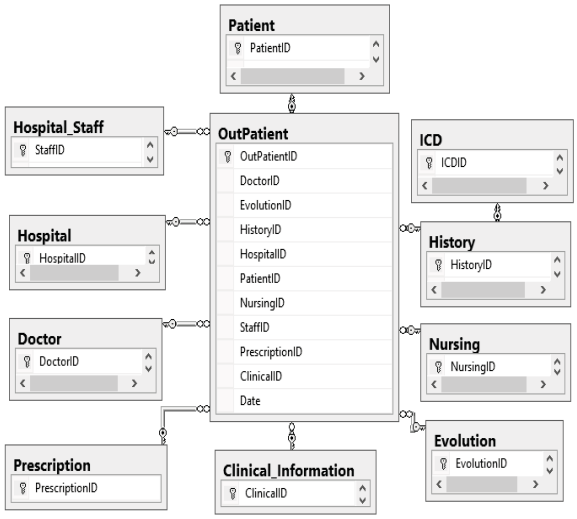


Figure 5. Legacy Data Schema

It is possible to see, in Figure 5, that entities representing demographic information (i.e. Doctor, Hospital, Hospital_Staff, Patient and Nursing_Staff) reflect this modeling practice. In other words, if an actor plays a role, new instances are created for each entity, making their information redundant in the EHR.

In the ArcheER model proposal, actors are modeled in their more generic way, with new instances being created from the roles played. Therefore, an actor may have several roles in an organization and still keep its record unique. As shown in Figure 6, the entity *Person_EHR* represents the most generic characteristics of the actor, while entities *Hospital_Staff, Patient, Nursing_Staff* and *Doctor* represent the roles played by this actor in EHR. To play a role, the actor must have training that qualifies them to perform the referred role – in this case, the *Council* entity illustrated in Figure 6 represents the professional record that the actor needs to have in order to play the role of a physician.

Besides the roles played in a Health domain, an actor may take the form of an organization that provides health services, or that is directly involved in the application context. In this sense, the entity *Hospital* of Figure 6 represents the organization responsible for providing services to the patient. This case shows that the advantages of the ArcheER model go beyond the input of demographic information into the EHR modeling: due to the specified constraints, a demographic entity may only be related to other concepts of EHR (i.e. clinical care, administrative) by means of a role played. In this case, if necessary, only new instances of the roles played by an actor are created, keeping its most generic characteristics preserved, thus ensuring the uniqueness of EHR. As Figure 6 shows, all relationships having an entity that represents patient care (i.e. OutPatient) are established by means of the roles identified in the described application.

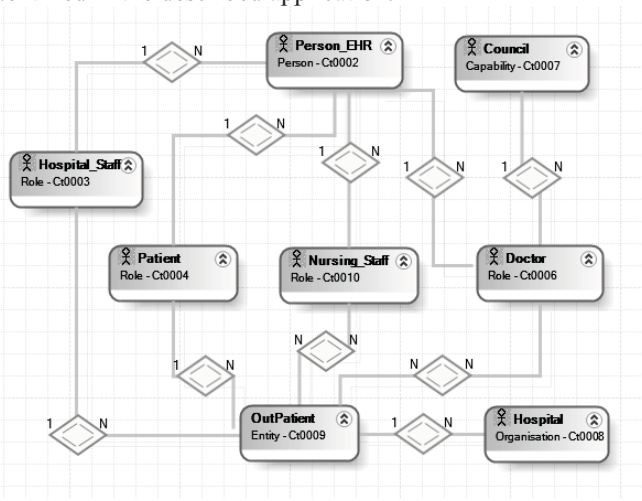


Figure 6. Demographic Conceptual Schema

Figure 7 portrays entities that model clinical care, administrative and knowledge information. Entities *Snomed*, *List_Presc* and *ICD* show the knowledge modeled in the ArcheER conceptual schema. The first entity expresses the terminology and constraints of health care regarding the construction of laboratory examinations, while the entity *Item_Presc* models an internal coding that standardizes the prescription items of a hospital. Finally, the entity *ICD* represents the terminology used to define the patient diagnosis internationally.

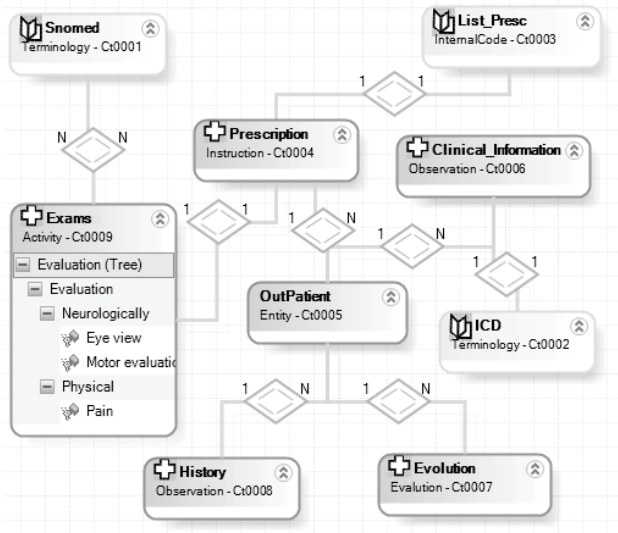


Fig. 7.Clinical Conceptual Schema

For modeling clinical care information, ArcherERCASE provides the following entities types: *Admin_Entry, Observation, Evaluation, Instruction, Action and Activity*. All those types represent abstractions of clinical concepts found in

a Health domain. One can see in Figure 7 that the entities denoting the concepts of patient clinical care are *Exams, Prescription, History, Evolution and Clinical_Information*. The importance of having modeling constructors that represent such concepts is justified by the following aspects: firstly, it helps in the understanding of how to identify and classify EHR clinical information, and secondly, each instance of a clinical care entity represents a potential archetype that may be reused.

5 Conclusions

This paper proposed a novel conceptual data model, named ArcheER, based on archetypes and dual modeling. The benefits of using dual modeling constructors in the conceptual modeling of health information systems have not been studied so far. The modeling constructors that compose ArcheER and the modeling technique selected for diagrammatic representation were chosen from openEHR specifications.

ArcheER is an extension of the ER data model because this model has been recognized in literature as a simple and efficient approach for the elicitation of data requirements, providing the abstraction required for representing the concepts of archetypes through its graphical notation. As main contributions, we highlight a reduction in the representation of redundant entities and a gain concerning the modeling of entities characterizing knowledge. Also, a CASE modeling tool based on ArcheER was presented and a set of OCL constraints was specified, illustrating how the ArcheER model provides uniqueness to the EHR. The specification of semi-automatic generation of archetypes in ADL from the data requirements modeled by an ArcheER conceptual schema is a possibility for future research.

Acknowledgment

This work was partially supported by Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco (FACEPE), under the grants APQ-0173-1.03/15 and IBPG-0809-1.03/13.

6 References

[1] Elmasri R, Navathe S. B. "Fundamentals of Database Systems". Addison-Wesley, 6th ed, 2011.

[2] Marco E, Thomas A, Jorg. R, Asuman D, Gokce L. "A Survey and Analysis of Electronic Healthcare Record Standards"; ACM Computing Surveys, pp. 277–315, 2005.

[3] Mu-Hsing K, Tony S, Andre W.K, Elizabeth M.B, Daniel K.G. "Health big data analytics: current perspectives, challenges and potential solutions"; Int. J. Big Data Intelligence, pp.114-126, 2014.

[4] Späth M. B, Grimson J. "Applying the archetype approach to the database of a biobank information management system"; International Journal of Medical Informatics, pp. 1-22, 2010.

[5] Chen R, Klein G. O, Sundvall E, Karlsson D, Åhlfeldt H. "Archetype-based conversion of EHR content models: pilot experience with a regional EHR system"; BMC Medical Informatics and Decision Making, pp. 9-33, 2009.

[6] Garde S, Hovenga E, Buck J, Knaup P. "Expressing clinical data sets with openEHR archetypes: A solid basis for ubiquitous computing"; International Journal of Medical Informatics, pp.334–341, 2007.

[7] Lezcano L, Miguel A. S, Rodríguez S. C. "Integrating reasoning and clinical archetypes using OWL ontologies and SWRL rules"; Journal of Biomedical Informatics, pp. 1-11, 2010.

[8] Oriol X, Teniente E, Tort A. "Fixing Up Non-executable Operations in UML/OCL Conceptual Schemas"; In: LNCS. Vol.8824, pp. 253-496, 2014.

[9] Dinu V, Nadkarni P. "Guidelines for the Effective Use of Entity-Attribute-Value Modeling for Biomedical Databases"; International Journal of Medical Informatics, pp. 769-779, 2007.

[10] Bernstein K, Bruun R. M, Vingtoft S, Andersen S. K, and Nøhr C. "Modelling and implementing electronic health records in Denmark"; International Journal of Medical Informatic, pp. 213-220, 2005.

[11] Jeffrey A. L, Jeffrey L. S, Blackford M. "Method of Electronic Health Record Documentation and quality of primary care"; J Am Med Inform Assoc, pp.1019-1024, 2012.

[12] Georg D, Judith C, and Christoph R. "Towards plug-and-play integration of archetypes into legacy electronic health record systems: the ArchiMed experience"; BMC Medical Informatics and Decision Making, pp.1-12, 2013.

[13] Bernd B, "Advances and Secure Architectural EHR Approaches"; International Journal of Medical informatics, pp.185-190, 2006.

[14] Martínez C. C, Menárguez T. M, Fernández B. J. T, Maldonado J. A. "A model-driven approach for representing clinical archetypes for Semantic Web environments"; Journal of Biomedical Informatics, pp.150–164, 2009.

[15] Buck J, Garde S, Kohl C. D, Knaup G. P. "Towards a comprehensive electronic patient record to support an innovative individual care concept for premature infants using the openEHR approach"; International Journal of Medical Informatics, pp.521-531, 2009.

[16] Arlow J, Neustadt I. "UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design". Addison-Wesley, 2nd ed, 2005.

Classroom Attendance Detection using a Wi-Fi Positioning Algorithm

Mao Zheng[1], Sanhu Li[2], and Hao Fan[3]
[1]Department of Computer Science, University of Wisconsin-La Crosse, La Crosse, WI, USA
[2]Department of Computer & Information Science, University of Delaware, Newark, DE, USA
[3]School of Information Management, Wuhan University, Wuhan, Hubei, China

Abstract *– Wi-Fi positioning plays an increasingly important role in indoor positioning techniques since it is possible to locate the position of almost every Wi-Fi compatible device without installing extra software or manipulating the hardware. It will make use of existing Wi-Fi infrastructure, although it was never designed to do so. This paper discusses the design and implementation of a Wi-Fi position algorithm that is used to detect the classroom in a class attendance mobile application. The results are promising.*

Keywords: Wi-Fi, WLAN, Wireless Indoor Positioning, Indoor Location Sensing, Wireless Localization.

1 Introduction

In 1978 the first GPS satellite was launched [1] and in 1995 GPS worked to full capability for the first time [2]. Unfortunately, the satellite signals were not strong enough to work indoors. In 1997 IEEE Standard 802.11 was set and the first version of Wireless Local Area Network (WLAN) was born.

WLAN, Wi-Fi and IEEE 802.11 all mean the same thing: they determine the industrial standard for wireless data transmission. The latter is the most used expression. Wi-Fi uses electromagnetic waves to transmit data over the airwaves. The electromagnetic waves spread out evenly and lose more and more of their signal strength with increasing radius. This loss of signal strength is due to energy transformation because, in physics, energy is never lost, but instead converted. Consequently the amplitude of the signal becomes smaller and smaller. In summary, if the distance to the station is increased in any direction, the signal strength will decrease steadily.

Nowadays Wireless Local Area Network technology can be found in almost every building. This widespread infrastructure offers the possibility to locate mobile devices in an economical way. Position determination using Wi-Fi technology has the advantage that it can perform indoors and outdoors, in a different way to GPS. And, although Wi-Fi was never made for positioning, it is more accurate than a Global System for Mobile Communications (GSM) indoor positioning [3]. In some cases, it is also more accurate for positioning an object outdoors. By using Wi-Fi Positioning Systems it is possible to locate the position of almost every Wi-Fi compatible device without installing extra software or manipulating the hardware. In the course of time many methods that were initially used with other positioning technologies were applied to Wi-Fi positioning. Wi-Fi positioning also allows the use of location-based services (LBS) indoors, which allows for different industrial uses. Useful applications of this technology are, for example, for indoor navigation at shopping malls or finding a lost child in an indoor area. Lost devices or items can also be found with this technology. Additionally, this technology is especially useful for hospitals because sometimes when staff move certain pieces of equipment it can be hard to find these items again right away.

The indoor environment has many disruptive factors like walls, windows, doors, and so on. If a wave bumps into a different material, it converts more energy than in the air. During this process, the signal strength is decreased more rapidly, because the energy is transferred to the material as heat. Furthermore, the signal is also reflected from the material. An additional problem is the wireless overlay. In an office or apartment building, there are several dozen wireless stations that provide interference. A positioning system must be able to handle these problems and deliver good results. A higher accuracy is also required for indoor usage because it is important to locate a user in the right room. A few meters can make a big difference. Wi-Fi is especially useful in an indoor setting because there are no other positioning services running.

We are interested in exploring the Wi-Fi positioning algorithm in a class attendance mobile application to detect if the student is physically in the classroom. The testing result of our algorithm is promising.

The paper is organized as follows: In section 2, we introduce the class attendance mobile application. In section 3, we present our Wi-Fi positioning algorithm. In section 4, we discuss the results of our algorithm implementation. Section 5 concludes the paper and outlines the directions of our ongoing research.

2 Class Attendance Mobile Application

Recording classroom attendance by hand for large classes can be a time consuming task. We have developed a mobile application and registration system that allows for the automatic collection of attendance information. The information is collected with mobile devices and transferred and stored in a web-based student registration system. Both the mobile application and the registration system are part of this project. The registration system serves as a test-bed for the recorded data and can be used to analyze the data.

There are two main challenges when recording class attendance. Firstly, it is important to be certain the student physically comes to the classroom. Secondly, the application needs to verify the recorded data is for the correct student. The current solution provided in this project is to let the instructor generate a QR-code during class time, in the classroom. The student will use his/her mobile device to scan the QR-code and submit the scanned data along with his/her student ID to the registration system during the recording time period. Only students who are enrolled in the class are able to submit the scanned data and this data can only be recorded once during a valid time period. The system also checks whether the student is actually in the right classroom for this class. The instructor can view the recorded class attendance information and each student can verify that his/her attendance was recorded correctly.

3 Proposed Wi-Fi Positioning Algorithm

Indoor positioning is a difficult problem our project needed to slove. Compared to other locating methods, GPS does not locate the position of an object well indoors. Using specially designed hardware is costly and hard to maintain. Wi-Fi positioning is the best solution because it supports indoor positioning and it can easily distinguish objects on different floors. Moreover, the Wi-Fi sensor is built into every mobile device nowadays.

There are currently two major types of algorithms to do Wi-Fi positioning: the triangle algorithm and the position fingerprint algorithm. Both of these algorithms are too complex and hard to implement. Different applications may require different types of location information. In this project, we are interested in campus classroom location and we propose an indoor Wi-Fi positioning algorithm based on the idea of the position fingerprint algorithm but in a simpler version. We made some assumptions to simplify this algorithm. It is assumed that when doing the scanning process, if the student is close to the correct classroom, the student should be in that classroom. We made this assumption because the Wi-Fi sensor in smart phones may not give an accurate result.

The proposed Wi-Fi positioning algorithm is described below in a number of steps:

1. Establishing a profile for each classroom.

This step is a manual process to configure the classroom. For each classroom, we scan and record the routers whose signals can reach to this classroom. We need to repeat this process a number of times in order to select the top K routers that has a signal strength greater than N and is always available, where K and N can be fine tuned in the implementation. In the database, each classroom has a table stored with the top K routers' MAC addresses and SSID.

2. Identify the classroom where the student is in:

   a) Scan the signal strength at the object point, sort the routers by the signal strength and record the top K routers' MAC addresses at that moment.

   b) Search all the records in the database that have the same MAC address with one of the scanned routers' MAC address.

   c) Group the records in every classroom and count the number of records in every group.

   d) The classroom with the most records (the number should be greater than M) is the nearest classroom.

Through above steps, we can obtain the nearest classroom.

In our project, we also require the instructor generate a random QR code in the classroom during the class attendance recording time. The student will need to scan the QR code and submit the scanned record via one of the Wi-Fi signals in the classroom. The recording system will check the received information and verify if the student is currently enrolled in the class, and if the class attendance recording time is valid. The class attendance recording time must be within the range of class time existed in the system. Only when a student currently enrolled in the course submitted the scanned QR code during the valid recording time via one of the classroom Wi-Fi signals, the class attendance is properly recorded in the system.

Figure 1 is the screen shot of a location's routers' information obtained by our WiFiScan app. All the Wi-Fi signal strengths and BSSID are listed.

Figure 1 Using a Mobile Device to Scan Wi-Fi Signals

The sample Java source code for scanning through all the Wi-Fi signals in a mobile device is shown in Figure 2.

```
private ArrayList<String> getBSSIDs() {
    List<ScanResult> results = wifi.getScanResults();
    Collections.sort(results, new Comparator<ScanResult>() {
        @Override
        public int compare(ScanResult lhs, ScanResult rhs) {
            return rhs.level - lhs.level;
        }
    });
    ArrayList<String> bssids = new ArrayList<>();
    int size = results.size() < MAX_ROUTERS_NUM ? results.size() : MAX_ROUTERS_NUM;
    for (int i = 0; i < size; i++) {
        bssids.add(results.get(i).BSSID);
    }
    return bssids;
}
```

Figure 2 Source Code for Getting Wi-Fi Signals

After each classroom's Wi-Fi profile is established in the system, the student can locate him/herself once he/she arrives at the classroom. Figure 3 is the screen shot of the location result in an Android device.

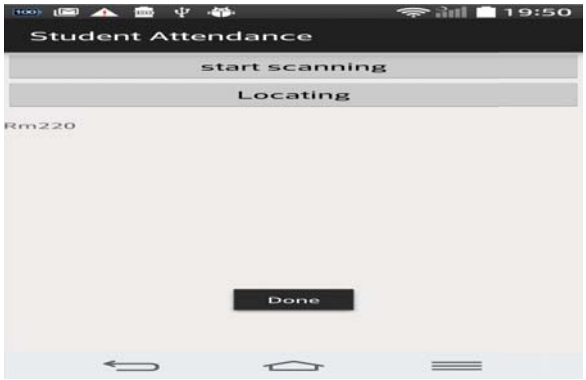


Figure 3 Locating the Student

4 Related Work

There are many different approaches for locating a mobile device using Wi-Fi technology. In general, the methods need to know the position of the Wi-Fi stations (access points) as a reference point. Afterwards, the approximate position of the mobile device can be obtained [4]. A prerequisite for having a good-working Wi-Fi positioning system is adequate coverage of the access points. This coverage is called the Basic Service Area (BSA). The expression of the BSA determines which positioning method is the most suitable. The methods differ in the minimum required number of stations and its accuracy. This varies between building part accuracy and room accuracy to an accuracy of a few meters difference.

1) Methods based on proximity sensing are among the simplest and fastest, but they are also imprecise. A position calculation can be done with just a single station.

2) Methods based on trilateration needs at least three fixed points to determine a position. The challenge for a trilateration method lies in the best possible determination of the distance between the station and the device. Methods that are based on time measurements have to guarantee a good synchronization on the stations or mobile devices. On the other hand, methods that are based on the signal strength have problems with interferences and reflection. Therefore, they are probably better suitable for outdoor detection than for indoor detection.

3) Methods based on triangulation use geometry to determine the angle of the arriving signals. It requires at least two stations and the modification of the hardware [5].

4) Methods based on pattern recognition uses a previously created database of signal patterns, which need to be matched for positioning only. Fingerprinting, also called location patterning [6], does not need modification of the hardware. Furthermore, no time synchronization is necessary between the stations. Before a position can be determined, the entire area in which the positioning is supposed to work must be recorded.

Our Wi-Fi positioning algorithm is based on the idea of a fingerprint algorithm. We established a set of Wi-Fi BSSID data for every classroom and stored the information in the database. This information will be used in the locating phrase. The sample Java code for locating the classroom is shown in Figure 4.

```
@Override
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    String[] bssids = req.getParameterValues("bssid");

    if (bssids != null && bssids.length > 2) {
        StringBuffer qms = new StringBuffer("?");
        String sql = "SELECT crid, count(crid) as n FROM crdata where bssid in (%s) group by crid order by n desc";
        for (int i = 1; i < bssids.length; i++) {
            qms.append(", ?");
        }
        sql = String.format(sql, qms.toString());

        List<Object[]> result = DataCenter.SQLQuery(sql, bssids);
        if (result.size() > 0 && Integer.parseInt(result.get(0)[1].toString()) > 2) {
            Integer crid = Integer.parseInt(result.get(0)[0].toString());
            resp.getWriter().println(findClassroomByCrid(crid));
        } else {
            resp.getWriter().println("No results found.");
            throw new RuntimeException("No result found");
        }
    } else {
        throw new RuntimeException("Need more wifi information");
    }
}

private String findClassroomByCrid(Integer crid) {
    String hql = "from Classroom where crid = ?";
    List result = DataCenter.query(hql, crid);
    if (result.size() > 0) {
        Classroom cr = (Classroom) result.get(0);
        return cr.getCrname();
    } else {
        throw new RuntimeException("No classroom has such a crid " + crid);
    }
}
```

Figure 4 Sample Source Code for Locating the Classroom

5 Conclusions

Wi-Fi has become a wide spreading technology. However it is a technology that was never designed for localization. Nevertheless, Wi-Fi positioning performs well in comparison to other positioning technologies, and has the favorable advantage that it is based on an existing infrastructure. When indoors, positioning fingerprinting delivers the best results. However, a lot of effort is required to develop a good fingerprint position algorithm.

For this project we created an automated class attendance system where instructors can use the system to record students' attendance. The Wi-Fi positioning algorithm checks whether a student is physically in the classroom. It was very exciting when a simplified positioning algorithm was developed.

Our positioning algorithm is sufficient for the attendance checking task. We recorded the routers' MAC addresses that were giving out a strong signal for the test classroom, in the database. When locating our position, we compared the strong signal routers with the ones stored in the database. If more than three of the routers are the same, we believed the user was close to that classroom. Based on our assumption, we consider the user is in the classroom.

So far, we finished the Android mobile application for the attendance checking application, and the web-based student information system. We used the http protocol in our clients to communicate with the server, which means it's very easy to build clients for other platforms, such as IOS.

In the next step, more work can be done to automatically upload the Wi-Fi information into the database after the data is collected. Currently, we are scanning the Wi-Fi signals in every classroom and manually entering the information into the database. Figure 5 shows the screen result after scanning the routers' information. Figure 6 is the screen shot that we manually added of the scanned information information into the database.

We have conducted testing for ten classrooms at the University of Wisconsin-La Crosse. The results are promising.

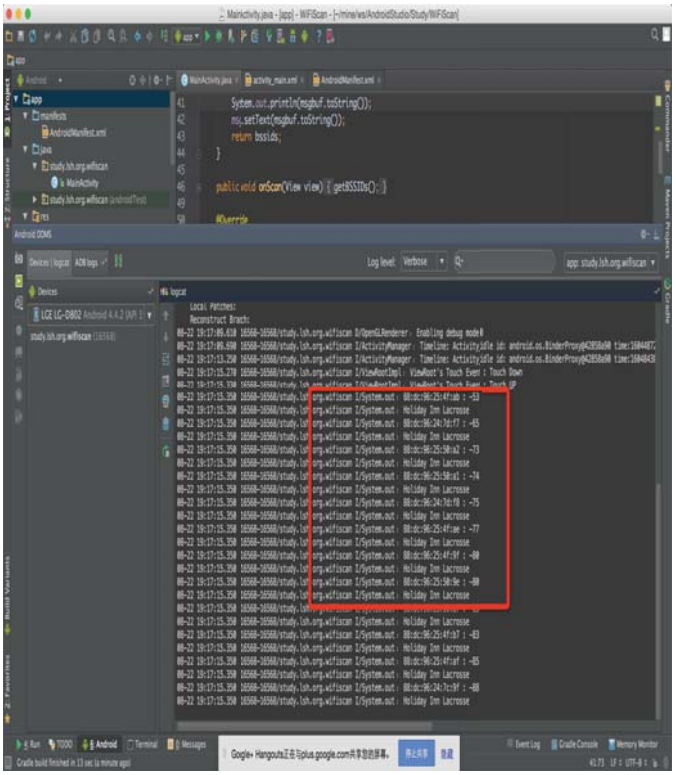


Figure 5 Routers' Information Shown on the Screen

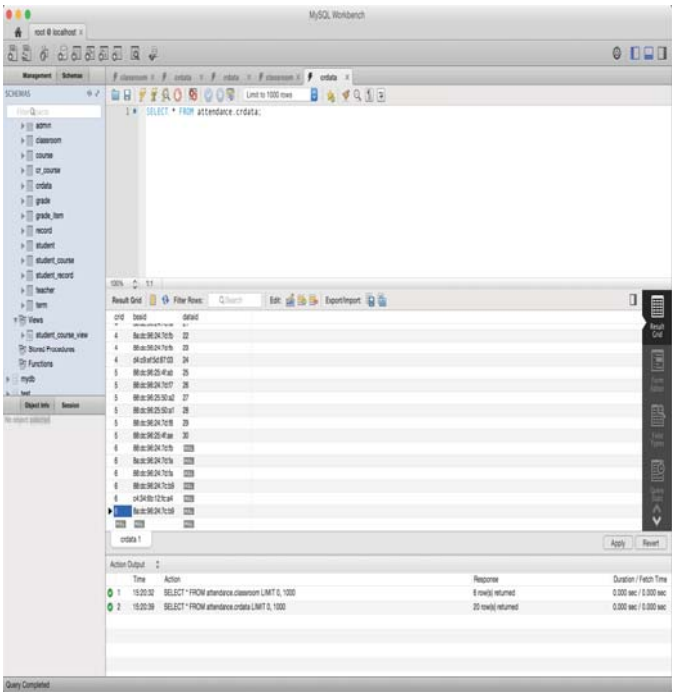


Figure 6 Storing Information into the Database

6 References

[1] Wilfried Ley, Klaus Wittmann and Willi Hallmann *Handbook of space technology* 2009 John Wiley and Sons, Ltd

[2] Elliott D. Kaplan, Christopher J. Hegarty Understanding GPS: principles and applications, 3rd ed. London, England: Artech House, 2006.

[3] Heikki Laitinen, Jaakko Lahteenma¨ki, Tero Nordstro¨m *Database Correlation Method for GSM Location*, VTT, FINLAN, 2001.

[4] Jami,I., Ali,M., Ormondroyd,R.F, Comparison of Methods of Locating and Tracking Cellular USA: Dept. of Aerosp. Power & Sensors, Cranfield Univ., Swindon, UK 2010.

[5] Dipl.-Wirtsch.-Ing. Rene Dnkler AoA - Angle of Arrival, Online: http://www.iis.fraunhofer.de/bf/ln/technologie/aoa/ last access: 01/18/2012

[6] IshratJ, Quade,BinghaoL, Wendi(Patrick)Peng, Andrew G. Dempster, *Use of Fingerprinting in Wi-Fi Based Outdoor Positioning*, 1st ed. The University of New South Wales, Sydney, Australia, 2007.

[7] Rui Ma, Qiang Guo, Changzhen Hu, and Jingfeng Xue, *An Improved WiFi Indoor Positioning Algorithm by Weighted Fusion*, Sensors (Basel, Switzerland), 15(9): 21824-21843, Sep. 2015.

[8] Hui Liu, Houshang Darabo, Pat Banerjee, and Jing Liu, *Survey of Wireless Indoor Positioning Techniques and Systems*, IEEE Transactions on Systems, Man and Cybernetics – Part C: Applications and Reviews, vol. 37, No. 6, November 2007.

[9] Robin Henniges, *Current Approaches of Wifi Positioning*, Service-Centric Networking, Seminar WS2011/2012.

Development of an Idol Entertainment Application with Focus on " Hagashi" Act

Yosuke Kanai [1], **Takayuki Fujimoto**[2]
Kujirai 2100, Kawagoe-City, Saitama, Japan
kanaip@icloud.com, me@fujimotokyo.com

Abstract - *Videogames and Idols are typical contents in our country. In Japan, there are many contents such as games and idols. Since the 1980s, they have become very popular and have been exported to the other countries in the world. However, they are similar and stuck in a rut. Therefore, most of the people feel that they are not interesting. This study created the combination of the game and the idol; not only just an idol but also fans and staff. The theme of this study is "the problem between idols and fans". Based on them, we have developed an entertainment application which makes the users interested in the*

Keywords: apps, game, idol, OTAKU, HAGASHI

1 Introduction

The app by this study was developed in for free uploading because free apps have more distribution than paid apps. It is often the case to obtain a profit in the advertising with free apps comparing with getting the sales of paid apps. More free apps are downloaded than paid apps. The popular apps are often free. Therefore, the business model that earns an income by clicks on an ad or by sales of the additional contents with charge is more general than the model of the paid apps. We investigated the elements of possibility related to a hit app. This analysis is based on the reference to the famous web site, "the apps download ranking: Game genre" (It is provided from the APPANNIE.)

It is a statistical review of that ranking for the top ranked apps. The following factors were scooped. (We focused on only the factors, which account for over 5 percent of the total.).

"It is suitable for playing in the boring time"

It means that a lot of people are playing the game in boring time on the train or in the class of the university.

"It can be some refreshment for the stress"

Achievement of clearing the game leads to this feeling.

From these, it was found that the ideal game app has a lot of goals, and provides immediate judgements at loss and gain.

"Control is simple."

Smartphones are not appropriate for complex controls. Therefore, smartphone apps are preferred to require a simple operation.

Our app is classified as an idol game. There are many idol game lovers in Japan because its characters are highly appreciated. However, the theme of this study is not just an idol. We focused on "today's problem between idols and fans"

In recent years, Japanese idol market has changed greatly because CD sales is no longer big as before. Therefore, the idol business was shifted to obtain a profit by sales of promotion items in lives and events. As an alternative, the idol industry is pushing purchase of a ticket that comes with a CD. It is a ticket that allows a handshake with an idol at the event. However, a problem occurred. That is "the problem of the handshake events". A CD contains one ticket. Therefore, fans buy the surprising number of the CDs. Actually, each and every one of some fans bought more than 5000 CDs. CD sales went up in this business. Handshake events with idols were successful, however most of fans began to seek direct contacts because by handshakes they can feel that they are most close to the idols. Handshaking with idols made fans want more direct contacts As a result, fans have started buying the CDs just to get a ticket for legalized and direct contacts. This was taken up by each media the idol business was changed. It has become a social problem of today. In the indie idol industry, some groups dare to dive into crowds of fans to obtain popularity, using more direct contact. Touchable Idols mean the decline of the idol business.

2 Purpose

In this study, we devised and developed the app that includes the quite new elements described in the previous chapter. The Developed app is a game that combines an idol game and a falling block game. Main characters of the most idol games are idols, and they are the subject of actions. However, the main character of the developed app in this study is "an event staff". There are two new factors in this study.

One is the fact that the player character is the staff. The target of the game is to stop the Otaku approaching to the idol (Otaku is a popular name for enthusiastic fans in japan). Achievement is judged by how secure the user can protect the idol within the time limit. The number of caught Otakus becomes the score.

The other is incorporated “Hagashi” as a game action. When the Otaku touches the idol, the staff (player) will pull away the Otaku from the idol. This action is called “Hagashi”. This act has been actually carried out by the staff at the handshake events. As far as I know, there is no game in which HAGASI is the main character.

By incorporating Hagashi in the game, we targeted the new game design. In the normal action games, to recover from the bat status, it is necessary to wait for the lapse of time or to use some items. In our app, the bad status is restored by Hagashi, which is the player's action. Players need to perform two actions, “Blocking contacts" and "Pulling the Otaku from the idol". We were aiming to parody a social phenomenon: the problem between the idols and the fans through the entertainment app. to get the attention of the users. The following are the reasons why Hagashi became the necessary action.

Hagashi has become indispensable at the idol handshake events. Hagashi is the idol’s bodyguard and has a role to pull the fans away preventing a handshake over time Also pulling fans away aims to protect the idols, in order to manage the events smoothly. Presence of Hagashi has become a necessary change with the times. This means that they are the “symbol of today's idol culture”.

3 Precedent

3.1 Falling block game

The app developed in this study is classified as a falling block game. Falling block game has been a popular game for a long time.

3.2 Idol game

In the game genre, there is an idol game in which a fictional idol or an animated character of a real idol appears. Both still are popular in Japan.

“The high school life with Miho Nakayama” (1987), in this game a real popular idol "Miho Nakayama" was featured Idol games in which the real idol is used as a character, was a major genre in the early idol games.

· “HIKARI GENJ Roller Panic” (1989) is a game in which male idols are featured. After this game idol games with motif of real idols were not developed

· “Idol HAKKENDEN” (1989) is a game in which a fictional idol appears. Idol adventure games and idol-producing games with a fictional main characters have been a main stream.

4 Proposed application

The app developed in this study is classified as a falling block game. In the Falling block games, the user will get a score by catching the items falling from the top of the screen, targeting a high score. We changed a point of view in this study and applied "falling items" to "approaching Otakus", and “getting items” to “catching Otakus”. Score is determined by the number of the caught Otakus. If the user wants to get a high score, he or she must catch more Otakus.

4.1 App mechanisms

The App has been developed as an application for iPhones. It has been developed by using cocos2d-x in Xcode. Development language is C ++. The application is comprised of three screens at the present stage of the development. They are the "title", "main game", and "collection".

From the title screen, the user can move to each screen. Currently, iOS devices in the market have a wide variety with different screen sizes. Therefore, to cope with different resolutions, the image and buttons to be displayed are adapted to multi-devices by getting a resolution in advance, and setting/displaying them along the relative coordinates from the screen size with implementation of hard coding.

Incidence of the character (the idol fan), which is an important element to enhance the entertainment, is at random. However, the game is designed to compete by the score count of the items. Therefore, large variations of the appearance ratio would occur. So we had to create a random number generator in order to adjust the quality of the random number. Appearance ratio has been set so as to increase exponentially with the lapse of time to play the game. Thus it prevents the game from monotonicity.

In this study, we have linearly changed the appearance ratio of items by the remaining number of seconds.

“Initial Appearance ratio (%) + Increase of the Appearance ratio (%) ×elapsed seconds”

We have set the appearance ratio in this equation. Too large number of occurrences of the item is prevented by setting the maximum value.

4.2 The start-up screen

It is shown in Fig 1. When you tap the button in the "Collection", and you can move to the collection screen. A

variety of Otakus who appears in the game is recorded in the collection.
By tapping the "Start", the game will be started. (Fig.2)

Fig1: Start-up screen

Fig2: Game-start screen

4.3 Game-playing screen

Displayed on the top of the screen is the "enthusiastic fan (Otaku)". (Fig3) They appeared at random from the top of the screen, comes approaching toward the idle at the bottom of the screen. Girls that displayed at the bottom of the screen is "idle". Character of the man wearing a hat is the "player character (he called HAGASHI)".

Fig3: game-playing screen

You can move the player character by tapping and dragging. It follows your dragging from the tapped point of the screen.
Like a real Hasashi staff, his movement is limited. The player character can be moved only to the horizontal direction. The idol on the bottom of the screen starts to move to side to side just after the game start. If you can stop the Otaku, the score will increase. (Fig4)
If the Otaku is in contact with the idol, the score will be reduced. (Fig5)

Fig4: Increase of score

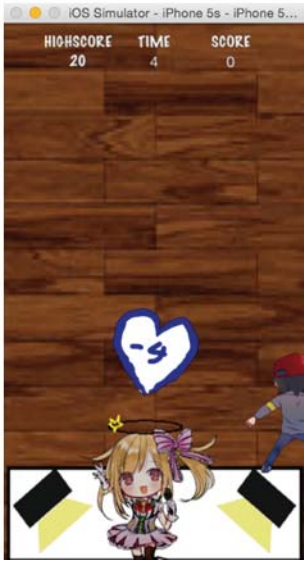


Fig5: Decrease of score

When the Otaku touches the idol, Otaku will keep sticking to the idol. This state is referred to “crash"(Fig. 6)

Fig6: State of crash

If you go to the state of “crash”, the player character will be uncontrollable. The cancellation of the crash, you need to pull away the Otaku. Pulling away, "Hagashi" can be done by tapping and sliding the idol. Out of the approaching characters, there is a "producer". If you catch the "producer", the score will be reduced. If you catch he idol producer, the score will be reduced. (Fig7)
He is an important character in order to make the idol more popular. If you catch him, score will be decreased. Therefore, you need to avoid catching him.

Fig7: idle producer

Information on the game will appear on top of the screen (Fig8). The highest score ever will be shown under "High score”. The remaining time will be displayed under "Time”. The score of the game which you are playing will be displayed under "score".

Fig8: information on game

4.4 The collection screen.

You can move to the collection mode screen by tapping the "collection" button on the start-up screen, You can enjoy the collection book, which records details and remarks of "Otakus" which you have already caught in the collection mode. These will be acquired by the number of times playing with the high score in the game. The looks and personalities of the "Otakus" are striking and unique. You can enjoy some types of typical "Otakus", if you are interested in subcultures.

5 Conclution and future research

In this study we found the common points of the hit game apps, and have developed a simple game application for "short time”. The new element, the “idol” was incorporated into the falling block game, to avoid "boredom”. In addition, by obtaining a novelty with focus on Otakus, a social phenomenon is parodied.

Regarding future tasks, the adjustment of the game balance is needed. Since there is no limitation with respect to the operation of the player, the player character can move to the tapped position too easily. Also, the collision detection for each character is done rather roughly, and consequently protecting the idol is currently very easy. Setting the speed limit for the player character’s move to the touch position and reconsidering the collision detection are required for improvement. By introducing new operations or gimmicks in addition to “stopping” or “avoiding”, the app is expected to enhance the entertainment with these improvements, we would like to bring better distribution to the app.

The app developed in this study contains a metaphor about the Japanese idols. The idol culture is very active in Japan. It is difficult to become a popular idol in the presence of a large number of idols. However, the number of young people who dream of becoming the idols has not been reduced. They are intended to collect customers in a variety of events in this society. These events are successful most of the cases, but instead the idols turn out to lose the essence as idols. Indie idols has started being the idols just to work for the events. This means that the concept as the idol is collapsed as a whole. The current state of Japanese idols is referred to in this study. This issue should be paid more attention.

6 References

[1] Akira Aizawa, “Game Theory Training”,Kanki Publication, Inc, 2003

[2] Takahiro Wahanabe, “Game Theory (Illustrated Trivia)”,Nathumi Publication, Inc, 2004

[3] Koji Fukada, “Why social game is addicting? Customer satisfaction Gamification is change”,Softbank creative, Inc, 2011

[4] Kouyo Mathuura, Eji Furuki, Kenji Saito, “Recipe of cocos2d-x development”,Syowa System Publication, Inc, 2013

[5] Kouki Miki, “Smartphone game development in the cocos2d-x”, Gizyuthu Hyoron Publication, Inc, 2015

[6] Kosho Mori, “Gently start school in the development of the iPhone app”,MyNavi Publication, Inc, 2012

[7] Scott Rogers , Yosuke Shiokawa, “Game design of "level-up"”,Olayly Japan Publication, Inc, 2003

[8] Kazuya WADA, Masanori TAKANO, Ichiro FUKUDA“Analysis Of User Playing Continuance On Smartphone Game.”, Entertainment Computing Symposium2014 (EC2014), pp 304-306, 2014-9

[9] Tomoe SEKINE“How use the internet long time in 20s and 30s.”, Broadcast research and survey 63(4)、pp 32-43, 2015

An Approach for Generating Class and Sequence Models

Márcio A. Miranda[2,3], **Marcos G. Ribeiro**[4], **Renan D. Tavares**[3], **Thiago H. B. Dias**[3], **Humberto T. Marques-Neto**[3], **Mark A. J. Song**[1,3]
[1]Department of Computer Science / Centro Universitário UNA / Belo Horizonte, MG, Brazil
[2]Department of Computer Science / Federal Institute of Minas Gerais / Ouro Branco, MG, Brazil
[3]Department of Computer Science / Pontifical Catholic University of Minas Gerais / Belo Horizonte, MG, Brazil
[4]Department of Computer Engineering / Federal Center of Technological Education of Minas Gerais / Timóteo, MG, Brazil
marcio.assis@ifmg.edu.br, marcos.ribeiro.timoteo@cefetmg.br, {rdtavares, thdias}@sga.pucminas.br, {humberto, song}@pucminas.br, mark@prof.una.br

Abstract— *The use of domain-specific languages has been gaining traction in the requirement analysis and discovery process due to features such as establishing standardized team communication, allowing the automation of certain stages of the process, and bringing productivity gains without compromising quality. In this paper we proposed and implemented the Language of Use Cases to SEquence Diagram (LUCSED), a domain-specific language for the textual specification of use cases and, through our LUCSED-tool, automatically generate use case, sequence and class diagrams. To assess the viability of our solution, we carried out several tests aiming to cover a diversity of scenarios found in software development. Our approach can be useful in requirement analysis and modeling, and seeks to minimize problems present in natural language specifications, such as: uncertainty, ambiguity, complexity and an intense dependence on domain knowledge by specialists.*

Keywords: Automatic Generation, Class Diagram, Domain-Specific Language, Sequence Diagram, Use Case Specification.

1. Introduction

Currently, there are many different ways to analyze and specify software requirements, such as user stories, models and formal languages. To [1], capturing and mapping requirements is mostly done through textual use case specifications, so that even laymen are able to understand them. However, the inherent use of natural language can negatively affect artifact quality, due to difficulties such as ambiguity, redundancy, inconsistency and incompleteness [2], [3].

According to [4], representing requirements in a standardized way which is easily understood by all participants of a project can mitigate the problem, and even bring a certain level of automation to the process [5]. Thus, to reach this goal, it is necessary to formalize structural and behavioral aspects of use case specifications [2], [6], [7].

One of the resources that have been generating interest are *Domain-Specific Languages* (DSL). Although limited in scope, they define a communication standard between engineers and domain specialists. They can also help minimize natural language uncertainty and ambiguity. Readability, understandability and productivity are also fundamental characteristics to justify the use of DSL to describe use cases [8], [9], [10].

In this paper we propose and implement an external DSL called *Language of Use Cases to SEquence Diagram* (LUCSED), which defines standards of a language which allows requirement analysts to specify textual use cases. We also propose a tool called LUCSEDTool which comprises automatic generation of use case, class, and sequence diagrams from the specification. The tool maps the specification following language rules for the Extensible Markup Language Metadata Interchange (XMI) input standard, supported by Unified Modeling Language (UML) modeling software, such as *Astah* [11] and others.

The remainder of this article is structured as follows. Section 2 briefly describes some of the relevant related work developed in the last two decades. Section 3 presents our proposed approach, highlighting the main grammar rules in LUCSED, the sentence patterns supported by the language and the artifact generation process. We present and discuss our results in Section 4. Finally, we conduct a conclusive analysis and state our final considerations.

2. Related Work

In the beginning of the past decade, [12] proposed a set of rules to normalize textual use case specification. Following this standard, analysts can infer which classes, objects, associations, attributes and operations belong to a use case and generate sequence diagrams from this information. However, this is done manually instead of automatically, since no tool to test the proposed standard has been implemented, relying instead on specialist knowledge.

A few years later, [13] proposed a solution to explore the most common problems in use case modeling, showing inconsistencies between use case models and their textual specifications. Later, [14] presented a meta-model to describe textual use cases. It defines a textual representation of use case behavior, easily understood by readers who do not have full command the subject. To model use cases narratives, they developed the *Narrative Use Case Description Toolkit for Evaluation and Simulation* (NaUTiluS) tool.

The work [15] proposed and implemented a tool called Procasor to automatically generate executable code from use case specifications. Similarly to our work, they defined a standard to specify textual use cases in a format recognized by their tool, but they did not generate UML models, and LUCSEDTool does not generate executable code. In [4], authors present a language for use case specification based on Xtext and called SilabReq. From use cases, the tool generates domain models, a list of system operations, a UML use case model, and state, activity and sequence diagrams. In the following year, the same author proposed dividing specifications into different abstraction levels, since use cases are used by people in different roles with different needs during software development, from end users, requirement engineers and business analysts, to project engineers, developers and testers.

In the papers [16], authors proposed and implemented a tool to automatically generate sequence diagrams from use case specifications written in the English language. The solution uses the natural language parser Stanford Parser [*The Stanford Natural Language Processing Group*] [17] to identify objects and interactions among them from use case specifications. The parser analyzes sentences and classify words into adjectives, adverbs, articles, pronouns, nouns, verbs, etc. Therefore, the solution ignores situations in sequence diagrams such as combined fragments and messages to self, and does not generate other UML models.

In the following year, the work [18] proposed an approach called *aToucan*, based on existing solutions, to automatically generate UML analysis models comprising class, sequence and activity diagrams from a use case model. They also used natural language specifications (English) and used Stanford Parser [17] to map specifications.

However, natural language is free, ambiguous and defines no team communication standard, making automatically generating artifacts, and consequentially traceability between specification, model and source code, very difficult.

Conversely, properly designed domain-specific languages, unlike natural languages, establish a common language to be used by all team members, and their formality standardized communication between stakeholders [19], [20].

Additionally, DSL allows solutions to be expressed at application domain level, enabling business analysts to understand, validate, change, and even develop features using the language. We can also highlight that domain specialists and software engineers are already used to programming language formality, and this reduces language learning time and optimizes resource reuse [5], [21].

3. Proposed Solution

Our proposed DSL (LUCSED) contemplates use case detailing and along with LUCSEDtool enables the automatic generation of UML diagrams in requirement-oriented software processes, following object orientation principles and the *Model-View-Controller* (MVC) architectural model.

LUCSEDtool is a support tool used to map relevant information in use case detailing into an XMI input standard, recognized by UML modeling software. It is also able to generate artifacts directly in the .astah format, and if users already have Astah installed in their computers, they can choose to open the diagram in Astah in the correct format. LUCSEDtool is available for download at *https://github.com/assismiranda/LUCSEDTool*.

The conversion of use case specifications into UML diagrams follows the process shown in Figure 1. The process consists in mapping textual use cases written in the LUCSED language into object oriented models, specifically use case, classes and sequence diagrams. Note that a possible subsequent stage is the automatic generation of model source codes, since most modeling tools are capable of generating source code from UML models.

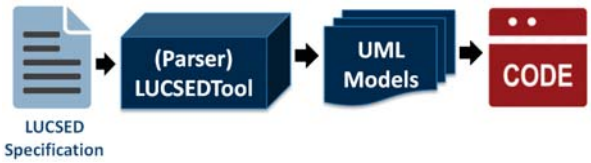


Fig. 1: LUCSED-to-UML transition

When designing the DSL, we sought to follow a set of relevant requirements, to achieve features like simplicity and objectivity, in order to offer a clear and self-explanatory syntax which would help reduce language learning time.

LUCSED use case detailing was based on the template proposed by the unified software development process OpenUP (Open Unified Process) [22]. Thus, the specification is composed by the following attributes: use case name, brief description, actors, basic and alternate flows, main scenarios, pre-conditions, post-conditions, and special requirements and extension points, as shown in Figure 2. The highlighted words are reserved words in the proposed DSL.

The original template was focused on natural languages, which do not require any standardization and usually are not bound by any restrictions. They also present syntactic phenomena which would culminate in semantic imprecision, making use case process automation more difficult. So, to correctly map detailed use cases into class and sequence

```
Use Case: Use case name.
Brief Description
"Brief Description of the use case.".
System: System name.
Primary and Secondary Actors
Primary Actors: Actor name 01,..., actor name N.
Secondary Actors: Actor name 01, ..., actor name N.

Main Flow: Main flow name.
Actor starts Use Case.
.....
Actor finishes Use Case.

Alternate Flows
Alternate Flow 01: Alternate flow name.
.....
Alternate Flow N: Alternate flow name.
```

Fig. 2: LUCSED template

diagrams, it was necessary to adapt the base template to accommodate necessary and sufficient DSL information required to automate the generation of software artifacts supported by LUCSED.

Some of the necessary adaptations were: making class attributes and methods explicit; in addition to primary and secondary actors, it is necessary to inform the name of the system actors will interact with; for sequence diagrams, it is also necessary to state the combined decision, repetition and concurrency fragments (*if, loop* and *concurrency* respectively), according to the DSL syntax.

LUCSED has mechanisms to describe several situations in which an actor interacts with a system in order to perform a certain action. The standardization proposed by the language makes it easier to map information present in a use case detailing into the appropriate classes, such as relationships, attributes and methods.

The main LUCSED grammar rules in *Extended Backus-Naur Form* (EBNF) notation are presented and explained below, organized in 5 parts.

In the first part of the grammar we have the rules that compose the main structure (skeleton) of use case detailing.

- LUCSED ::= UseCaseHeader UseCaseFlows UseCaseFooter
- UseCaseHeader ::= UseCaseName UseCaseBriefDescription SystemName PrimarySecondaryActors
- UseCaseFlows ::= MainFlowName MainFlowScope [AlternateFlows]
- UseCaseFooter ::= Key Scenario {KeyScenario} PreConditions PosConditions SpecialRequirements ExtensionPoints

In the second part, we have rules responsible for generating specification headers. We can highlight rules *UseCaseName, SystemName, PrimaryActorName* and *SecondaryActorName*, since the name of the control class, system name, and primary and secondary actor names are respectively mapped through these rules.

- UseCaseName ::= “Use Case: ” ControlClassID POINT
- UseCaseBriefDescription ::= “Brief Description” [Text] POINT
- SystemName ::= “System: ” SystemID POINT
- PrimarySecondaryActors ::= “Primary and Secondary Actors ” PrimaryActorName [SecondaryActorName]
- PrimaryActorName ::= “Primary Actors: ” ActorID {“,” ActorID} POINT
- SecondaryActorName ::= “Secondary Actors: ” ActorID { “,” ActorID} POINT

The most important LUCSED rules, the rules that make up the main flow of the use case specification, are presented below.

- MainFlowName ::= “Main Flow: ” MethodControlClassID POINT
- MainFlowScope ::= [ActorID “ starts Use Case” POINT] MainFlow [ActorID “ finishes Use Case” POINT]
- MainFlow ::= MainFlowElements {MainFlowElements}
- MainFlowElements ::= MainFlowCore | FlowIf | FlowLoop | FlowConcurrency
- FlowIf ::= “If ” Condition MainFlow [“Else ” MainFlow]“EndIf”
- FlowLoop ::= “Loop ” Condition MainFlow “EndLoop”
- FlowConcurrency ::= “StartConcurrency ” MainFlow “concurrent” MainFlow “EndConcurrency”
- MainFlowCore ::= ((ActorID | SystemID) (MainFlowTabTransVerb | TABINTRANSVERB) POINT) | ReturnMessage
- MainFlowTabTransVerb ::= TABTRANSVERB [“MethodBoundaryID”] (MainFlowAttibutes | “on” MainFlowBoundaryClass)
- MainFlowAttibutes ::= [[“the”] TABNOUN [“(” [AttributeTypeID] AttributeID {, [AttributeTypeID] AttributeID}“)”]] ((“on” MainFlowBoundaryClass | (“of”) MainFlowEntityClass)) | MainFlowsActorClass)
- MainFlowBoundaryClass ::= BoundaryClassID
- MainFlowActorClass ::= ("for" | "to") ["the"] ActorID
- MainFlowEntityClass ::= EntityClassID [“on” MainFlowBoundaryClass | “by” MainFlowCommunication | MainFlowsActorClass)]
- MainFlowCommunication ::= CommunicationID
- ReturnMessage ::= (SystemID TABTRANSVERB SimpleReturnMessage (“to” | “for”) ["the"] ActorID | SystemID)

The part of the grammar that contemplates alternate flows is quite simple, because its main rule (*AlternateFlowCore*) derives to the most important main flow rule (*MainFlow*). This means that the same specification pattern defined for the basic flow is also accepted for alternate flows.

- AlternateFlows ::= “Alternate Flows ” AlternateFlowScope { AlternateFlowScope}
- AlternateFlowScope ::= “Alternate Flow ” NUM “: ” MethodControlClassID POINT AlternateFlowCore
- AlternateFlowCore ::= MainFlow

Through these rules, it is possible to map boundary and entity classes, their attributes and methods. In order for class attributes and their types to be mapped, the analyst must explicitly state them in the specification, otherwise they will not appear on the diagram. The rules that represent class attributes and their types are *AttributeID* and *AttributeTypeID*

respectively. The name of the control class is generated from the use case name, through rule *ControlClassID*. Conversely, the names of boundary and entity classes are generated through rules *BoundaryClassID* and *EntityClassID* respectively. Control class methods are generated from the names of each flow (main and alternate), through rule *MethodControlClassID*. Boundary class methods are identified by rule *MethodBoundaryID*.

A key point in the process is the ability to identify a certain class and its type. Note that in addition to the adaptations made to the base template and the aforementioned rules, we defined another set of rules based on the prepositions that precede the names of each class in the sentence. Such rules are specified in Table 1.

Table 1: Rules with prepositions

Previous Preposition	Type Class
Of	Entity
On	Boundary
To/For	Actor
By	Communication

In addition to mapping classes and their attributes and methods, LUCSEDTool also stores action execution flows and consequentially messages exchanged between objects, thus enabling the automatic generation of sequence diagrams. Rules *FlowIf, FlowLoop, FlowConcurrency and Condition*, defined in the grammar, were specifically created to map combined fragments present in these diagrams, as per Figure 7.

To identify interactions between classes in the sequence diagram, we defined a syntactic structure for each sentence pattern supported by LUCSED. Thus, it is possible to identify who is the sender, who is the receiver, and what is the message to be sent. The main rules are defined in table 2.

The sentences contained in the specifications written in the LUCSED language must conform to the patterns defined in the aforementioned rules, so the parser can correctly map interactions for the automatic generation of sequence diagrams without compromising quality.

Following the same order as the rules presented in Table 2, we provide a sample sentence conforming to each of these patterns:

1) ***Patient*** *tells the attributes to the* ***Clerk***.
2) ***Clerk*** *selects "MaintainPatient" on* ***MainForm***.
3) ***Clerk*** *enters attributes (...) of* ***Patient***.
4) ***System*** *returns "Input mode screen" to* ***Clerk***..
5) ***System*** *sends the notification by* ***e-mail***.
6) ***System searches*** *for the Patient*.
7) ***System retrieves*** *the attributes of Patient*.
8) ***System saves*** *the attributes of Patient*.
9) ***System validates*** *attributes of Patient*.
10) ***System verifies*** *the attribute (condition) of the Patient*.
11) ***System displays*** *the attributes of Patient on MainForm*.

Table 2: Class interaction rules

Nº	Syntactic Structure	Sender	Receiver	Operation
1	**ActorS** Verb Noun Preposition **ActorR**	ActorS	ActorR	Verb+Noun
2	**Actor** Verb Noun/Method Prep. **Boundary**	Actor	Boundary	Verb+Noun/Method
3	**Actor** Verb Noun Preposition **Entity**	Actor	Boundary	Verb+Noun+Prep.+Entity
4	**System** Verb Noun/Message Prep. **Actor**	Controller	Boundary	Verb+Noun/Message
5	**System** Verb Noun Preposition **Communication**	Controller	Controller	Verb+Noun+Prep.+Commun.
6	**System VerbEntityWithReturn** Noun Prep. Entity	Last Receiver Class (LRC)	Entity	Verb+Noun+Prep.+Entity
7	**System VerbEntityWithReturn** Noun Prep. Entity	Entity	Controller	Return+Noun+Prep.+Entity
8	**System VerbEntityWithoutReturn** Noun Prep. Entity	Controller	Entity	Return+Noun+Prep.+Entity
9	**System VerbValidation** Noun Prep. Entity	LRC	LRC	VerbV.+Noun+Prep.+Entity
10	**System** VerbProcessing Noun Prep. **Entity**	Controller	Controller	VerbP.+Noun+Prep.+Entity
11	**System VerbReturnInBoundary** Noun Prep. Boundary	Controller	Boundary	VerbReturn+Noun

Rules 6 and 7 are identical, therefore we have a type of pattern that generates two operations, namely having two different behaviors. For example: When the user searches for something (verb *searches*), the message must be relayed from the *Boundary* class to the *Entity* class. The same thing happens when the system retrieves the data being searched: a message is sent from the *Entity* class and is replicated until it gets to the *Controller* class. In some cases, messages originate from Controller classes, which is why we defined the Last Receiver Class (LRC) time, because during the flow it is necessary to control in which class a method was last invoked. Such a situation can be seen in the sequence diagram presented in Figure 8.

Still pertaining to the generation of sequence diagrams, it is important to highlight that the core of the parser (LUCSEDTool) contains two fundamental methods, namely *identifySentence()* and *identifyMessage()*. The first method analyzes each word in a sentence and assigns it a grammatical class, such as verb, preposition, noun or other terms in the specification. After classifying the words in all sentences, the *identifyMessage()* method is called, identifying message description, and origin and destination class of a message, based on the grammatical class of words from the sentence passed as parameter.

Some verbs are rule exceptions due to needing special treatment going beyond a direct message between two sentence elements, that is, these verbs behave differently than the others. Verbs such as *validates* and *verifies*, which validate input restrictions or business rules having a *message*

to self behavior in the sequence diagram are treated by a class called *ClassVerbsValidation*. Other verbs that signal a return to the control class, like *searches* and *retrieves* are identified by class *ClassVerbsEntityReturn*.

In an effort to achieve simplicity and more user interactivity, we created three special terms that reference sets of words previously registered in LUCSEDTool. Term *TABTRANSVERB* refers to a set of transitive verbs and term *TABINTRANSVERB* refers to a set of intransitive verbs. These terms are implemented in LUCSEDTool in the form of tables, with a list of pre-registered verbs that are used to semantically verify sentences used in use case detailing, according to rules presented in Table 2. This verification is done through validating some grammatical rules, for example, some verbs only make sense in the context of a phrase if they are related to an actor. Conversely, other verbs only make sense if they are related to the system.

Nouns included in the LUCSED dictionary are also extremely important. We therefore created the term *TABNOUN*, representing a list of nouns pre-registered in LUCSEDTool, in order to increase the possible number of sentences users can write in the proposed DSL. The list can easily be edited using the tool, just like pre-existing verbs in the transitive and intransitive verb tables.

Finally, we have the rules responsible for specification footers.

- KeyScenario ::= “Key Scenario ” NUM “: ” MethodControlClassID POINT
- PreConditions ::= “Pre Conditions ” [Text] POINT
- PosConditions ::= “Post Conditions ” [Text] POINT
- SpecialRequirements ::= “Special Requirements ” [Text] POINT
- ExtensionPoints ::= “Extension Points ” [Text] POINT

4. Results

In this section, we present some of the results we obtained when using LUCSEDTool in tests carried out in software from different fields, such as e-commerce, cellulose, mining, school management and medical software. We gathered documentation developed by specialists from several companies, which had detailed use cases and UML diagrams generated from them. The original specifications had to undergo some changes to adequately satisfy LUCSED grammar rules.

The tests presented here were done in the software system of a medical company which has subsidiaries in several cities. The use case presented as a result is *CRUDPatient*, which has the clerk as primary actor. For this requirement, the cited actor has the privileges of inserting a new client, editing existing information, disabling a client’s subscription, and searching for existing clients. To have access to these features, users must be logged into the system, as stated in the pre-conditions of the use case detailing.

The following is the header in LUCSED language for the use case *CRUDPatient*.

```
Use Case: CRUDPatient.
Brief Description
"This use case allows a clerk enter, edit, disable,
and search for a clinic patient.".
System: System.
Primary and Secondary Actors
Primary Actors: Clerk.
Secondary Actors: Patient.
```

Fig. 3: Header - UC CRUDPatient

The *AddPatient* main flow of the use case is shown in Figure 4. This flow occurs when there are no detours in the basic flow, because if there are any detours, alternate flows or exceptions will be executed. The main flow describes the

```
Main Flow: AddPatient.
Clerk starts Use Case.
    Clerk selects "MaintainPatient" on MainForm.
    Clerk selects "AddPatient" on MainForm.
    System returns "Input mode screen" to Clerk.
    Loop ["Required fields empty"]
        Clerk enters attributes (Int id, String name, String
            socialSNumber, String gender, Date
            birthDate, String homePhone, String
            cellPhone, String email, String
            status ) of Patient.
        System validates attributes of Patient.
    EndLoop

    System searches for the Patient.

    If ["Does not exist"]
        System saves attributes of Patient.
        System returns "Successfully saved" to Clerk.
    Else
        System displays the attributes of
        Patient on MainForm.
    EndIf
Clerk finishes Use Case.
```

Fig. 4: Main Flow - UC CRUDPatient

process of registering a new client in the system, where the user selects the desired option and the system shows the client registration screen. After getting patient data as input, the system will validate the required fields until all of them are filled. Then, it will check the database for duplicate records and notify the user. Finally, it will save the record and send the user a positive feedback.

As mentioned in the previous section, the Alternate flow structure is similar to the main flow. The specification of alternate flows *ModifyPatient*, *DisablePatient* and *ReadPatient* is presented below, as seen in Figure 5. The clerk can modify, disable and search for patients registered in the clinic.

Based on the presented specification, we generated the class diagram, containing the relationships between classes and their attributes and methods, as seen in Figure 6. In the diagram, we identified a boundary class (*MainForm*), a control class (*CRUDPatient*) and an entity class (*Patient*). The name of each class, as well as the names of attributes and methods, depends directly on what the analyst specification

```
Alternate Flows
Alternate Flow 01: ModifyPatient.
    Clerk selects "ModifyPatient" on MainForm.
    Clerk types the attributes (id) of Patient on MainForm.
    System retrieves the attributes of Patient.
    System displays the attributes of Patient on MainForm.
    Loop ["Required fields empty"]
        Clerk modifies the attributes (name, socialSNumber,
        gender, birthDate, homePhone, cellPhone, email,
        status ) of Patient on MainForm.
        System validates attributes of Patient.
    EndLoop
    Clerk selects "SavePatient" on MainForm.
    System updates the informations of Patient.
    System returns "Successfully modified" to Clerk.

Alternate Flow 02: DisablePatient.
    Clerk selects "DisablePatient" on MainForm.
    Clerk types the attributes (id) of Patient on MainForm.
    System retrieves the attributes of Patient.
    System displays the attributes of Patient on MainForm.
    Clerk selects "ConfirmAction" on MainForm.
    If ["There is inconsistency"]
        System returns "There is inconsistency" to Clerk.
    Else
        System returns "Successfully disabled" to Clerk.
    EndIf

Alternate Flow 03: ReadPatient.
    Clerk selects "ReadPatient" on MainForm.
    Clerk types the attributes (id, name) of Patient on
    MainForm.
    System retrieves attributes of Patient.
    If ["Does not exist patient"]
        System returns "Does not exist patient" to Clerk.
    Else
        System displays the attributes of
        Patient on MainForm.
    EndIf
```

Fig. 5: Alternate Flow - UC CRUDPatient

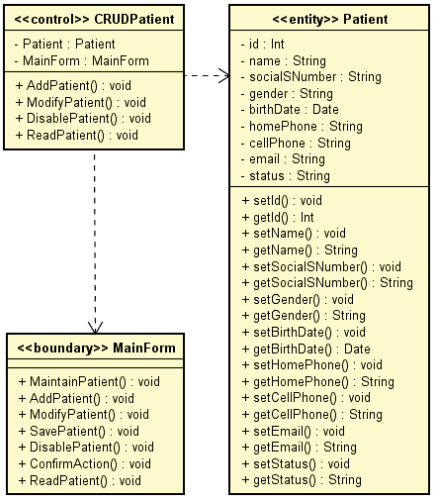


Fig. 6: Class Diagram UC CRUDPatient

was. Note that when comparing the presented specification and diagrams, the names defined in both specification and diagrams are the same.

The first sequence diagram presented pertains to the main flow *AddPatient*, as seen in Figure 7. We can observe that this diagram contains the main features present in a sequence diagram, such as calls to self, return messages and combined fragments.

The diagram shown in Figure 8 refers to the alternate flow

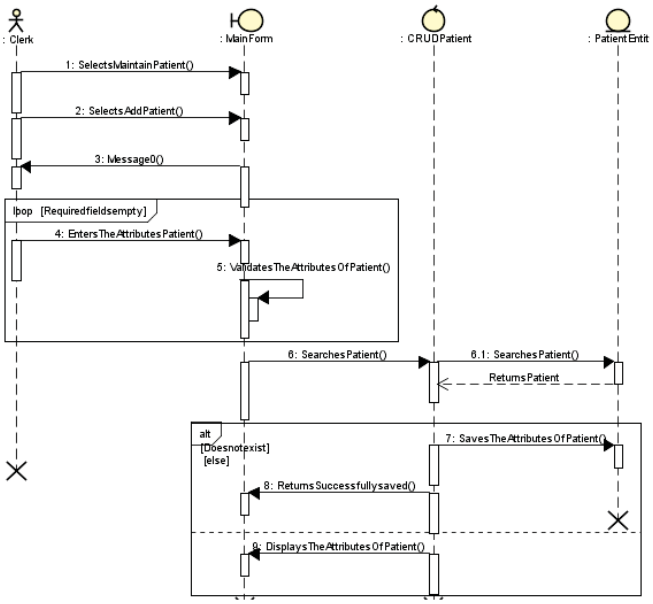


Fig. 7: Sequence Diagram AddPatient

DisablePatient. It is a simpler diagram representing the flow of disabling a patient in the company's system. The flow basically consists of the user providing the system with the identification of the patient to be disabled, the system searching for the data, checking if there are any pendencies and providing adequate feedback to each situation.

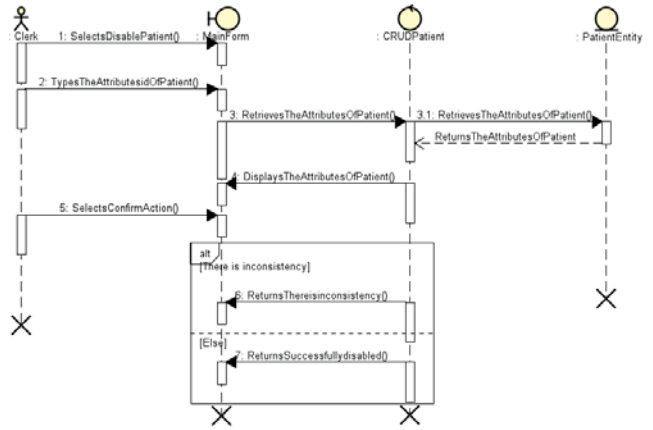


Fig. 8: Sequence Diagram DisablePatient

We deemed the results produced by LUCSEDTool to be satisfactory, because they came very close to what was produced by specialists. In some cases, the diagrams automatically generated by the tool were richer in detail than the original documentation, making it evident that diagrams created by specialists do not always contemplate everything that is in the specification, or the other way around.

5. Conclusions

The LUCSED DSL, along with LUCSEDTool, has features that contribute to the standardization of important stages in software development, such as the requirement specification and use case modeling stages. Standardization would open the doors for clear and precise communication among the entire team, mitigating inherent natural language problems such as ambiguity, uncertainty and complexity. The tool is also able to automatically generate software artifacts such as use case, class and sequence diagrams. These artifacts can be used to automatically generate source code. Our solution offers additional benefits like: improved previously developed artifact reusability; increase in quality and productivity; more control over each stage of the process; mitigation of several errors that can arise from the lack of standards; and offering a common language to analysts and users.

One of the challenges in using such as resource is the effort required by the whole team to learn the language, since every DSL has syntactic and semantic rules that may require some training to master. However, the initial efforts are paid off once the entire team has a good command of the chosen language and tool [5], [21]. We also conclude that, when designing a DSL, we must always have users and their needs in mind, and it is crucial to define simple grammar rules, be guided by relevant requirements and strive to achieve premises such as simplicity and objectivity, to offer a clear, self-explanatory syntax culminating in a more gentle learning curve and less ignored features.

Our tests led us to find some drawbacks in our tool, which will consequently inspire future work, such as improving DSL and LUCSED features and incorporating new features: improving usability ("folding", "autocomplete", "syntax highlighting" and "outline"); implementing version control for generated artifacts; expanding the DSL to support additional languages other than English; include support to generate additional UML diagrams; generate screen prototypes; and allow users to import specifications generated by other tools.

To improve automatic consistency between a UML use case model and its corresponding textual specification set, textual representations of use case relationships present in the UML diagram must be created from effective identifications, which is not a trivial task. Enforcing coherence between a UML model and textual descriptions requires a certain degree of formality in specifications. Conversely, it became clear to us that there are many benefits to formalize use case specifications, even considering the inevitable learning curve [14].

Acknowledgment

The authors acknowledge the financial support received from FAPEMIG, PUCMinas and Centro Universitário UNA, Brazil.

References

[1] C. Alistair, *Writing Effective Use Cases*. Addison-Wesley, 2001.
[2] P. Jayaraman and J. Whittle, "Ucsim: A tool for simulating use case scenarios," in *Software Engineering - Companion, 2007. ICSE 2007 Companion. 29th International Conference on*, May 2007, pp. 43–44.
[3] S. Tiwari and A. Gupta, "A systematic literature review of use case specifications research," *Information and Software Technology*, vol. 67, pp. 128 – 158, 2015.
[4] D. Savic, I. Antovic, S. Vlajic, V. Stanojevic, and M. Milic, "Language for use case specification," in *Software Engineering Workshop (SEW), 2011 34th IEEE*, June 2011, pp. 19–26.
[5] M. Fowler and R. Parsons, *DSL - Linguagens Específicas de Domínio*. Porto Alegre: Bookman, 2013.
[6] M. G. Georgiades and A. S. Andreou, "Formalizing and automating use case model development," *The Open Software Engineering Journal*, vol. 6, pp. 21–40, 2012.
[7] T. Yue, L. C. Briand, and Y. Labiche, "Facilitating the transition from use case models to analysis models: Approach and experiments," *ACM Trans. Softw. Eng. Methodol.*, vol. 22, no. 1, pp. 5:1–5:38, Mar. 2013.
[8] P. J. Clemente, J. M. Conejero, J. Hernández, and L. Sánchez, "Haais-dsl: Dsl to develop home automation and ambient intelligence systems," in *Proceedings of the Second Workshop on Isolation and Integration in Embedded Systems*, ser. IIES '09. New York, NY, USA: ACM, 2009, pp. 13–18.
[9] D. Ghosh, *DSLs in action*. Manning Publications Co., 2010.
[10] M. Freudenthal, "Using dsls for developing enterprise systems," in *Proceedings of the Tenth Workshop on Language Descriptions, Tools and Applications*, ser. LDTA '10. New York, NY, USA: ACM, 2010, pp. 11:1–11:7.
[11] ASTAH, "Astah professional," Agosto 2015, acesso em: 10 ago. 2015. [Online]. Available: http://astah.net/editions/professional
[12] L. Li, "Translating use cases to sequence diagrams," *2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)*, vol. 0, p. 293, 2000.
[13] C. Williams, M. Kaplan, T. Klinger, and A. M. Paradkar, "Toward engineered, useful use cases." *Journal of Object Technology*, vol. 4, no. 6, pp. 45–57, 2005.
[14] V. Hoffmann, H. Lichter, A. Nyßen, and A. Walter, "Towards the integration of uml-and textual use case modeling." *Journal of Object Technology*, vol. 8, no. 3, pp. 85–100, 2009.
[15] V. Šimko, P. Hnětynka, and T. Bureš, "From textual use-cases to component-based applications," in *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing 2010*, ser. Studies in Computational Intelligence, R. Lee, J. Ma, L. Bacon, W. Du, and M. Petridis, Eds. Springer Berlin Heidelberg, 2010, vol. 295, pp. 23–37.
[16] J. S. Thakur and A. Gupta, "Automatic generation of sequence diagram from use case specification," in *Proceedings of the 7th India Software Engineering Conference*, ser. ISEC '14. New York, NY, USA: ACM, 2014, pp. 20:1–20:6.
[17] S. N. Group, "Stanford parser," 2015, acesso em: 03 ago. 2015. [Online]. Available: http://nlp.stanford.edu/
[18] T. Yue, L. C. Briand, and Y. Labiche, "atoucan: An automated framework to derive uml analysis models from use case models," *ACM Trans. Softw. Eng. Methodol.*, vol. 24, no. 3, pp. 13:1–13:52, May 2015.
[19] W. Heijstek and M. Chaudron, "The impact of model driven development on the software architecture process," in *Software Engineering and Advanced Applications (SEAA), 2010 36th EUROMICRO Conference on*, Sept 2010, pp. 333–341.
[20] D. Ghosh, "Dsl for the uninitiated," *Commun. ACM*, vol. 54, no. 7, pp. 44–50, July 2011.
[21] G. Gupta, "Language-based software engineering," *Science of Computer Programming*, vol. 97, Part 1, pp. 37 – 40, 2015, special Issue on New Ideas and Emerging Results in Understanding Software.
[22] ECLIPSE, "Openup," 2015, acesso em: 02 ago. 2015. [Online]. Available: http://epf.eclipse.org/wikis/openup/index.htm

Proposal of the Killing Time Smartphone Apps for Optimize the Lifestyle of Smartphone Users

Ziran Fan [1], Takayuki Fujimoto[2]
[1]Graduate School of Information Sciences and Arts, Toyo University, Kawagoe, Saitama, Japan
[2] Graduate School of Information Sciences and Arts, Toyo University, Kawagoe, Saitama, Japan

Abstract – *In this study, we focused the needs that the smartphone applications to kill time are most needed by smartphone users. At this point, there is no objective definition of the killing time applications and the conditions of the killing time applications have not been disclosed either. Through this study, we made a survey about the smartphone usage environment of society and the life styles of smartphone users. The requirements of the killing time smartphone applications are established through the consoderation. Then, we show the objective definition of the killing time applications and the application prototypes are proposed based on those results.*

Keywords: Smartphone Apps, Killing Time, Smartphone Users' Needs, Smartphone Users' Lifestyle, Smartphone Usage Environment, Application Prototypes.

1 Introduction

Smartphone has been spreading at high speed recently. We can see people using smartphone in one hand at everywhere, for example, in the subway, the bus stations and the Starbucks. Those scenes are certainly becoming popular. With the functions of smartphone such as telephone call, E-mail, surfing the Internet, game and SNS really have been making our life more convenient.

For the most of smartphone users, the purpose of the smartphone use is to spend free time in life. For example, the time you are waiting for a date, boredom when you are taking bus, rest between works. Because smartphones are more portable than wallet or ID card and could be used at all times and all places freely.

Regarding the needs of the smartphone users, many killing time applications, which take advantage of smartphone features, have been released. The popular applications we are using have some of the elements of that needs. Most of the case, they are game applications. Many developers are planning to make a hot application as killing time applications, however, there is still no objective definition of the killing time applications. The idea of the killing time applications is based on the subjective sensation of the users' experience. Of course, if the usage environment changed, the sensation about the killing time applications also would be changed too.

Even though the killing time applications are the biggest needs from smartphone users, the objective definition and the conditions have not been determined. We could not judge what the killing time application was.

The points of this study are survey, analysis and proposal. First, we made the survey to elucidate the usage environment in Japan. Then we analyzed the lifestyles of smartphone users: what they are using and what they want to use. Finally, we proposed the objective definition of the killing time applications and tried showing it with a prototype application in the optimized way for users' needs.

2 Background

Today, the number of the smartphones use is high in Japan. People who have smartphones as their first mobile phones is increasing mainly among the middle-school and high-school students. From the consumer confidence survey of cabinet office in March, 2015, the rate of diffusion with phone is 94.4% and the rate of diffusion with smartphone is 60.6%. From 2015 Information and Communications Use Trend Survey of Ministry of Internal Affairs and Communications, the rate from the age group showed that the rate of diffusion of twenties is 83.7% the highest and thirties is 72.1% the second in a series. The rate will also raise at the base of old people and young generation in the future. It is expected that the base of the user in Japan will be expanded with the spread of smartphones.

3 Motivation

The lifestyles of the smartphone users has been becoming complicated by the spread of smartphone, the change of the usage environment and advanced technology for the development of smartphone and applications.

From a survey of JustSystem Co.,Ltd which was taken with 960 smartphone users whose age was between 10 and 70, game applications are used most often in a day and the average time is 63.8 minutes. Perusal of Internet contents and social communication applications such as Facebook and Twitter are also often used and the average time in a day is 57.1 minutes. Time of use tends to increase at young generation clearly.

Smartphones are different from the phones based on the call function. The number of the usage is as many as the number of the applications you installed. According to a survey about the usage of smartphones, spending free time: watching some interesting videos, net surfing, playing game

applications and checking the shopping net site, is the most common and the rate on the whole is 18.5%.

With those survey results, it is clear that spending free time (killing time) is the most common purpose for the use at the lifestyles of smartphone users today. So, there is a question: what kind of free time do users have and spend? The survey shows the users tend to use the smartphones to kill time on a train.

From survey results we can see that smartphone users use smartphones to spend their free time most commonly, and a time when getting on a train is specifically boredom to the users. It was usual for people to read newspapers or magazines to spend free time on a train, but with the significant changes of the information usage environment from the spread of smartphone, people use the smartphones instead of books to spend free time on a train.

This study focuses on spending boring time on the train, which is the biggest needs from the users and proposes a smartphone application to meet that demand.

4 Purpose

4.1 Meaning of Killing Time

Killing time is defined as taking acts and works which are not requested to spend time when one has spare time. Definition for spare time is different from the respective conditions and the standard to make a judgment of spare time is not absolute. So it is difficult to take an objective definition.

4.2 Smartphone Usage on A Train

In the previous chapter, the results show that smartphone users tend to use the smartphones to spend their spare time on a train. We assume that two main elements are important.

One is the social environment. Taking train is the major mean of transportation in Japan. The Japanese people use the train to go to the school or work more frequently than other transportations such as the buses or the cars. Many people decide their work or school on the basis of the range in which they can commute by train. Since the train is always crowded in the rush hours, smartphone's smallness is suitable to use with one a hand in the limited place.

The other is today's our lifestyles. Nobody would get on a train every day without a destination, except those who have a hobby to enjoy to get on a train for nothing. Taking train is a mean of transportation and people have a certain purpose to arrive at somewhere they have to go. So we could declare that they are actually free in a time during which they have to wait until they arrive at the destination. Unless they meet an accident, the thing that they should do is just waiting. In fact, a time when we are getting on a train is the biggest spare time in our life, and we always have to spend that spare time with something on purpose on the train.

In that kind of free time we have to spend, the question is how we spend it. If we pay attention to the passengers on the train, we would know that most people use their smartphones. The things we can do with smartphone on the train are not only information collection but also playing the games, chatting with someone on applications, surfing the Internet and schedule management on that small box. From these, it could be said that smartphone is a suitable tool to spend free time on the train.

5 Definition of Free Time

The killing time applications have a great market value because of the needs from the users. But there is no objective definition as long as we investigated. This study is based on the survey for the smartphones' usage environment in Japan and the lifestyles of the users to propose the definition on the killing time applications.

As we have described, the meaning of free time is different from the respective condition which people have. However it is common for everyone that one is free for transit time on the train until he/she gets to the destination. Because the reason why we all have to take train is just to transfer to the destinations. Using the time effectively to do something such as remembering a few English words is thought to be a sensible way of spending free time on the train when you go to the office or the school. The time on a moving train is a free time for most Japanese people and it is inevitable. So this study focus on transit time on the train to plan the killing time application.

The sensibility of People to feel a time as free time is relative, but a transit time is commonly definite. For example, if the time for which a person gets on the train is 5 minutes, free time he/she could feel is invariably limited to a range of 5 minutes. Most of the case, he/she has some purposes or tasks after getting off a train.

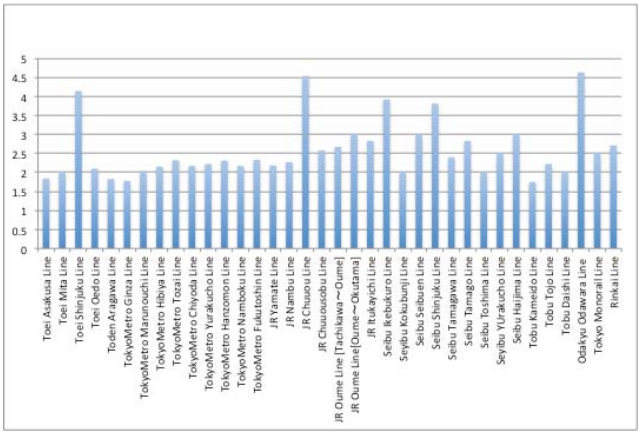
We collected the statistics for the survey to research the average transit time from one station to the next station of 36 main train lines in Tokyo, and show the result in Tab1 and Graph2. The result shows that the indicated time is free for the passengers. The average transit time from one station to the next station is about 2 minutes and 27 seconds.

The users would not always get on a train for one station, but the time we got, 2 minutes 27 seconds, is a standard value for a time during which the users feel free. If the user get on a train for three stations, by just adding 2 minutes 27 seconds three times over, we can get the average value of the user's free time.

Tab1. Average moving time of lines (1 station) in Tokyo

Line	Station	Time
Toei Asakusa Line	20	1.84
Toei Mita Line	27	2.00
Toei Shinjuku Line	8	4.14
Toei Oedo Line	11	2.10
Toden Aragawa Line	30	1.83
TokyoMetro Ginza Line	19	1.78
TokyoMetro Marunouchi Line	25	2.04
TokyoMetro Hibiya Line	21	2.15
TokyoMetro Tozai Line	23	2.32
TokyoMetro Chiyoda Line	19	2.17
TokyoMetro Yurakucho Line	24	2.22
TokyoMetro Hanzomon Line	14	2.31
TokyoMetro Namboku Line	19	2.17
TokyoMetro Fukutoshin Line	16	2.33
JR Yamanote Line	29	2.18
JR Nambu Line	26	2.27
JR Chuuou Line	14	4.54
JR Chuuousobu Line	20	2.58
JR Oume Line [Tachikawa～Oume]	13	2.67
JR Oume Line[Oume～Okutama]	25	3.00
JR Itukayichi Line	7	2.83
Seibu Ikebukuro Line	13	3.92
Seibu Kokubunji Line	5	2.00
Seibu Seibuen Line	2	3.00
Seibu Shinjuku Line	17	3.81
Seibu Tamagawa Line	6	2.40
Seibu Tamago Line	7	2.83
Seibu Toshima Line	2	2.00
Seibu Yurakucho Line	3	2.50
Seibu Haijima Line	8	3.00
Tobu Kameido Line	5	1.75
Tobu Tojo Line	24	2.22
Tobu Daishi Line	2	2.00
Odakyu Odawara Line	20	4.63
Tokyo Monorail Line	11	2.50
Rinkai Line	8	2.71

Figure2. Average moving time of lines (1 station) in Tokyo



This study assumes that there are two ways to achieve killing time in transit time on the train regarding the biggest needs of the users. One is the usual way such as reading newspapers or magazines, which we have been doing. Relatively, using the Internet for reading news sites is also a common way to spend free time on the train. The other is the way to spend free time on the train by playing the games or enjoying any kind of the entertainment applications, it is coming popular with the spread of smartphone.

Regarding the former way, as there are already many news sites on the Internet, we did not treat it in this study. This study tests the latter way: spending free time on the train unwittingly by the game, which enables the users to get a kind of fulfillment. It seems that the killing time method with fulfillment to challenge the user's gaming mind is the biggest needs because the rate of diffusion about smartphone is the highest at twenties who is familiar with the game applications and other entertainment applications through their everyday life.

6 Application Implementation

6.1 Apps Summary

The significant feature of killing time applications must be simple. Everyone can easily use it everywhere. The simplicity does not mean uninteresting. We can spend free time by just gazing steadily at the screen of smartphone. But the users would not be satisfied with that and it cannot be expected that the users continuously use it. On the other side, it does not mean that application must be interesting unlimitedly. It is necessary to project more functions to make the applications more interesting. As a result, the users may spend the time with an addictive joy instead of just spending free time. It is confuse of natural order of our project if the users spend more time on playing to enjoy the game with addiction rather than using it to kill time. So it is very important to secure the effect of killing time in a proper way in the limited average transit time of 2 minutes 27 seconds.

The application proposed in this study has the simple structure and operability with which the users can play it with only one finger. And the system sets the play time based on the average train transit time to meet each user's needs to spend free time of the time on the train. The system has been designed to enable the users to feel fulfillment.

6.2 Design Principle

To meet the needs from the smartphone users to spend free time on the train properly, we decided to narrow down to one train line from 36 lines which we had checked. JR Yamanote Line is the most famous train line in Japan, and is also the train route used most frequently. JR Yamanote Line is the biggest line that connects the main area in Tokyo. JR Yamanote Line connects the busiest stations in Japan, such as

Tokyo Station, Shinjuku Station, Shibuya Station and Akihabara Station.

We have designed the application based on a motif as JR Yamanote Line in this study. The application could offer the users better killing time experience with this motif when they use it. And we can also collect more accurate data easier in our study. It is shown in Figure 2 and Figure 3.

It is important to consider how to apply the purpose of killing time to the goal of the game in design principle, because the concept as the application to spend free time is the prerequisite for this application proposed in this study.

First, the play time of the application is set based on the train transit time of the time determined by the departure station and the arrival station. There will be some inevitable errors on the time, but the conditions of the time when the users get on the train are changing. Therefore it will not make any trouble for the study. The application we proposed has a function that occurs game over when the play time passes, it will prevent the users from the overuse which is not supposed to be the use just for spending free time.

The operation of the application is designed simply. The users can play the application with only one finger without any other operation, so everyone could use it intuitively. And also it has no functions which is not just for spending free time. Two elements on the design principle which we have to prevent are getting tired and becoming absorbed. And we have also tried some actions to make the applications more interesting for use: adding more variation of the game design and reflecting the results for the user's each use of the game as a form of score.

The user's experience as just spending free time leads to the long-term use of the application. The point of the users' continuous use is enabling the users to feel satisfaction. We assume that satisfaction is achievement for the game in this study. The most important element of achievement for the game is the relative difficulty of the game. Users would not keep up using if the game is too hard, and on the other hand, if the game is too easy it will make users tired when they play only once. A clear goal (the ending of game) is specially should not exist in the application. The reason of that is to secure the long-term users with the design principle which makes no finish deliberately. We can find that feature in many famous social network game applications. So we had to design the application flexibly to give the users suitable pressure on playing and to make sure that pressure could be accepted by people as much as possible.

But, there is a potential that it will make a wasted feeling with the endless game. The functions to enhance the achievement are important. With these functions, the users could use the application with interest every time when playing the game. We took that mainly as two elements in this study.

One is the user's feeling that their playing skills are being improved. Spending free time is not just the expenditure of time. The results which people get from that action is important. The design principle of the application is based on this idea and focus on the change of the relative difficulty on the game. Users would play easily first and when the paly time is passing, the users can get used to the difficulty of the game. So we have designed the application in which the relative difficulty of the game would be improved as time passes. We strived for providing a more interesting game experience to the users by visualizing the change of the relative difficulty and indicating that process to them.

The other is a reward that the users can get in the game. The feeling of skills improvement is not attractive enough for the users, and if the users cannot get anything as they spend time for the game, playing will be a tired work. The score function is added into the application and its form is designed as the theme of the application and like the trace of users' playing.

We thought those design principles could prevent the users from feeling wasted when using the application, and we have developed the application based on them.

Figure 2. Station-choosing

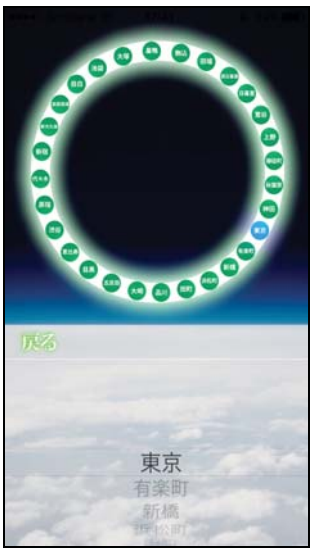


Figure 3. Stations Selection

6.3 Apps Construction

The illustrations of the application which we developed are shown in Figure 4 and Figure 5.

The purpose of the application is a practice for validating the theory which we have proposed. And we will show the explanation of the application at this section.

First, the user should touch the title screen (Figure 4) to open the station-choosing screen (Figure 2). Since this applications is designed on a motif of JR Yamanote Line, the user should choose the station from JR Yamanote Line station list (Figure 3). When both departure station and arrival station are chosen, the user should press the "Start" button to start game. The user should tap the game character (player) on the screen (Figure 5) to play the game.

The image of the train in the left of the screen is the game character which the user will operate. The train will be falling constantly when the user play the game, and it will go the game over when the train touches the white area on the bottom. When the user taps the screen, the train will spring up and the degree of height and intensity will be changed by the speed of the user's tap. If the user taps it too quickly, the train will spring up too high to hit against the top on the screen, and the white area on the top is same as the bottom, which will make the game over when train touches it.

To make the layout of the game useful, the time until the arrival of the train: the real train on which the user is getting) is shown in the white area on the bottom and another white area is showing some information about the game to the user when they are playing. And the present score (reset when a game is over) and the total score are also shown on the upper part, so that the user can always catch the process of playing.

The point of the game is that the user should control the game character's springing up between the top and bottom artfully with the apposite power at good timing to advance the game.

Two obstructions will be drifting from the right side to the left side. When the game character touches the obstructions, it will also go game over. And the size of the obstructions will be bigger as the playtime passes. So, the user should control the game character to ward off the obstructions and it will give the user suitable pressure and make the playing amusingly not as a boring work.

Operating the game character to avoid three things: the bottom area, the top area and two obstructions, and keeping the game play is the structure of this game.

Finally, we will take an explanation of the function of score of this game (Figure 4). The score function, which has been designed as the results for the user's play, will raise as user's experience stacks. The score, the numbers is not only the value of the game. Some "new thing" can be opened by the user if he/she is playing consecutively. This function is based on "trophy" system. These functions will be useful to secure the long-term user.

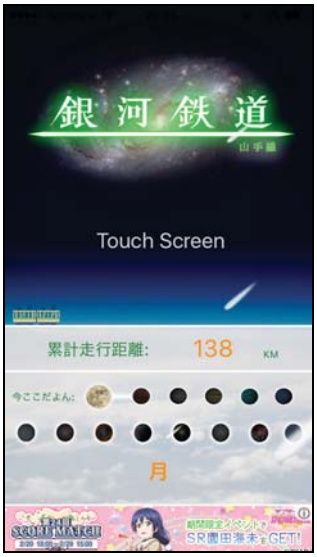


Figure 4. Title Screen

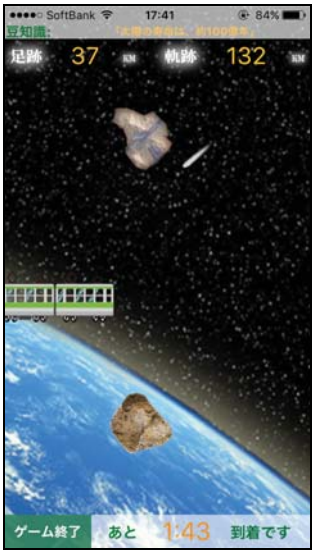


Figure 5. Playing Screen

7 Conclusions

Spending free time with smartphone is the biggest needs from the smartphone users. But there is still no objective definition on killing time applications, and the smartphone applications optimized for the users' also do not exist at present.

We propose the smartphone application which is adapted to the purpose for spending free time with the analysis of the needs from today's smartphone users, based on the survey on the users' lifestyles.

Game applications are the most frequently used smartphone applications by the users and there is the most common trend that smartphone users are usually using smartphones to spend free time when getting on a train.

We have fixed the standard value of the time in which the users use smartphones to spend free time when they get on the train by the statistics about the average transit time of the main train lines in Japan, based on the survey of the smartphone usage environment in Japan. We have tried proposing the objective definition on the killing time application. The smartphone application on the presupposition of time expenditure in which the users could be satisfied with some results from the use for 2 minutes 30 seconds.

We considered the development of the application based on the results of the study for accuracy enhancement. We could propose the killing time application that is optimized for the conditions of the smartphone users' lifestyles.

One of the tasks in the future is to take an actual condition survey for the application. It is necessary to check how effective the killing time application use is for the users' lifestyles. We will make an adjustment and improvement based on the results of the survey.

This study plans to show new possibility to the trade of smartphone applications through the proposal of the killing time applications optimized for the lifestyles of the smartphone users (it is unexampled). It is also necessary to consider what kind of influence this study could have consecutively.

We examined today's smartphone users' life styles in Japan and it was found that the biggest need from the users is spending free time. Smartphone is developing strikingly as the representative of information and communications technology in the world. For this reason, it is meaningful to research the smartphone users' lifestyles internationally, not only for the users in Japan. The application has been devised for the users, who live in Japan in this study. Therefore it is necessary to check if it is efficient in the world range. It is also beneficial for further research to examine how the lifestyles of the smartphone users change under the different social environments, cultural backgrounds and the modes of daily lives.

8 References

[1] Japan Cabinet office. "Consumer Confidence Survey"; 2014. http://www.esri.cao.go.jp/jp/stat/shouhi/shouhi.html

[2] Ministry of Internal Affairs and Communications. "2015 Information and Communications User Trend Survey"; 2015. http://www.soumu.go.jp/johotsusintokei/statistics/

[3] AppMarketingLab. Website, 2014. http://appmarketinglabo.net/appmarket2014/

[4] D2C So.,Ltd. "Multidevice Use Survey"; 2014. http://www.d2c.co.jp/news/2013/07/04/959/

[5] Appstudioz. "Mobile App trends Worldwide"; 2014. http://www.appstudioz.com/blog

[6] JustSystem Co.,Ltd. "Questionnaire about the change of Smartphone and Lifestyle";2014.http://www.justsystems.com

ISBN: 1-60132-446-4, CSREA Press ©

SESSION

LATE BREAKING PAPERS

Chair(s)

TBA

ISBN: 1-60132-446-4, CSREA Press ©

ISBN: 1-60132-446-4, CSREA Press ©

A PROPOSAL FOR AN ADAPTATION TO THE UNIFIED PROCESS FOR THE DEVELOPMENT OF GRP (*GOVERNMENT RESOURCE PLANNING*) SYSTEMS

Mauro Borges França[1], Alexandre Cardoso[2] and Edgard A. Lamounier Jr.[2]
[1]Instituto Federal de Educação, Ciência e Tecnologia do Triângulo Mineiro (IFTM) Uberaba, MG – Brasil
[2]Universidade Federal de Uberlândia (UFU) - Uberlândia, MG – Brasil
{mauro}@iftm.edu.br, {alexandre,edgard}@ufu.br

***Abstract* – To obtain a software process that adequately affords trustworthy development, along with providing good compliance and does not overly demand time for generating artifacts is an excellent response to market demands, as well as adding strong motivation to research related to the theme. One of the great challenges associated with this process is to adapt the proposals from different methodologies for software development to the workplace reality and task force of an Information Technology team (IT). This research study proposes the suitability of a methodology based on the Unified Process, aiming its adaption in particular to the development of software for public office. One of the main motivations behind this proposal is the fact that there does not exist a developmental standard in teams of this kind. Therefore, this research covers the application of a methodology for the development of systems for the construction of an integrated system with the features of GRP (*Government Resource Planning*). In order to evaluate the proposed development model, three projects were selected that commonly hold aspects of complexity and strength.**

***Key Words* -: Software development, Unified Process, agile methodologies, GRP, productivity.**

I. INTRODUCTION

Software Engineering arose from the need to develop, through use of engineering methods, the production of computer programs, which involve processes and time metrics, cost, involvement of people, results and possible advances (Sommerville 2011).

In this context, it is of no small order to say, the development of quality software is a great challenge for IT companies, independent of their size. Besides this, to find the adequate process for such development is a motivating factor behind the study of many a researcher. The work developed by Machado (2000) relates that the appearance of international standards for software processes, such as the ISO/IEC 12207 standard and maturity models (CMM, TRILLIUM, BOOTSTRAP and ISO/IEC 15504) influence organizations to direct their efforts, not only in process definition, but also in establishing mechanisms for their continual improvement. However, this continual development of software, in many cases being an activity heavily dependent on the individual abilities of the developer, leads to many organizations not having defined processes and little knowledge concerning the maturity of such processes.

It is important to highlight that the adoption of developmental methodologies directs software project members towards activities, actions and tasks necessary for the development of high quality software. Faced with the present scenario, it becomes necessary to define the process, which for the IEEE, is a "sequence of steps executed with a determined objective", and for the CMMI, it is "a set of inter-related actions performed to obtain a specific set of products, results or services" (PÁDUA FILHO, 2009).

The research studies of Fuggetta (2000), Pressman (2006) and Osterweil (1987), affirm that the quality of the development process impacts directly upon the quality of the artefacts contemplated for development. Still further, according to Greer and Conradi (2009), the need for a defined process is recognized as a strategy towards reducing risks in project management. The work presented by Bertollo and Falbo (2003), adds, "the main cause of problems in the development of software is the lack of a development process that is clear and effective in its definition". However, it becomes evident that to reduce the problems linked to software development, risk reduction and the delivery of quality software, the adoption of some type of development process is of extreme importance when dealing with projects of this type.

Among the various existing processes, the traditional and agile methods are given prominence. There are those enterprises that believe that the software they produce can be understood simply by reading its source code (agile methods). Other producers document their artefacts in an intensive arrangement (traditional method) (SHACH, 2009).

However, software development centres for public agencies present problems in the adoption of both traditional and agile methods. This is due to inherent distinct features, those of which are commonly found in all other software industries. Highlighted among such are a reduced number of professionals, high staff turnover, overly high bidding process latency, heterogeneous teams, lack of standards, lack of or incomplete documentation.

In this context, there arises the challenge of choosing one methodology that guides development systems and which is followed by its developers. Therefore, allowing agility and complete updating of the whole development process attributed to many public agencies.

Under the pretext of working towards an answer to the problem presented herein, along with proposing feasible solutions for small teams, this work proposes a set of computational and management techniques. In this manner, adapting the methodology behind the Unified Process to the features of the software development sector environment associated with public offices.

II. FUNDAMENTALS

Over the last decades, a number of strategies have been created for software development, in accordance with Rasmsin and Paige (2008), which are most commonly supported upon conventional or agile methodologies. According to Alves Paim *et al*. (2011), the complex nature of software development and the wide variation of already existing methods generate difficult and imprecise comparisons between traditional and agile methodologies. Looking from this perspective, Leffingwell (2006) discusses these differences, which are summarized on Table I.

TABLE I
Characteristics of the paradigm traditional vs agile, according to (Leffingwell, 2006)

Point of view	Traditional	Agile
Measure of success	Conformity with the plan	Response to change, operational code.
Cultural management	Leadership and control	Leadership/ collaboration
Requests and architecture	Initially large	Continuous/ emergent
Guarantee of quality test	Large, planned / late test.	Continuous/ competitor / early test.
Planning and chronogram	Detailed, fixed scope, time and resources limited.	Planning on two levels, fixed date, estimated scope.

In a synthetic manner, one observes that while the traditional paradigm has been recommended for projects of a large scale and high risk, the agile paradigm has shown itself as being more appropriate for low risk projects made up of small teams (Boehm; Turner, 2004; Lindvall; Costa, 2004; Cohen; Nord; Tomayko, 2006; Ramsin; Paige, 2008). In the general sense, large and critical projects can be hindered through the lack of rigor and predictability of the agile paradigm, while small and low risk projects may incur an unnecessarily high cost and inadequate time frame. These occur through the lack of simplicity and flexibility of the traditional paradigm, which generally imposes procedures of a complex nature, along with comprehensive documentation. Given such concepts, the adaptation of methodologies has become a constant factor, in order that they comply with software production processes. It is for this reason that software manufacturers always look to customizing software processes in accordance with their organizational structure.

III. RELATED WORK

A lot of effort has been made in the sense of adapting both traditional and agile methodologies, so that they attend to the specific needs of organizations. The most well-known of these in the software industry is the RUP – Rational Unified Process, which contains the same roots as the Unified Process – UP. However, there exist various other relevant publications that make use of a combination of traditional and agile methodologies. (SCOTT, 2003).

Software development processes always work in accordance with the environment in which they are applied. A number of related studies present adaptations to the processes in conjunction with the applications development domain, taking into consideration the scenarios inherent to such domains, such as size and qualification of the development team, structure of software industry premises that house the team, management experience and the artefacts derived from the process. In the following, studies that focus on the adaptation of these methods is presented.

A. A Proposal for the Agile Development of Virtual Environments

In Mattioli *et al.* (2015), agile development is proposed, where it is applied in environments that develop Virtual Reality Systems – VRS. In order to reach this goal, the authors customized the agile methodology XP to be a process that adds features of prototyping, iterative and the evolutionary development of software projects, in a manner that attends to five key principles in VRS systems: 1) the evolutionary nature of the VRS; 2) the interactivity of the construction and the high fidelity of the models; 3) the need for client feedback; 4) the need for interaction and usability tests; 5) the modularization of the VRS. As a means to apply the VRS development process, six basic activities were used, Planning, Analysis, Project, Codification, Tests and Integration. All these activities are developed at each iteration, as seen in Figure 1.

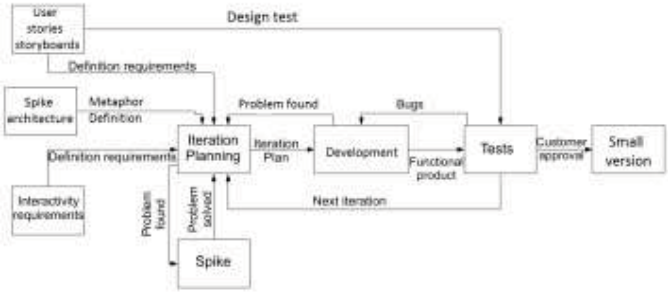


Fig. 1. Full cycle of the VRS agile development [extracted from (MATTIOLI *et al.* 2015)].

The planning for an iteration is made, starting from feedback received from previous iterations, where at each iteration the system models are updated.

One important aspect is the interactivity requirement phase, which occupies a high-ranking position among development processes, due to the fact that in the building of solutions for Virtual Reality Systems, it is primordial to contemplate usability and interactivity into the system. However, to apply this model in public offices for the building of integrated systems, a limiting factor that needs to be taken into account is the time needed to design the interactivity requirement phase, which is unnecessary in light of the features linked to this project.

A. Characterization of a Software Process for Free Software Projects

The work put forward by Souza et. al. (2004), presents a method with the same features as RUP, with the exception that these are realized not only in the scope of the system, but also in each subsystem, where there exists a functional rotation scheme between them (Figure 2). Allowing therefore that after the transition phase (implementation) of the first subsystem(s), the knowledge acquired, along with the artefacts produced in its development are aptly used in the

remaining subsystems. Besides this it permits that improvements be made to the evaluation, concerning the running and estimations involved in the project as a whole, as a sampling of the system has been acquired, at each of its phases.

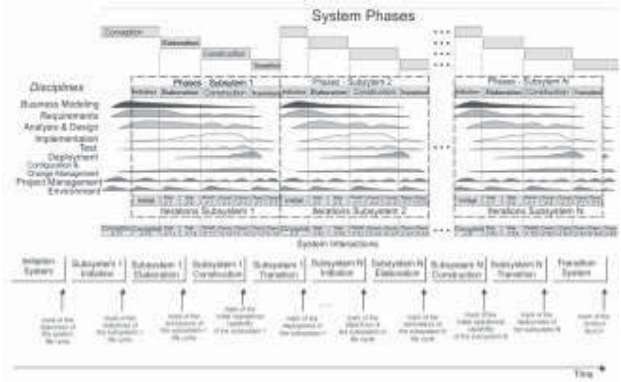


Fig. 2. The RUP development cycle for N subsystems. [Extracted from (SOUZA *et al.,* 2004)]

As this process uses all the RUP phases over all the subsystems, this becomes extremely invariable when applying the features from this case study, which is due to all the subsystems having to go over the four phases of the RUP process again. This entails a significant demand on human resources in terms of assistance across the whole process. To meet these demands, public offices need to significantly increase their staff to make teams.

C. Adaptation of the Rational Unified Process (RUP) in small Distributed Development teams

The work by Rocha et al. (2008) presents a study applied to adapting the Development Process based on RUP, to a small team using Distributed Software Development (DSD).

In order to enable the proposals from this work, its research was integrated into an academic discipline from Software Engineering, with focus given to its implementation in software industries, which use Distributed Software Development (DSD) for realizing real projects. Through the separation of clients and projects, a simulation of a software industry environment was sought, to offer artefacts that were decided in agreement with the client. The company referred to in this study carries the name of TechnoSapiens, it was started by nine students, working in a distributed mode with part of the team active in the same city, where the others were spread among three other municipalities. It is worth mentioning that none of the team members had worked together before, thus working together for the first time at the start of this project.

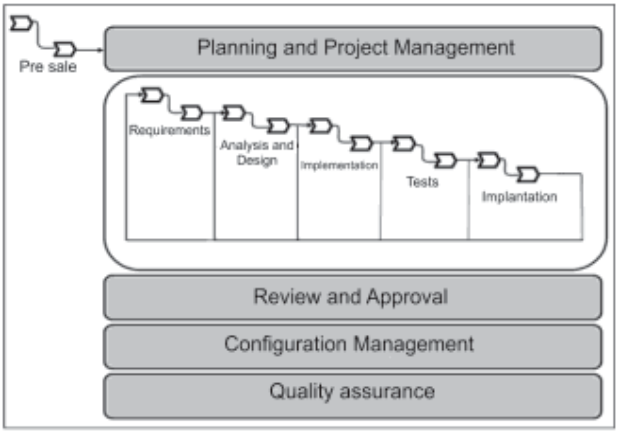


Fig. 3. The adaptation process of RUP for DSD [extracted from (ROCHA et al., 2008)].

Besides the adaptation of the RUP process, this work highlights the development of distributed systems, this characteristic causes various problems when it comes to applying the process proposed herein. These are based on the demand on aspects for the mitigation of risks, which are normally higher due to this characteristic. Therefore, this study is not a viable option for application to the case study related in this research work.

TABLE II
Table with comparisons made between studies

	Agile SRV	RUP – N Sub-Systems	RUP in DDS
Iterations and additions	x	x	x
Client on-site		x	
Prototyping Interfaces	x		
Continual Integration	x	x	x
Simplified design		x	x
Daily meetings	x		

By analyzing Table II, one notes that none of the systems studied present all the necessary criteria as an integrated set. However, it is believed that a methodology that contemplates all these criteria possesses greater potential for influencing the productivity of an IT team.

Therefore, this work in the sense of collaborating in the establishing of a software development methodology, aims at producing a bond between these characteristics, in order that it can attend to public office teams that have software industry characteristics. Details concerning this methodology are presented in the next section.

IV. PROPOSED METHODOLOGY

This work proposes an integration of the Unified Process and the agile method, aiming primarily at customizing this with a considerable reduction in the number of phases and artefacts. With its focus upon obtaining an improved management of the software development process in companies with public office characteristics. The hope therefore is to contribute to the perspective of the integration of hierarchical models, conceptually defined in government corporate projects, such as GRPs.

The methodology proposed herein will be referred to as SDM-GRP, which stands for Systems Development Methodology – Government Resource Planning.

The objective behind SDM-GRP is to direct the public office development teams in the use of the methodology as a reference guide for the building of modules and/or sub-modules, which make up integrated systems compatible with GRPs. It is important to highlight that although the methodology can be inspired upon common public office processes related to education, there does not exist any reason through investigation or otherwise, as to why it should not be used in projects of a different nature.

Besides this, SDM-GRP is organized into papers, artefacts, projects and maintenance, where each project contemplates a sequence of phases and each phase defines activity flows that contemplate the realization of planning meetings, follow-ups and the production of control artefacts.

The SDM-GRP is organized into four phases, inception, elaboration, construction and transition, in accordance with that proposed by the UP. However, the nature of each phase is differentiated in relation to the traditional processes of UP. In fact, at each phase a development adjustment is proposed, which was guided through a common necessity of the development teams in public environments. Such environments contain in their essence; characteristics close to those of software development projects.

The idea proposed herein is related to the fact that at each phase a set of activities are produced, and consequently a set of artefacts for documenting the development process. Therefore, at the end of each phase, one hopes to obtain such artefacts, be they textual or diagrammatic, depending on the phase in question. Following on, Figure 4 presents the structure for the phases of SDM-GRP.

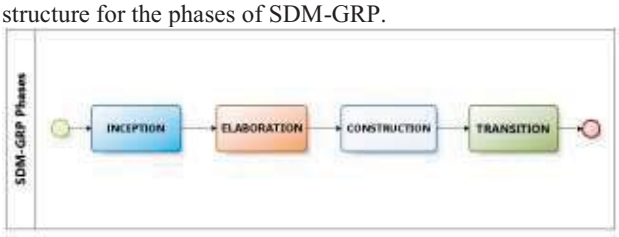


Fig. 4 - SDM-GRP Phases.

A. Inception Phase

The Inception phase has as its underlying objective the formalization of the module for development. This formalization must be duly registered by the client of the module, under the pretext of identifying the demand necessary in order to attend to that area of business. This registration will be realized by means of the Document of Official Demand – DOD, which is an official document within any public institution. After the filling out of the DOD, an analysis will be made between the acting parties of Project Manager and the Sponsor to render the initial view of the project scope, and register into the documentation the Term of Project Initiation – TPI. This is aimed at starting the Project Plan that will envisage the planning over the execution of every production process in the module.

By means of these three documents, the Project is initiated, together these should provide conditions necessary for the building of the module ready for development. Presented in Figure 5 is a clear vision of the Inception phase.

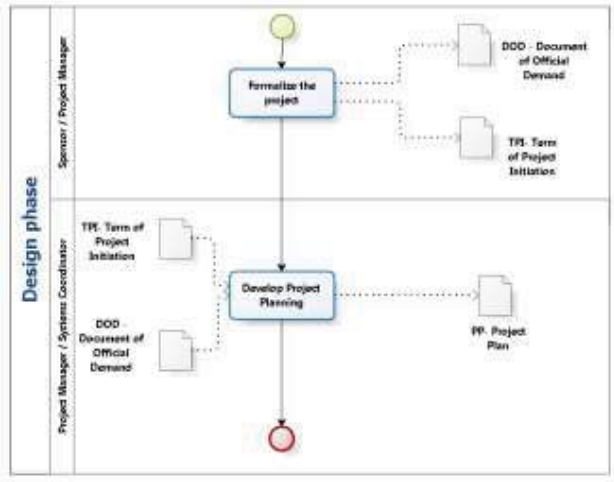


Fig. 5 – Project Inception Phase.

B. Elaboration Phase

The Elaboration phase will be realized by the team denominated as "Requirements team", and will have as its underlying role to inspect the requirements for the demanded feature. This phase should carry out the approximation and communication activities among the other teams that work on other project modules related to the integrated system. This alignment will occur by means of weekly meetings between the team Systems Coordinators. Presented in Figure 6 are the processes for the Elaboration Phase.

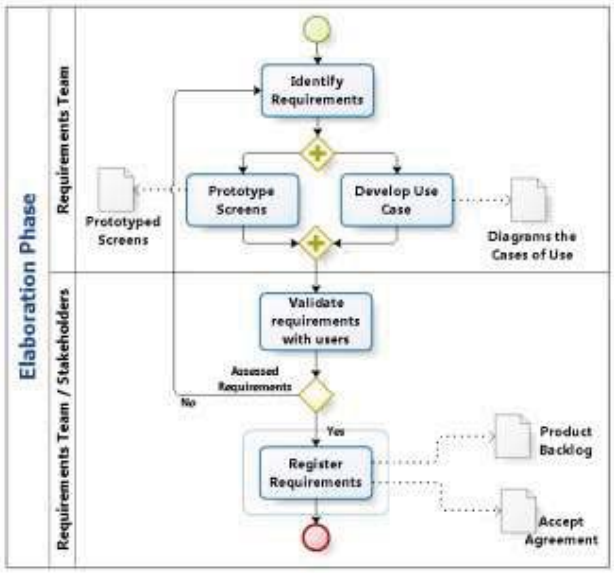


Fig. 6 – Elaboration Phase of the MDS-GRP.

In order to demonstrate how the iteration should transpire, the requirements team needs to carry out weekly meetings with the stakeholders, so that these present the concepts behind the definitions and flow necessary for attending to the demand. All requirements should be registered by the requirements team, and then are transformed into screen/report prototypes. Apart from the prototypes, the requirements team needs to draw up the diagrams for Cases

of Use with the objective of presenting to the module stakeholders, the understanding behind the conducted interaction. When inconsistencies occur in the requirements raised and these are constituted as invalid, new iterations are performed in order to validate the feature demanded. This flow continues until the final validation of the requirements by the applicant. Once the understanding has been obtained it needs to be formalized through the artefact "Agreement Acceptance", which will contain the signature from one of the stakeholders indicated by the sponsor or by the project sponsor. Further still, in this phase the requirements team should always feed the Product Backlog document with the features registered in the module, as well as the time estimated for implementing each one.

It is recommended that in order to identify the requirements, the cycle be divided into three instants, those being the first for raising the feature requirements, the second for presenting the prototypes and the diagrams for Cases of Use and carry out any possible adjustments. Finally, the third for presenting the realized adjustments, thus closing the raising requirements phase cycle validated by the client.

C. Construction Phase

The Construction phase refers to the activities for system architecture, implementation and tests. For this phase, the team responsible for all the activities is denominated as "The Development Team", and has as its responsibility to perform the codification and the tests of the proposed module. Also in this phase, the participation of the stakeholder is projected into the performing of tests on the features that have been released. Following on, Fig. 7 presents the Construction phase.

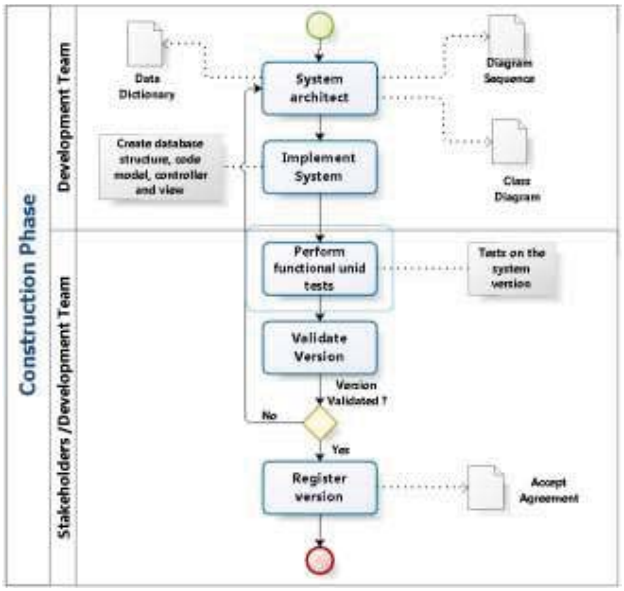


Fig. 7 – Construction phase of the MDS-GRP.

In this phase, besides the textual artefacts and graphs being generated, the source code and all the data structure necessary for the production of the module are also generated. Highlighted here therefore, is the need for integrating the already existing modules, along with the data structures common to the assistance provide to all system components in an integrated system with GRPs features. Hence, the development team should try to see that there are no inconsistencies or redundancies contained in the features or information.

Another important point is that the teams should be aware of, is the standardization in the three-tier development (model, vision, controller), in order to establish a unit and an alignment between all the developers from all the teams. Moreover, the codifications should be directed by artefacts defined by the requirement team, and these can only be codified after the acceptance agreement has been signed by the requesting agent involved in the iteration.

D. Transition Phase

Finally, the Transition phase culminates in the closing of the development cycle for the module. In this phase, a report is put together that should register the possible integrations with other modules, user training for those that will operationalize the module and the final acceptance agreement for the module signed by its sponsor. In Fig. 8, the processes developed in the Transition phase are presented.

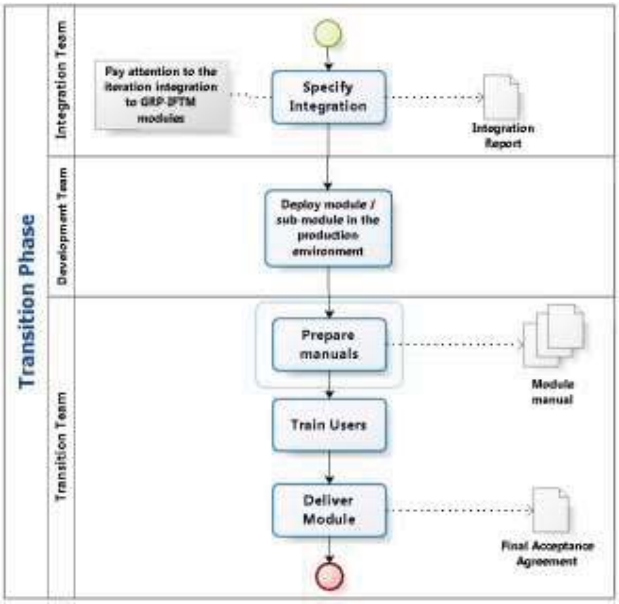


Fig. 8 – Transition phase of the MDS-GRP.

V. CASE STUDY

In the interest of defining a proof of concept, the SDM-GRP was applied to the three teams from the software manufacturer of a public educational institution known as the Federal Education Institute, Science and Technology of Mineiro Triangle (IFTM), in Brazil. The objective therefore was to evaluate the strong and weak points in the teams to reach better qualitative results in the process for systems development within the IFTM ambient.

The IFTM Institute

The Federal Education Institutes were created in December of 2008, with the objective of attending to the professional education and technology in the social context of Brazil. In consequence, the IFTM Institute was established. This institute

currently counts on 1000 full time staff for serving the 7830 classroom students and 1200 distance learners. As one can see, there is a meaningful amount of people and procedures to process all together.

The Teams

In the furtherance of attending to demands from the area of Information Technology and Communication, the organizational structure from the institution created a department called the Board of Information and Communication Technology. The purpose of this board was to attend to all the software projects from the IFTM. For this case study, the collaborators from the system teams were involved, in accordance with the organization scheme presented on Table III, which has three teams with a leader and two developers each. These teams work in line with one another in order to synchronize the development actions for the software projects.

TABLE III
Team Identification

Team Identification	Quantity of collaborators
Team1	One coordinator and two developers
Team2	One coordinator and two developers
Team3	One coordinator and two developers

Identification of the specific Case Study problem

The administrative and management models went through severe changes with the implementation of this new institution, which demanded emergent solutions concerning the automatization of these processes.

In addition to all the mapping performed at the inventory of the existing software at the IFTM facilities, an investigation was performed into how the systems implemented at the prior institutions were developed, and detected that were no formal methodologies registered, only individual activities and small systems were developed to meet the specific needs of sectors using the product-code methodology. Thus, there were no artefacts found, which existed as documentation only non-structured source-codes from different software.

In this scenario, the IFTM management flagged various problems, such as system integration, communication protocols between people and processes, as well as a significant amount of time spent in the detection of solutions to problems in the systems, among other forms of misuse.

The GRP-IFTM project

Under the presented scenario, a project called GRP-IFTM was created, which has as its aim to integrate all the administrative and management processes developed in the IFTM facilities. This project was divided into modules, as shown in Figure 9.

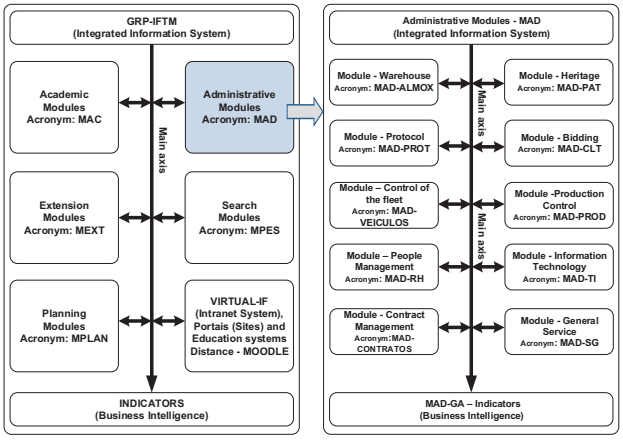


Fig. 9. Design vision of the project GRP-IFM

For this case study, the following modules MAD-ALMOX, MAD-GAB and MAC-PROT were separated, in order to be applied to the proposed methodology. The distribution of the sub-modules was organized as presented in Table IV.

TABLE IV.
Distribution of the sub-modules between teams

Team	Module	Description of the sub-modules
Team01	MAD-ALMOX	Inventories Module
Team02	MAC-PROT	Academic Protocol Module
Team03	MAD-GAB	Office Manager Module

VI. APPLICATION OF THE PROPOSED METHODOLOGY

The application of the proposed process was initiated through means of training, incorporating every member from the three teams. During training, all phases were presented, the papers and the artefacts that need to be generated. Besides the training of the proposed process, the management environment from the Redmine® project was presented and configured for this case study. In so doing, it was possible to monitor all the project activities, along with permitting the storage of the artefacts produced.

In the pursuance of evaluating the methodology in its varied stages, the Project denominated as MAD-GAB – (Office Management Module – the module that controls activities, demands, official documents and other administrative tasks associated with the IFTM rectory), was chosen as a case study. This choice is based on the level of involvement and area of expertise of the team in the referred project. In the following, the events (steps) associated with the evaluation of the methodology are presented in chronological form.

Step 1: the first interaction, planned as an activity in the design proposal stage, took place between the project sponsor (usually someone in senior management), with the Project Manager (Fig.). In this meeting, the outline of the project was established and its initial scope was set out. Through this, the first artefact designed for the methodology was produced and defined as the Document of Official Demand – DOD. In this same interaction, the macro features were presented by the sponsor (Private and Public agenda of

the Rectory, Registration of Demands, Document Emission and Meetings Manager) in order to attend to the required demand. These features were used to promote another forecast artifact: the Project Opening Agreement – POA.

In possession of these two artefacts DOD and POA, it was possible to elaborate the Project Plan (PP), where an estimated chronogram was elaborated for the next interactions. As a task for the Project Manager, it falls upon the activity organization and project tasks in the Management environment – Redmine®.

Step 2: the next iterations occur between the requirements team and the stakeholders responsible for the specific knowledge of the discussed area (chief of staff, executive secretary and administrative assistant's office). In the first team iteration, the previously generated artefacts were presented (DOD, POA and the Estimated Chronogram) and the first discussion cycle was initiated for raising the system requirements, directed by its features, as provided in the elaboration phase (Fig.). In practice, the implementation of a feature generates various requests. For example, a feature from the MAD-GAB module, denominated here as "Demand Management", generated various requirements necessary for its construction. For each feature/requirements identified, the team responsible registered this information on an appropriate form. Besides this, as previously stated, the minutes were taken for the discussions held by all present.

Step 3: Following on, the teams responsible for each group of requirements, that are using the methodology, go on to elaborate the screen prototypes and the diagrams for Cases of Use. To reach an approval for a screen design it is necessary to get the appropriate release and signature from the sponsor or a stakeholder designated by the very same sponsor.

After realizing these activities, it becomes the responsibility of the coordinator to elaborate the plan for the next meeting. Once again, the artefacts and the information resulting from the iterations are registered into the Redmine management system.

Step 4: Conducive to presenting the work produced, other iterations were performed in the furtherance of approving the requirements regarding each feature. In this meeting, the screen and the Cases of Use were adjusted in accordance with the understanding of those involved. When there is reached a mutual agreement and acceptance by the members of the project, two new artefacts are produced, the Product Backlog and the Acceptance Agreement. It is important to highlight that the Product Backlog is built upon and updated at each iteration. However, in its final version, this document presents an effort estimate as support to the development team.

Step 5: Now, in the construction phase, the developer team uses the requirements (and artefacts) registered in the Redmine environment, for the elaboration of the Data Dictionary, the Sequence Diagram and the Class Diagram (Fig.). Next, the team created the codification artefacts and definitions for the bank structure for attending to the features.

Step 6: Finally, in the Transition phase (Fig.), the integration team presents the feature to those teams responsible for other GRP components, and register the possibilities for integration with other modules from the system. The artefact hoped for through this step is the Integration Report.

Step 7: Still in the Transition phase, the development team implements the feature from the module on the production server, under the intention of delivering the functional component in operation. Thus, it is first necessary to elaborate the part of the manual that refers to the approved feature. This step is important for training the users on how to operate the released feature in an effective and efficient manner. The last artifact from the cycle is the Final Acceptance Agreement, duly signed by the stakeholder involved or by the project sponsor.

VII. CONCLUDING REMARKS

This work presented a proposal for the adjustment of the Unified Process for the development of software. This was applied through means of a case study at the Federal Institute of Education, Science and Technology of Triângulo Mineiro. The aim herein was to offer a good observance to the development process, reflecting upon time saving during the generation of artifacts. Noteworthy, is the fact that the software producer had no prior methodology to which it adhered. Therefore, the challenge laid down in this proposal was to adapt known methodologies to the reality of the IT teams involved.

In the interest of measuring the result of the methodology and perform the comparison between the teams, a questionnaire was put to both the system coordinators and its developers. In this questionnaire, aspects of relevance were touched upon to demonstrate the importance of adopting this methodology. To reach this end, the evaluators were classified as "Systems Coordinator" or "Developer", and went on to answer ten multiple-choice questions, under the options of "Very Satisfied", "Satisfied" and "Unsatisfied". Further still, for each question, the questionnaire allowed the evaluator to freely describe opinions and/or possible improvements for adjustments to the actual version of the SDM-GRP.

After the application of the questionnaire and by means of an analysis of the results obtained separately from both System Coordinators and Developers, these demonstrated the need for some required adjustments, intended for the advance of this work.

In terms of conclusions concerning the application of the methodology, through use of the presented case study, some highlighted points are addressed and the conclusion drawn is as follows:

- The application of the SDM-GRP produced improved results, in aspects concerning communication between the development teams.
- A revision based on the practice of interface prototyping for consideration by stakeholders is required.
- With a view to obtaining improved results from the activities related to client requirement surveys, it is pertinent to consider suggestions from evaluators concerning the artifact Activity Diagram, as this is grounded on an improved process compliance. However, the remaining artifacts worked upon were

considered sufficient for an understanding of the module.
- The need for the training of some team members for the mounting of diagrams UML.
- There should be more profound details for the execution of tests, before the final product release.

Finally, the adoption of methodology had as a result, better practices in project management for software and therefore one hopes to contribute in the perspective of the integration with other hierarchical models from the GRP-IFTM project.

VIII. FUTURE WORK

This work finds itself in the adaptation phase for validation of such techniques. Therefore, important points need to be addressed in future work, as for example, the raising of qualitative production data, for certifying if the adoption of this proposal is able to enhance the capacity for software production of its collaborators. Another relevant aspect is the application of the IBSP.BR framework (Improvements to Brazilian Software Processes), in order to improve the development capacity of the GRP-IFTM.

IX. ACKNOWLEDGMENTS

The authors would like to thank UFU (Federal University of Uberlandia), IFTM (Federal Institute of Triangulo Mineiro), CAPES (Coordination for the Improvement of Higher Education Personnel), CNPq (National Development Council Science and Technology) and FAPEMIG (Support Foundation Research of the state of Minas Gerais) for all their financial support given to this research project.

X. BIBLIOGRAPHIC REFERENCES

ALVES, N., CARVALHO, W., CARDOSO, A., LAMOUNIER JUNIOR, E.. **Um estudo de caso industrial sobre integração de práticas ágeis no RUP**. Revista Ciência e Tecnologia, América do Norte, 14, mar. 2012. Disponível em:http://revistavirtual.unisal.br:81/seer/ojs-2.2.3/index.php/123/article/view/169. Acesso em: 21 Fev. 2016.

BERTOLLO, Gleidson; FALBO, Ricardo de Almeida. **Apoio Automatizado À Definição de Processos em Níveis**. In: II Simpósio Brasileiro de Qualidade de Software, 2003, Fortaleza, Ceará. Anais do II Simpósio Brasileiro de Qualidade de Software, 2003. p. 77-91.

BOEHM, B.; TURNER, R. **Balancing Agility and Discipline:** Evaluating and Integrating Agile and Plan-Driven Methods. International Conference on Software Engineering. Washington, DC, USA: IEEE Computer Society. 2004a.

COHEN, D.; LINDVALL, M.; COSTA, P. **An Introduction to Agile Methods**. Amsterdam: Elsevier, v. 62, 2004.

FUGGETTA, Alfonso. **Software Process: A Roadmap**. In: 22nd International Conference on on Software Engineering (ICSE), Proceedings of the Conference on the Future of Software Engineering, New York: ACM Press, 2000. pp. 25-34.

GREER, D.; CONRADI, R.. **Software Project Initiation and Planning: An Empirical Study.** Institution Of Engineering And Technology, Northern Ireland, UK, v. 5, n. 3, p.356-368, 01 out. 2009.

ISO/IEC 12207, 1995**, Information Techonology – Software Life-Cycle Processes**

LEFFINGWELL, D**. Scaling Software Agility**. Reading: Addison Wesley, 2006.

MACHADO, Luis Filipe Dionisio Cavalcanti. **MODELO PARA DEFINIÇÃO DE PROCESSOS DE SOFTWARE NA ESTAÇÃO TABA.** 2000. 123 f. Dissertação (Mestrado) - Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2000. Cap. 1.

Mattioli, F. E. R. ; CAETANO, D. ; Cardoso A ; Lamounier, E. . **On the Agile Development of Virtual Reality Systems**. In: Int´l Conf. Software Eng. Research and Practice - SERP´15, 2015, Las Vegas. The 2015 World Congress in Computer Science Computer Engineering and Applied Computing. San Diego, CA, USA: CSREA Press, 2015. v. 01. p. 10-16..

NORD, R.; TOMAYKO, J. Software Architecture-Centric Methods and Agile Development. **IEEE Software**, 2006.

PAULA FILHO, Wilson de Pádua. **Engenharia de Software:** Fundamentos, Métodos e Padrões. 3ª ed. Rio de Janeiro: Ltc, 2009.

PRESSMAN, Roger S.. **Software Engineering:** A Practitioner's Approach. 6th New York: Mcgraw-hill, 2006.

OSTERWEIL, L.. **Software Process Are Software Too**. In: 9th International Conference on Software Engineering (ICSE), Monterey, Estados Unidos, 1987, p. 2-13.

RAMSIN, R.; PAIGE, R. F. **Process-Centered Review of Object Oriented Software Development Methodologies**. ACM Computing Surveys, v. 40, n. 1, 2008.

ROCHA, Rodrigo. *et al*. **Uma Experiência na Adaptação do RUP em Pequenas Equipes de Desenvolvimento Distribuído**. In: II WORKSHOP DE DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE - WDDS, 2008, Campinas-SP. Anais do II Workshop de Desenvolvimento Distribuído de Software - WDDS. Campinas-SP: Universidade Estadual de Maringá (UEM), 2008. p.81-90

SHACH, Stephen R.. **Engenharia de Software:** Os Paradigmas Clássico Orientado a Objetos. 7ª ed. São Paulo: Mcgraw-hill, 2009.

SOMMERVILLE, Ian. **Engenharia de Software.** 9ª ed. São Paulo: Pearson, 2011.

SOUZA, Francisco Flavio de ; BRAGA, R. T. V. . **Um Método de Desenvolvimento de Sistemas de Grande Porte Baseado no Processo RUP**. In: 1º Simpósio Brasileiro de Sistemas de Informação, 2004, Porto Alegre - RS. Anais do 1º SBSI, 2004. p. 31-38.

A Validation Strategy for an Automatic Code Generator using Java Pathfinder

S. L. M. Barrocas[1] **and M. V. M. Oliveira**[1,2]
[1]Departamento de Informática e Matemática Aplicada, Universidade Federal do Rio Grande do Norte
[2]Instituto Metrópole Digital, Universidade Federal do Rio Grande do Norte
Natal, RN, Brazil
Contact: marcel@dimap.ufrn.br

Regular Research Paper

Abstract— *The use of formal methods in software engineering considerably reduces the number of errors throughout system developments by enforcing a rigorous specification and verification before reaching a final implementation. The translation from a formal specification to executable code can be automatically achieved using automatic translators. The correctness of such translators, however, is still an open issue. This paper proposes an approach to validate such translators and use it to validate JCircus, a translator from Circus to Java. The validation encompasses the development of a strategy to model check executable programs automatically generated from Circus specifications by JCircus. Here, we focus on refinement model checking; hence, our strategy checks if the generated code refines its source specification. Together with coverage-based testing techniques, our strategy validates JCircus, making it a more robust and trustable automatic translator from a formal specification language to a programming language.*

Keywords: formal methods, model checking, validation, *Circus*, code synthesis

1. Introduction

Formal methods are techniques that specify systems using formal notations underpinned by rigourously defined semantics. Due to the effort required to apply such techniques, their application usually become expensive. For this reason, their use have normally been justified only in the implementation of concurrent and safety-critical systems. An interesting effort to minimize the costs of applying formal techniques is the implementation of automatic translators from formal notations to mainstream programming languages.

Circus [1] is a formal specification language whose syntax combines the syntaxes of Z [2] and CSP [3]. This feature of *Circus* allows the representation of concurrent systems with large amount of data in a non-implicit fashion. *Circus* has a refinement calculus [4] with transformation rules that can be used to refine *Circus* abstract specifications into *Circus* concrete implementations. *Circus* has been provided with both, a denotational semantics [4], and also an operational semantics [5] that contains rules that can be applied to generate labelled predicate transition systems (LPTS) of a given specification. A LPTS is a structure that represents the specification as a graph with nodes that represent the states of a *Circus* specification, and arcs that represent its possible execution paths.

JCircus [6], [7] is a tool that translates *Circus* specifications into executable Java code. The generated code uses JCSP [8], an API that provides abstractions to easily implement in Java some CSP constructs such as communication, multi-synchronisation, parallelism and choice. *JCircus*' generated code not only contains a JCSP implementation of each process in the specification, but also a Graphical User Interface (GUI) that enables the interaction of the user with the translated process.

The use of automatic translators like *JCircus* facilitates the implementation of systems that are based on a formal specification. These translators, however, are themselves usually susceptible to implementation errors. This may be due to errors in the translation strategy or even to simple implementation errors. Thus, to increase the confidence of the generated code, a validation strategy is imperative. An interesting approach is the combined use of software model checking and testing techniques. In this paper, we propose a strategy to validate *JCircus* by verifying, using software model checking, the correctness of the code generated by *JCircus* for a set of *Circus* specifications whose translations guarantee maximum Decision-Coverage (DC) of the *JCircus*' source code. This strategy, depicted in Figure 1, is based on Java Pathfinder (JPF) [9], a software model checker developed by NASA. The strategy can be summarised as:

1) Generate the Java code that implements the specification using *JCircus*;
2) Generate the LPTS of the specification, which represents the semantics of the specification;
3) Generate a JPF Model of the LPTS, and analyse its interaction with the generated code to check if it correctly implements the original specification;
4) Using EclEmma[1], repeatedly apply steps 1 to 3 to the

[1]http://www.eclemma.org/

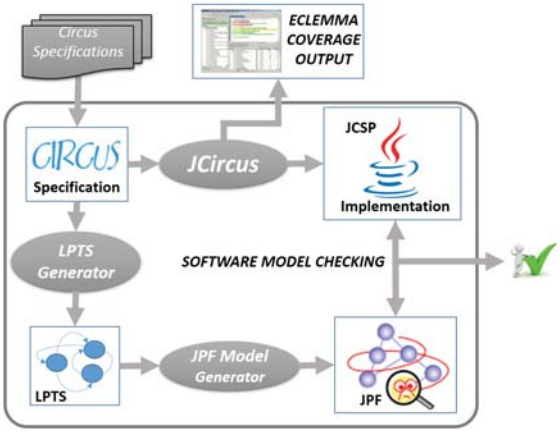


Fig. 1: Overall *JCircus* Validation Strategy

elements of a set of *Circus* specifications, ensuring maximum Decision-Coverage of the *JCircus* source code.

This paper is organised as follows. In Section 2, we introduce the relevant preliminary material. Section 3 details our strategy for validating *JCircus*, briefly explaining the *Circus* operational semantics and how it is used to generate the JPF model used to model check the generated Java code. Finally, Section 4 describes related work and Section 5 summarises our results and future directions of our work.

2. Preliminaries

2.1 *Circus*

Circus [1] is a formal specification language that combines Z [2], CSP [3] and Dijkstra's language of guarded commands [10]. This combination makes *Circus* suitable to represent concurrent systems with complex data structures. *Circus* has an associated refinement calculus [4] with rules that allow transformations from a centralised abstract specification to a concrete distributed implementation.

A *Circus* specification is defined in terms of paragraphs. Each of these paragraphs can either be a Z paragraph, a channel definition, a channel set definition, or a process declaration. The declaration of a process is composed of its name (possibly followed by parameters) and its definition. A process may be explicitly defined, or it may be defined in terms of other processes (compound processes). When a process is explicitly defined, besides the definitions of the state (using Z) and the main action, we have in its body Z operations on the state and definitions of (possibly parametrised) actions; they are used to specify the main action of the process.

An action can be a schema expression, a guarded command, an invocation to a previous defined action, or a combination of these constructs using CSP operators. Furthermore, state components and local variables may be renamed; however, no channel name can be changed.

$$
\begin{array}{l}
\textbf{process}\ CONTROLLER \mathrel{\widehat=} \textbf{begin} \\
\textbf{state}\ ACST == [preferred : \mathbb{N}] \\
INIT \mathrel{\widehat=} preferred := 25 \\
CTR \mathrel{\widehat=} \\
\quad \mu X \bullet switchoff \rightarrow Skip \\
\qquad \Box\ preferredtemp?np \rightarrow preferred := np; X \\
\qquad \Box\ startcycle \rightarrow getplug?p \rightarrow getturn?t \rightarrow \\
\qquad\quad gettemp?tp \rightarrow \\
\qquad\qquad \left(\begin{array}{l} (p = IN \land t = ON \land preferred < tp)\& \\ \qquad cooldown!preferred \rightarrow endcycle \rightarrow X \\ \Box\ (p = IN \land t = ON \land tp \leq preferred)\& \\ \qquad cooldown!tp \rightarrow endcycle \rightarrow X \\ \Box\ (p = OUT \lor t = OFF)\& \\ \qquad endcycle \rightarrow X \end{array} \right) \\
\bullet\ INIT; CTR \\
\textbf{end} \\
\textbf{process}\ SENSOR \mathrel{\widehat=} \textbf{begin} \\
\quad \textbf{state}\ SensorSt == [memory : \mathbb{N}] \\
\quad INIT \mathrel{\widehat=} memory := 0 \\
\quad SNSR \mathrel{\widehat=} \mu X \bullet readtemp?nt \rightarrow memory := nt; X \\
\qquad\qquad\qquad \Box\ gettemp!memory \rightarrow X \\
\qquad\qquad\qquad \Box\ switchoff \rightarrow Skip \\
\quad \bullet\ INIT; SNSR \\
\textbf{end} \\
\textbf{process}\ AIRCONTROLLER \mathrel{\widehat=} \\
\quad (CONTROLLER \ [\![\{\!|\ gettemp, switchoff\ |\!\}]\!]\ SENSOR) \\
\quad \setminus \{\!|\ gettemp\ |\!\}
\end{array}
$$

Fig. 2: The specification of the *CONTROLLER*

Circus has three primitive actions: *Skip*, *Stop* and *Chaos*. The action *Skip* terminates successfully and does not change the state. The second action deadlocks and *Chaos* diverges. The prefixing operator is standard, but a guard construction may be associated with it. For instance, if a given Z predicate p is *true*, the action $p\&c?x \rightarrow A$ inputs a value through channel c and assigns it to the locally defined variable x, and then behaves like A, which has x in scope. If, however, the condition p is false, the same action deadlocks. Such enabling conditions like p may be associated with any action.

In order to illustrate *Circus*, we present the specification of a room controller that regulates its temperature according to the preference of the user. The room has a controller that receives the preferred temperature of the user and outputs an air with a temperature that cools down the room accordingly. There is a sensor that captures the environment temperature and communicates it to the controller. In order to work, the air controller needs to be plugged in and turned on. We present the specification of the controller in Figure 2.

The process *CONTROLLER* has a single state component, *preferred*, that stores the current preferred temperature. Furthermore, this process has two actions, *INIT* and *CTR*. The

action *INIT* initialises the variable *preferred* to 25. Next, the action *CTR* initially offers a choice between (1) switching off the controller (*switchoff*) and terminating, (2) setting the preferred temperature (*preferredtemp*?*np*), in which case the value of *preferred* is updated, or (3) starting a cycle (*startcycle*) for cooling down the room. If a cycle is started, the controller retrieves the status of the plug, the status of the sensor, and the environment's temperature. If the plug is in and it is turned on, it cools down the room by outputting the minimum between the preferred temperature and the room temperature, after which it indicates the end of the cycle through *endcycle* and recurs. If either the plug is out or it is turned off, it takes no further action and simply indicates the end of the cycle before recursing. The main action of the process is given after the symbol •. It defines the process behaviour: the *CONTROLLER* performs the initialisation *INIT* and then behaves like *CTR*.

The controller interacts with a temperature sensor, that captures the room temperature and sends it to the controller. After its initialisation, the sensor may either (1) retrieve the temperature of the environment (*readtemp*?*nt*), in which case it updates its memory, or (2) output its value through channel *gettemp*, or (3) be switched off via channel *switchoff*.

The process *AIRCONTROLLER* specifies the whole system. It is defined as the parallel composition of the *CONTROLLER* with the *SENSOR*, hiding the channel *gettemp* from the environment. The air controller and the sensor synchronise on channels *gettemp* and *switchoff*.

Circus has two main tools: *CRefine*, a tool that supports the application of the *Circus* refinement calculus [11]; and *JCircus* [6], a tool that translates *Circus* specifications into Java code. The generated code uses the JCSP API [8], which implements most CSP constructs, channels, prefixing, parallel composition, multi-way synchronisation, communication and choice. *JCircus* is based on transformation rules of a translation strategy proposed in [4] and extended in [7].

JCircus has an extremely simple GUI in which the user simply selects the input specification file and defines the name of the output java project. Finally, the user is asked to choose the system's main process. This choice defines the behaviour of the program that is automatically generated for specifications compliant with *JCircus*' requirements. If, however, the specification is not compliant, the translation aborts and errors messages are displayed.

For each process on the specification given as input, *JCircus* generates a Java program that implements its behaviour and a Graphical User Interface (GUI) that allows a direct interaction between the user and the processes. For example, Figure 3 illustrates the GUI of the *AIRCONTROLLER*. Each button corresponds to a visible channel of the process: *readtemp*, *startcycle*, *preferredtemp*, *switchoff*, *getturn*, *getplug*, *cooldown* and *endcycle*. The channel *gettemp*, however, is hidden from the external environment and is not part of its GUI. Each button may be associated with combo boxes

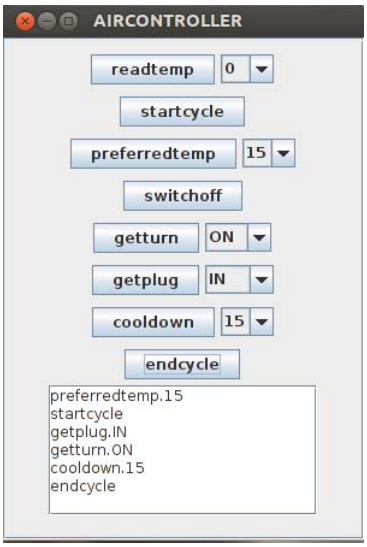


Fig. 3: GUI of the *AIRCONTROLLER* implementation

that correspond to communication fields. When a button is clicked, a successful synchronisation on the corresponding event is logged in the text area. If the event is not being offered, nothing happens.

2.2 Java Pathfinder

Java Pathfinder (JPF) [9] started as a software model checker. It, however, currently has various different execution modes and extensions. All these modes and extension are commonly used to verify Java programs. In our work, we focus on JPF's original feature, software model checking, which is still what JPF is mostly associated with.

JPF is able to verify the bytecode of a Java program. It works as a virtual machine that executes a Java program in all possible ways, searching, for example, for property violations and deadlocks. Using JPF, we are able to model check a given Java code by defining the target class, i.e. the class that we want to model check, and providing a JPF model that executes the target class in all possible ways. For example, let us consider the following Java program that sums two integers.

```
public class SumIntegers {
  int x, y;
  public SumIntegers (int x, int y){
    this.x = x; this.y = y; }
  public int sum () {
    return this.x + this.y; }
}
```

Using JPF, we may defined the following model that generates values for `SumIntegers`.

```
public class SumIntegersModel{
  public static void main (String[] args) {
    int x = Verify.getInt(1,3);
    int y = Verify.getInt(1,2);
    SumIntegers si = new SumIntegers(x,y);
    System.out.println(si.sum());
```

```
    }
}
```

This model exemplifies the use of the `Verify.getInt` choice generator, which generates execution paths for the given range of values. The execution of this model, for example, runs `SumIntegers.sum()` in all possible ways with `x` ranging from 1 to 3, and `y` ranging from 1 to 2.

3. Model Checking *Circus* Programs

In order to achieve our main goal, the validation of the automatic translator from *Circus* to Java, we have defined a set of *Circus* specifications whose translations satisfy coverage criteria described in Section 3.3. For each specification in this set, we follow a strategy that ensures that the behaviour of its translation is a valid refinement, which means that the execution of the Java code behaves in accordance with the original specification. This strategy automatically checks the possibility of execution of sequences of events (traces) and compares this behaviour (acceptance or refusal) with the original intended behaviour described in the specification. The strategy consists of using a compiler we developed (LPTS generator), which receives a *Circus* specification and produces the corresponding LPTS. Based on this LPTS, we automatically generate a JPF model that is used by JPF to exercise the generated code, checking the availability of the corresponding events. The LPTS generation was based on the *Circus* operational semantics, which we discuss on the sequel.

3.1 *Circus* Operational Semantics

Circus operational semantics was first presented in [12]. This semantics consists of a set of transition rules that allows the transformation of a *Circus* specification into a LPTS, which is composed of:

- A set of nodes: each node stores the current values of the state components (S), the text of the program that remains to be executed (P), and the constraint (C), a boolean expression that indicates if the node is enabled or not;
- A set of arcs: each arc links two nodes, a source node and a destination node. The arc can be labelled either with an ordinary event or with the special silent event τ. Labels with ordinary events indicate that the program may evolve from the source node to the destination node if, and only if, it performs this particular event. If it is labelled with a τ, the program may evolve silently from the source node to the destination node (silent transition). This means that, from the environment perspective, the system can be in either node.

Each of the transition rules links two nodes using an arc. A transition $n1 \rightarrow_a n2$ establishes that, if the event a occurs, the system evolves from state $n1$ to state $n2$.

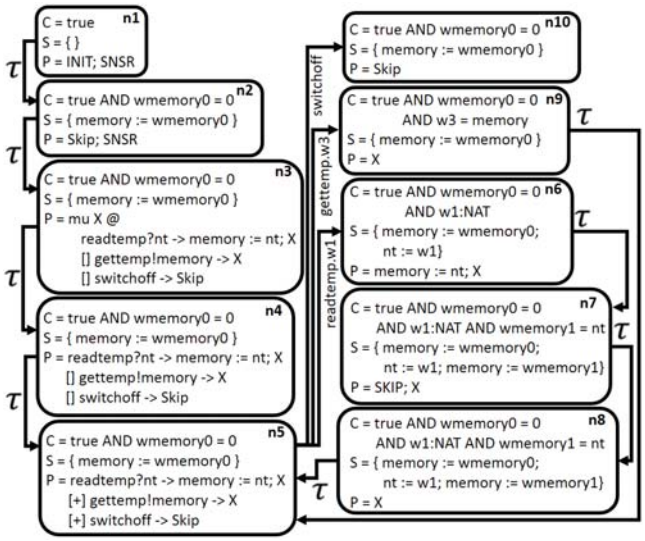


Fig. 4: LPTS of process *SENSOR*

By way of illustration, the LPTS of the *SENSOR* is presented in Figure 4. The initial node $n1$ is the source node of the silent τ transition to the destination node $n2$: it consumes the content of the call to *INIT*, which is an assignment, and then converts it to *Skip*. From $n2$ to $n3$, *Skip* is consumed (also silently) on the program text. The transitions from $n3$ to $n5$ are related to the consumption of the declaration of the *mu* action (μX) and the first step of the external choice treatment. The node $n5$ leads to three possible paths: *gettemp*.$w3$, *readtemp*.$w1$, or *switchoff*. When *gettemp*.$w3$ is consumed, the program text is formed by the recursive call X, and the program returns silently to node $n4$. If *readtemp*.$w1$ is consumed, there is a sequence of consumptions of the remaining sequence ($temp := nt; X$), which finishes with a silent transition that consumes X and leads back to node $n4$. Finally, if *switchoff* is consumed, the program terminates on node $n10$ with *Skip* on the program text. In some transitions like those that consume *gettemp*.$w3$ and *readtemp*.$w1$, loose constants like $w3$ and $w1$ are declared on the LPTS. In these cases, constraints on these constants are also added to the target node.

The output of our LPTS generator is the input given to our JPF model generator, which generates the JPF model that is used to validate the Java translation of the *Circus* specification. The generation of this model is discussed next.

3.2 JPF Model Checking

The JPF Model generated from the LPTS of a given specification exercises the code generated by *JCircus*. This exercise internally tries to synchronise on all possible combinations of events (considering different values) and checks if the acceptances and refusals correspond to the expected behaviour. The synchronisation attempts are done by simulating "clicks" on the GUI buttons that correspond to each

individual event with the desired values.

Our tool exercises all the transitions of all paths of execution. Basically, branches are originated either from choices (internal or external) or from the generation of values for communication, which are represented as loose constants in the LPTS. Finitely branching is guaranteed by imposing some restrictions on the domain of the variables.

The treatment of non-τ transitions is different from the treatment of τ transitions. The non-τ transitions are used to build the set of accepted events and the set of refused events at each source node. The former is composed of events whose transitions lead to destination nodes with constraints evaluated to *true*, and the latter is composed of events that either have no transitions leaving the node or whose transitions lead to destination nodes whose constraints evaluate to *false*. If either a click on an event is accepted and this event is in the set of refused events or a click on an event is refused and this event is in the set of accepted events, the refinement fails. Otherwise, the exercise continues by examining the target nodes.

The τ transitions do not cause any event click. In these cases, we just check the constraints of the destination nodes. If any of them fails, the refinement fails. Otherwise, the exercise continues by examining the target nodes. This examination continues until no further non-examined destination nodes are found. Finally, the refinement is successful if, and only if, it is successful for all paths.

By way of illustration, let us consider the following specification of a stateless process that internally chooses to offer either a or b and stops after communication:

$$\textbf{process}\, P \mathrel{\widehat{=}} \textbf{begin} \bullet a \rightarrow \mathit{Stop} \sqcap b \rightarrow \mathit{Stop}\, \textbf{end}$$

The first step of our strategy is to translate the specification of P into Java (using *JCircus*). The resulting code has a non-deterministic choice between both branches. The refinement analysis of each path is achieved by exercising all possible sequences of events of the process on the LPTS (if the arc of the transition is silent), checking if each sequence is supposed to be accepted or not. A simplified version of the LPTS of the P process is presented in Figure 5.

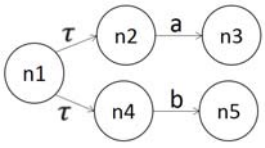


Fig. 5: LPTS of the P process

Using our strategy, the refinement of P is successful if, and only if, it is successful for all possible branches, both of which, start from $n1$:

$$\mathit{refinement}(P) \equiv \mathit{refinement}(n1)$$

In Figure 5, we can see that $n1$ is the source node of two τ transitions, whose destination nodes are $n2$ and $n4$. These nodes are reachable only if their constraints are evaluated to *true*. The function $\mathit{constrs}(n)$ returns the constraint of a given node n). As both transitions are silent, the refinement is successful for $n1$ if at least one of them is successful:

$$\begin{aligned}&\mathit{refinement}(n1)\\ &\equiv (\mathit{constrs}(n2) \wedge \mathit{refinement}(n2))\\ &\quad \vee (\mathit{constrs}(n4) \wedge \mathit{refinement}(n4))\end{aligned}$$

In our example, both constraints evaluate to *true*.

$$\begin{aligned}&(\mathit{constrs}(n2) \wedge \mathit{refinement}(n2))\\ &\vee (\mathit{constrs}(n4) \wedge \mathit{refinement}(n4))\\ &\equiv (\mathit{true} \wedge \mathit{refinement}(n2)) \vee (\mathit{true} \wedge \mathit{refinement}(n4))\\ &\equiv \mathit{refinement}(n2) \vee \mathit{refinement}(n4)\end{aligned}$$

Following the analysis, the node $n2$ has an arc labelled a going to $n3$, thus it expects a click on a to be accepted and a click on b to be refused. In what follows we use an auxiliary function $\mathit{acc}(e)$ that indicates if a given event e has been accepted by the translated code or not. The resulting refinement analysis of $n2$ is presented below. Since node $n3$ has no transitions leaving it, $\mathit{refinement}(n3)$ evaluates to *true*. Furthermore, no constraints were added and, therefore, $\mathit{constrs}(n3)$ also evaluates to *true*.

$$\begin{aligned}&\mathit{refinement}(n2)\\ &\equiv \mathit{acc}(a) \wedge \neg\, \mathit{acc}(b) \wedge \mathit{constrs}(n3) \wedge \mathit{refinement}(n3)\\ &\equiv \mathit{acc}(a) \wedge \neg\, \mathit{acc}(b) \wedge \mathit{true} \wedge \mathit{true}\\ &\equiv \mathit{acc}(a) \wedge \neg\, \mathit{acc}(b)\end{aligned}$$

Simmilarly, the resulting refinement analysis of $n4$ is:

$$\begin{aligned}&\mathit{refinement}(n4)\\ &\equiv \mathit{acc}(b) \wedge \neg\, \mathit{acc}(a) \wedge \mathit{constrs}(n5) \wedge \mathit{refinement}(n5)\\ &\equiv \mathit{acc}(b) \wedge \neg\, \mathit{acc}(a) \wedge \mathit{true} \wedge \mathit{true}\\ &\equiv \mathit{acc}(b) \wedge \neg\, \mathit{acc}(a)\end{aligned}$$

Concluding the refinement analysis for node $n1$, we have:

$$\begin{aligned}&\mathit{refinement}(n1)\\ &\equiv \mathit{refinement}(n2) \vee \mathit{refinement}(n4)\\ &\equiv (\mathit{acc}(a) \wedge \neg\, \mathit{acc}(b)) \vee (\mathit{acc}(b) \wedge \neg\, \mathit{acc}(a))\end{aligned}$$

The translated code randomly offers either a or b. We will name these possibilities (a) and (b), respectively. Our strategy exercises these two possibilities. A successful refinement requires both exercises to be successful. First, considering option (a), we have the event a being accepted and the event b being refused:

$$\begin{aligned}&(\mathit{acc}(a) \wedge \neg\, \mathit{acc}(b)) \vee (\mathit{acc}(b) \wedge \neg\, \mathit{acc}(a))\\ &\equiv (\mathit{true} \wedge \neg\, \mathit{false}) \vee (\mathit{false} \wedge \neg\, \mathit{true})\\ &\equiv \mathit{true}\end{aligned}$$

Finally, considering option (b), we have the event a being refused and the event b being accepted:

$$\begin{aligned}&(\mathit{acc}(a) \wedge \neg\, \mathit{acc}(b)) \vee (\mathit{acc}(b) \wedge \neg\, \mathit{acc}(a))\\ &\equiv (\mathit{false} \wedge \neg\, \mathit{true}) \vee (\mathit{true} \wedge \neg\, \mathit{false})\\ &\equiv \mathit{true}\end{aligned}$$

Hence, in both exercises we have a successful refinement. As a result, we conclude that our code is a successful refinement of process P.

The refinement analysis of non-recursive processes like P terminates when we analyse all possible paths of execution of P based on its LPTS. The analysis of each individual execution path terminates when a node that has no leaving arcs (*e.g.* nodes $n3$ and $n4$) is reached.

The verification of recursive processes as those presented in Figure 2 is achieved by tracking the nodes visited in the LPTS and the Java methods called during the system execution. A correct recursive behaviour presents a one-to-one correspondence between nodes (state values, constraints and program text) and method calls (and its real arguments). Hence, revisiting a node in the LPTS successfully terminates the verification of a particular execution path if, and only if, it is followed by the same method call in the Java code. The refinement, however, fails if the one-to-one correspondence is not respected.

3.3 Decision-Coverage Testing for *JCircus*

The model checking strategy described in the previous sections asserts the correction of a single *Circus* specification with respect to the *JCircus* code generated from it. It, however, does not guarantee the overall correctness of our translator. In order to increase the level of confidence ir our tool, we automatically applied the model checking strategy to a set of 67 *Circus* specifications whose translation requires maximum Decision-Coverage of *JCircus*' source code. This set was constructed considering not only the translatable subset of the *Circus* syntax [4], but also specific conditions used in the translation of communication, parallelism and multi-synchronisation (synchronisation of 3 or more parts). The translation of these constructs exercise different parts of *JCircus*' source code, depending on how they occur. For example, let us consider the following parallel composition of two actions running independently (also know as interleaving): $(a \rightarrow \text{Skip}) \,|||\, (a \rightarrow \text{Skip})$. This composition must be translated in a special way to avoid the parallel branches to synchronise on a. In such cases, *JCircus* execution flows differently and renames each occurrence a to avoid the actions to synchronise: $(a_1 \rightarrow \text{Skip}) \,[\![\, \{\!|\, a_1, a_2 \,|\!\} \,]\!]\, (a_2 \rightarrow \text{Skip})$. Internally, external communications on a are transformed to communications on either a_1 or a_2.

Using EclEmma, a free Java code coverage analysis tool for Eclipse, we analyse the Decision-Coverage of the source code of *JCircus* achieved by this exercise. The results are presented in Figure 6. As expected, our exercises did not achieve 100% of coverage. There are various acceptable reasons, described next, for not reaching full coverage.

Some of the code executed by JPF model generator like `jcircus.newutil` and `jcircus.parallelism` reuse libraries of *JCircus* that manipulate the AST. Some of the syntactic categories, like hiding and alphabetised

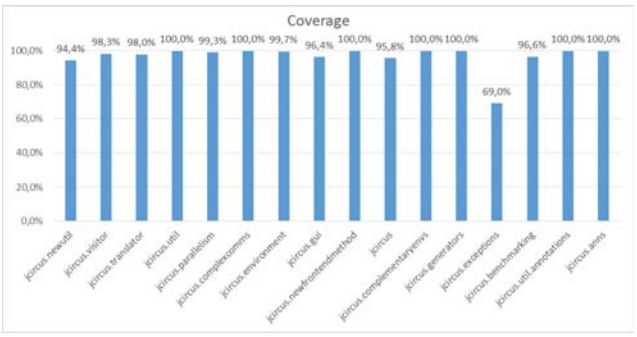


Fig. 6: Decision-Coverage of *JCircus*

parallel, are only used by the LPTS generator, but are not translatable. Therefore, they are not referred by *JCircus*. For similar reasons, the package `jcircus.exceptions` was not fully covered. This package contains exceptions that refer to non-translatable syntactic categories. This package also contains runtime exceptions, which are not expected to happen since this would indicate an ill-behaviour of *JCircus*. Finally, `Unmarshal` exceptions, which indicate problems in the parser execution, and `IO exceptions`, which indicate problems in the path to the input files are also exceptions that are not expected to happen in the translation of well-formed *Circus* specifications like those present in our test set. *JCircus* uses `Velocity`, an open source templating tool that generates code. Its classes are used in the packages `jcircus.visitor` and `jcircus.translator`. Among the referenced classes, some exceptions indicate missing templates for generating the JCSP code. These exceptions are not expected to take place because all templates are correctly available. Finally, as a good programming practice, at some points *JCircus* checks if objects are null before accessing them. Nevertheless, these parts of the program are unreachable in a well constructed program and are not exercised in our case.

4. Related work

Our approach is not the only one for validating a code generator. In [13], the authors present a strategy for validating a code generator for the B-method [14]. The strategy encompasses the automatic generation of inputs by encoding the grammar of B on an input generator called LGen [15] and the generation of test cases using BETA [2], which receives the B specification as input. The work of [13] is similar to ours because it uses Coverage-Based techniques to construct the test set of validation. The differences lie on the major conditions that guide the coverage measurement: while our work mostly considers inner conditions in *JCircus*, the approach of [13] focuses on grammar-based coverage criteria. Another difference is on the validation mode: while our strategy model checks the code generated by *JCircus*, the

[2]http://www.beta-tool.info/

strategy described in [13] checks the conformance of the generated code with the generated test cases.

The work presented in [16] describes a strategy to check the correctness of a code generator for Programmable Logic Controllers (PLC). Their approach generates a Timed Labelled Transition System (TLTS) [17] for both the input model and the generated PLC, and compares both TLTS using model checking. We, however, generate the LPTS of the specification and use it to construct a JPF model, which is then used to model check the translated code, using JPF software model checking.

5. Conclusions

The number of errors in a system development may be considerably reduced by using formal methods in the specification and verification of the system before reaching its final implementation. From a formal specification, the executable code may be automatically generated by translators. However, the correctness of such translators is still an open issue. This paper presents a strategy to validate *JCircus*, a formal translator from *Circus* to Java. This strategy focus on checking if the executable programs generated by *JCircus* refine their source specification using software model checking.

Our strategy for model checking involves the generation of the LPTS of the target specification and the generation of a JPF model that conducts the generated code. Our tool collects the results of this exercise and analysis the compliance of the behaviour of the executable code with the specification. This validation is done for a set of *Circus* specifications whose translations guarantee maximum Decision-Coverage (DC) of the *JCircus*' source code. As a result, by combining software model checking with coverage-based testing techniques, our strategy validates *JCircus*, making it a more robust and trustable automatic translator from a formal specification language to a programming language.

The approach presented in this paper differs from others in the literature that also envisage the validation of automatic translators [13], [16]. As far as we know, the strategies either use model checking at the level of the formal language or use testing at the level of the generated code. Here, we combine software model checking with testing by using the former to assert the correctness of the translation for a given specification and the latter to construct the set of specification to be translated.

The model checking strategy presented in this paper might be accommodated to validate further automatic translators from any formal language to Java. By providing an implementation of the components presented in Figure 1, the LPTS generator and the JPF model generator, developers may validate their translators by applying our validation strategy. An empirical study on the adaptation of our strategy to other formal languages and their code generators is in our research agenda.

Acknowledgments

This work was [partially] supported by the National Institute of Science and Technology for Software Engineering (INES), funded by CNPq, grants 573964/2008-4, 560014/2010-4 and 483329/2012-6. We thank Jim Woodcock for his suggestions on the implementation of the operational semantics.

References

[1] J. C. P. Woodcock and A. L. C. Cavalcanti, "A concurrent language for refinement," in *IWFM'01: 5th Irish Workshop in Formal Methods*, ser. BCS Electronic Workshops in Computing, A. Butterfield and C. Pahl, Eds., Dublin, Ireland, July 2001.

[2] J. C. P. Woodcock and J. Davies, *Using Z—Specification, Refinement, and Proof*. Prentice-Hall, 1996.

[3] A. W. Roscoe, *The Theory and Practice of Concurrency*, ser. Prentice-Hall Series in Computer Science. Prentice-Hall, 1998.

[4] M. V. M. Oliveira, "Formal Derivation of State-Rich Reactive Programs using *Circus*," Ph.D. dissertation, Department of Computer Science, University of York, 2006.

[5] L. Freitas, "Model-checking *Circus*," Ph.D. dissertation, Department of Computer Science, The University of York, 2005, yCST-2005/11.

[6] S. L. M. Barrocas, "JCircus 2.0: Uma extensão da Ferramenta de Tradução de Circus para Java," Master's thesis, Programa de Pós-Graduação em Sistemas e Computação - Universidade Federal do Rio Grande do Norte, 2011.

[7] A. Freitas, "From *Circus* to Java: Implementation and Verification of a Translation Strategy," Master's thesis, Department of Computer Science, The University of York, Dec 2005.

[8] P. Welch, N. Brown, J. Moores, K. Chalmers, and B. Sputh, "Alting barriers: synchronisation with choice in Java using JCSP," *Concurr. Comput. : Pract. Exper.*, vol. 22, no. **8**, pp. 1049–1062, June 2010. [Online]. Available: http://dx.doi.org/10.1002/cpe.v22:8

[9] W. Visser, K. Havelund, G. Brat, S. Park, and F. Lerda, "Model checking programs," *Automated Software Engg.*, vol. **10**, no. 2, pp. 203–232, Apr. 2003. [Online]. Available: http://dx.doi.org/10.1023/A:1022920129859

[10] E. W. Dijkstra, "Guarded commands, nondeterminacy and the formal derivation of programs," *Communication of the ACM*, vol. **18**, no. 18, pp. 453–457, 1975.

[11] M. V. M. Oliveira, A. C. Gurgel, and C. G. de Castro, "CRefine: Support for the *Circus* Refinement Calculus," in *6th IEEE International Conferences on Software Engineering and Formal Methods*, A. Cerone and S. Gruner, Eds. IEEE Computer Society Press, 2008, pp. 281–290.

[12] J. C. P. Woodcock, A. L. C. Cavalcanti, and L. Freitas, "Operational semantics for model-checking *Circus*," in *FM 2005: Formal Methods*, ser. Lecture Notes in Computer Science, J. Fitzgerald, I. J. Hayes, and A. Tarlecki, Eds., vol. **3582**. Springer-Verlag, 2005, pp. 237–252.

[13] A. M. Moreira, C. Hentz, D. B. P. D. E. C. B. de Matos, J. B. S. Neto, and V. G. M. Jr, "Verifying code generation tools for the b-method using tests: A case study," in *Tests and Proofs*. Springer, 2015, pp. 76–91.

[14] J.-R. Abrial, *The B-book: Assigning Programs to Meanings*. Cambridge University Press, 1996.

[15] A. M. Moreira and C. Hentz, "Geração de sentenças para testes a partir de descrições de linguagens," in *Proceedings of the Brazilian Workshop on Systematic and Automated Software Testing*, 2009.

[16] D. Pollmächer, W. Zimmermann, and H.-M. Hanisch, "Translation validation for model-based code-generators for plcs," in *Emerging Technologies and Factory Automation, 2005. ETFA 2005. 10th IEEE Conference on*, vol. 1. IEEE, 2005, pp. 8–pp.

[17] T. A. Henzinger, Z. Manna, and A. Pnueli, "Timed transition systems," in *Real-Time: Theory in Practice*. Springer, 1992, pp. 226–251.

Technical debt elicitation in text using natural language processing techniques

Adrian S. Barb
Information Science Department
Penn State University
Malvern, Pennsylvania 19335
Email: adrian@psu.edu

***Abstract*—In this article we explore a methodology for knowledge elicitation from textual information in software development communities. Our goal is to evaluate and predict whether technical depth is addressed in texts related to software development. Technical debt accumulation is a result of sub-optimal decisions taken during the life cycle of a software project and it is difficult to identify in textual information due to its pervasive nature. Our methodology used natural language processing techniques to possibilistically evaluate the relevance of a text to several aspects of software project management such as technical debt, architecture, code style, testing, design, organization governing, or documentation. We use data fusion to expand the inferred knowledge and the Choquet integral to evaluate the relevance of knowledge of each component of information. Further we identify a model of predicting technical debt using a linear combination of the other aspects. To test our approach we use sample text from IBM Developer Works website.**

***Index Terms*—Technical debt, software development, text mining, data fusion, Choquet integral.**

I. Introduction

A key component of agile software development methods is refactoring. Refactoring is the primary method to address the issue of technical debt that is characteristic to agile software development. Technical debt was originally described as "not-quite-right code" [1, page 30]. In agile projects, software teams are able to deliver projects faster by using rather immature architecture with the expectation that the sub-optimal parts would be be improved in subsequent releases. However, schedule pressure and developer specialization may lead, in the long term, to unmanageable and inflexible software projects [1] which require large amounts of refactoring.

Since its introduction, the concept of technical debt evolved to encompass any difficulties in the software development process including new requirements, architectural or code changes, documentation, or defects. Kruchten et al [2] proposed a theoretical framework for evaluating technical debt by limiting the scope of technical debt to only the invisible aspects of the project such as architecture, testing, documentation, or code debt. They define technical debt as "deferred investment opportunities or poorly managed risks" [2, page 20]. The task of evaluating the relevance of text that relates to a specific subject is difficult due to issues such as polysemy, synonymy, or idiosyncrasy of free text. For example, upon parsing a number of ten textbooks related to software, we created a dictionary of more than 10,000 different words related to software development life cycle. To handle such complexity, we need an effective method to categorize each evaluates word into more abstract categories that are easier to handle.

Researchers have developed classification schemes for linguistic events including several criteria for assigning verbs to classes. Research by Dowty [3] identifies four classes of verbs: state, activity, achievement, and accomplishment. The four classes are separated in terms of three characteristics: dynamicity, telicity, and durativity. A state verb, for example, differs from an activity verb by its lack of dynamic features. At the same time, an activity verb does not have a have a fixed termination point (atelicity) which is one of the main differences from an accomplishment verb. One of the most used verb classification system was developed by Levin [4] who classified over 3,100 English verbs in 47 top level categories based on their syntactical behavior. Similarly, the WordNet semantic network [5] was developed at Princeton University and includes more than 126,000 entries in four parts of speech: nouns, verb, adjective, and adverb. Each entry is assigned one or many lexical categories as shown in Table I. For example, the word "architecture is classified into three noun categories with different probabilities: artifact, action, and cognition with different degrees of belonging.

In this article we will assess the relevance of text to several software development related categories: architecture, code style, testing, design, governance, requirements, or documentation using quantitative measures. The models for each of these categories will include 15 conceptual spaces that correspond to the WordNet semantic classes of verbs. Each conceptual space will have 29 dimensions that correspond to the noun, adjective, and adverb categories. To create these spaces we extract subject-verb-object triplets from text. For each member of the triplet, we will determine the semantic class and assign a possibilistic value in the corresponding conceptual space. We will provide semantic expansion by identifying contextual entailments for each word and will aggregate the information using Choquet integrals. Our experiments will evaluate the correlation between technical debt and the rest software aspect models which will be evaluated against the Kruchten et al [2] proposed a theoretical framework. Then we conclude the article and provide future extensions to our approach.

ISBN: 1-60132-446-4, CSREA Press ©

II. Methodology

The main goal of our methodology is construct a quantitative representation of the input text across 15 verb conceptual spaces used in WordNet. For example, a sentence like "the developer anticipated a major refactoring effort" should return a higher relevance to the cognition space due to the fact that the "anticipate" verb is mostly used as a cognition verb. WordNet classifies this verb in the communication and social spaces as well but with a lower probability.

The methodology in shown in Algorithm 1. First, we parse a number of textbooks from the specific domain to build a custom dictionary, as shown in line 1-4 of the algorithm. Then for each custom text that we want to analyze, we extract the subject-verb-object triplets and assign them to word WordNet categories as shown in lines 5-10. Finally, we compute the relevance of a text to a software development aspect my computing the correlation between the target text and a reference text that was previously reviewed by experts. Details of the procedure are shown in subsequent sections.

A. Building a Software Development Dictionary

In the first step of the procedure is to build a dictionary for software development. We accomplish this by analyzing a number of eleven texts related to the software development subject as follows: software engineering, object oriented programming, software testing, project management, patterns, agile development, architecture, and refactoring [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16]. These books were indexed using an inverted index [17]. A list of most common 418 stop words was removed from the index and each term was stemmed using the KSTEM method [18]. Further, the number of terms was further reduced through lemmatization using the TreeTagger approach [19]. For each of the resulting terms we computed the term frequency using the formula:

$$f_w = \frac{1 + log(count(w))}{1 + log\left(\sum_{\mathrm{j=1}}^{\mathrm{size(W)}} count(w_j)\right)} \quad (1)$$

In this formula w is a term/word in the set W that appears $count(w)$ times in the indexed texts. This procedure resulted in a number of 8,530 words used by the domain specific literature as shown in Table II.

TABLE I
Classes of verbs according to WordNet [5]

Part of speech	Semantic Classes
Nouns	action, animal, artifact, attribute, body, cognition, communication, event, feeling, food, group, location, motive, object, person, phenomenon, plant, possession, process, quantity, relation, shape, state, substance, time
Verbs	body, change, cognition, communication, competition, consumption, contact, creation, emotion, motion, perception, possession, social, stative, weather
Adjectives	all, relational
Adverbs	all, participial

TABLE II
Frequency of generated words by part of speech

Type	Word Count	Maximum Term Frequency
Noun	3,896	1
Verb	2,162	0.953
Adjective	1,860	0.851
Adverbs	612	0.893
Total	8,530	1

Figure 1 shows the term frequency for the top 1,000 words in each each part of speech category. For example the most frequent noun in the surveyed literature is "test" with a term frequency of 1 while the most frequent verb is "object" with a term frequency of 0.953. This vocabulary will be used to evaluate the relevance of terms in the free text that we will analyze.

B. Information Extraction from text

The information extraction procedure is shown in Algorithm 2. First, the text is tokenized into sentences and then each sentence is processed using natural language techniques. For each word in a sentence we extract several characteristics such as lemma, part of speech, and its position in the dependency tree, as shown in lines 6-10. The dependency tree is used to to extract subject-verb-object triplets. that will be used in building our models. We extract triplets from three types of related words: subject-verb-object, noun-adjective, and noun-noun combinations. For example, the fragment "the developer anticipated a major refactoring effort" has the following triplets: (1) develope-anticipate-effort, (2) effort is major, and (3) refactoring is effort, which correspond to the three types of triplets mentioned above.

Below we give a more in-depth example of text processing. Consider the following sentence about software architecture from the Wikipedia website: "Software architecture refers to the high level structures of a software system, the discipline of creating such structures, and the documentation of these structures." After executing steps 6-10 from the Algorithm 2 we obtain the following dependency tree as shown in Figure 2. Note that the dependency tree shows the type of relationship

Algorithm 1 Information extraction from text

OUTPUT: possibilistic relevance of text to software aspects

1: Build software development dictionary D
2: **for** EACH term w in D **do**
3: compute term frequency $f(w)$
4: **end for**
5: **for** EACH input document I **do**
6: $T \leftarrow ExtractTriplets(I)$ - extract tripplets
7: $T^{+} \leftarrow T \cup entailments(T)$ - entailment expansion
8: $T(I) \leftarrow Choquet(T(I))$ - reduce the model
9: $M(I) \leftarrow as.matrix(M(I)$
10: **end for**
11: **return** $corr(M(I_{td}), M(I_{reference}))$

Algorithm 2 Extract triplets

INPUT: Raw Text T
OUTPUT: Subject-Action-Object triplets

$S \leftarrow tokenize(I)$ - extract sentences
$result \leftarrow \{\}$
for EACH Sentence S **do**
  $W \leftarrow words(S)$
  for EACH word $w \in W$ **do**
    Extract part of speech $P(w)$
    Extract root word $L(w)$
    Extract word dependency $D(w, w_1)$
    Extract dependency type $DT(w, w_1)$
    Extract type for each verb $t(w)|w$ is verb
    for EACH subject-action-object triplet (s, a, t) **do**
      $result \leftarrow result \cup (L(s), T(a), L(o))$
    end for
    for EACH noun-adjective pair (s, j) **do**
      $result \leftarrow result \cup (L(s), state, L(j))$
    end for
    for EACH noun-noun pair $(n_1, n_2|n_2 = obj(n_1))$ **do**
      $result \leftarrow result \cup (L(n_1), state, L(n_2))$
    end for
  end for
end for
return $result$

between two words. The word "software architecture" is a subject in relation to the verb "refers to".

This figure shows the lemma for each word in the sentence, the part-of-speech using standard natural language notation and the relation type among words in the sentence.An example of a noun-verb-object triplet is "software architecture", "refer to", and "structure". The first noun is the subject with a term frequency of 0.75 in our dictionary and it has a 60% probability it refers to an artifact, 20% probability it refer to a cognitive process, and 20% probability that it refers to a

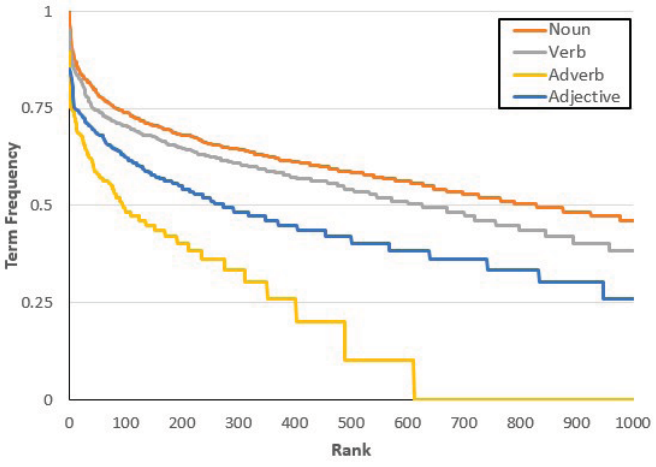


Fig. 1. Distribution of words by rank for each part of sentence type.

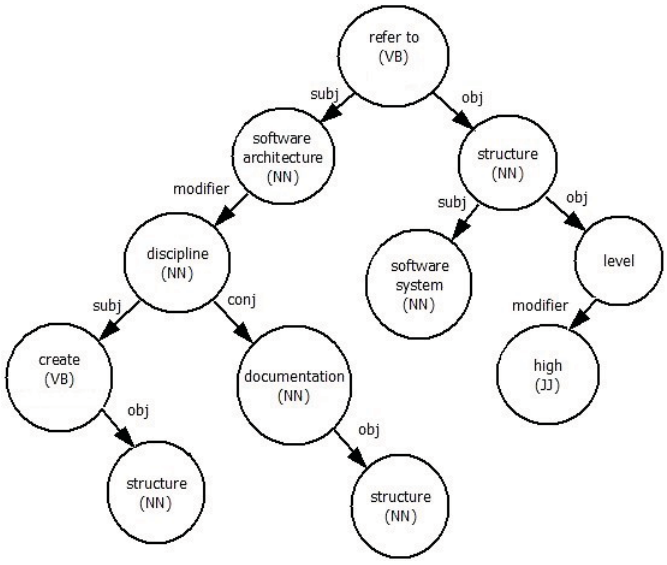


Fig. 2. Parsed dependency tree with part of speech and dependency type.

TABLE III
TRIPLETS EXTRACTED FROM THE PARSED DEPENDENCY TREE

Subject	Action Type	Object
architecture	refers to	structure
software	has	architecture
structure	has	level
level	is	high
system	has	structure
software	has	system
architecture	is	discipline
discipline	creates	structure
discipline	has	documentation
documentation	has	structure

human activity. Similarly, the noun "structure" is the object in the sentence and it refers to artifact, attribute, and cognition with probabilities 50%, 27% and 14.8% respectively. The verb "refer to" can be categorized as communication, stative, and cognition with probabilities 46.2%, 26.7% and 17.9% respectively. The Table III shows all the triplets extracted from this sentence.

C. Determining Contextual Entailments

One characteristic of text communication is the variability of semantic expression, where people express the same meaning can using different words [20]. To better generalize our models we need to evaluate words for which the meaning can be inferred from used in our text. For example, when analyzing the text provided above, one may imply that the use of the word "software system" may entail with some degree of relevance the word "application". That means that two different people may use the two words to convey the same meaning. In general, contextual entailments generation requires both understanding of language as well as common background knowledge and can be acquired by analyzing a large corpus of data. For the purpose of this article we will use the TextRazor [21] to generate contextual entailments.

TABLE IV
EXAMPLE OF ENTAILED WORDS WITH THEIR PROBABILITY FOR ONE SUBJECT-VERB-OBJECT TRIPLET

Word	Entailed Word	Contextual probability P	Term Frequency f	Choquet C	Relevant?
architecture	architecture	1.000	0.7504	0.3084	Yes
	system	0.697	0.9691	0.2776	Yes
	structure	0.561	0.7488	0.1726	Yes
	design	0.439	0.8725	0.1574	Yes
	layout	0.399	0.5118	0.0530	No
refer	refer	1.000	0.6431	0.6152	Yes
	utilize	0.270	0.2600	0.0671	No
	exemplify	0.201	0.3342	0.0642	No
	use	0.171	0.8363	0.1368	Yes
	form	0.167	0.7291	0.1164	Yes
structure	structure	1.000	0.7488	0.2021	Yes
	system	0.963	0.9691	0.2519	Yes
	architecture	0.941	0.7504	0.1906	Yes
	framework	0.843	0.7646	0.1740	No
	design	0.769	0.8725	0.1811	No

TextRazor provides for each word or phrase a list of possibilistic entailments, by taking into account meaning of the initial word in its context and provides a confidence score for each contextual entailment. TextRazor uses DBpedia and FreeBase as knowledgbases for their text processing.

Table IV shows the top five entailments for each word in the triplet "architecture-refer_to-structure" in the columns "Entailed word" and the probability of entailment in the column " Contextual probability". According to this data, there is a probability of 27% that the meaning of the word "refer" can be conveyed by using the word "utilize". Note that TextRazor returned a large set of entailed words: 12 entailments for "architecture", 127 entailments for "refer", and 43 entailments for "structure". Also, due to different context the two usages of the word "structure" will return different probabilities for entailment variations.

D. Model reduction using Choquet integral

There are two main issues with using the TextRazor service to generate the list of entailed words. First, TextRazor uses general knowledge bases to generate the list of entailed words. This means that this list may not be directly applicable to the software development domain which is specialized and idiosyncratic. Secondly, the number of entailed words is very large. This results in increased complexity in generating the models. To address these two issues we use the Choquet integral to aggregate and retain only the most relevant knowledge in our models [22].

The discrete Choquet integral can be used as a generalized method of knowledge aggregation[22]. Consider that we have a set of entailed words $E(w) = \{w_1, w_2, ...w_n)\}$ with contextual probabilities $P_c(E(w)) = \{P_c(w_1), P_c(w_2), ..., P_c(w_n)$ The set of entailed words has the term frequency $f(E(w)) = \{f(w_1), f(w_1), ...f(w_1), \}$. These formulas represent a non-decreasing permutation of the input entailed words on $f(w)$. The Choquet integral of each entailed word $C_v(w)$ is given in the equation below.

$$C_v(w(i)) = f(w) * (P_c(x(i)) - P_c(x(i-1)) \qquad (2)$$

In this formula, x_i refers to the Choquet capacity contained in the set $\{w_1, ..., w_i\}$. As seen in this formula, this methods assigns a higher relevance to the words relevant to the community, that have higher term frequency, as well as words that ave higher entailment probability. For example, in Table IV the use of word "structure" entails the word "system" with a probability of 76.9%. However, the the word system has a higher relevance in the context of software development and consequently it will have a higher Choquet relevance. After we compute the Choquet integral of each entailed word, we reduce the set of entailed using χ^2methods on the Choquet relevance value[23]. The final decision to keep the entailed word in the result is shown in the last column of the Table IV.

E. Model construction

To construct the model, we cross-product all the entailed words in each triplet. In the example shown in Table IV will result in 36 triplets with different degrees of relevance determined by the Choquet relevance. An example of such transformation is shown in Table V for triplet "architecture - refer - structure". In total, there are 36 category triplets of which we show, due space constraints, only the ten most relevant ones. All the word in each triplet are then assigned to WordNet categories. In this example the relevance of each category triplet is computed as the average of the Choquet

TABLE V
CATEGORIES FOR THE TRIPLET "ARCHITECTURE - REFER - STRUCTURE"

Cnt	Subject category	Verb category	Object category	Relevance
1	artifact	communication	artifact	0.5160
2	artifact	stative	artifact	0.4639
3	artifact	communication	attribute	0.4380
4	artifact	cognition	artifact	0.4114
5	artifact	motion	artifact	0.3954
6	artifact	communication	cognition	0.3859
7	artifact	stative	attribute	0.3827
8	action	communication	artifact	0.3546
9	cognition	communication	artifact	0.3434
10	artifact	cognition	attribute	0.3306

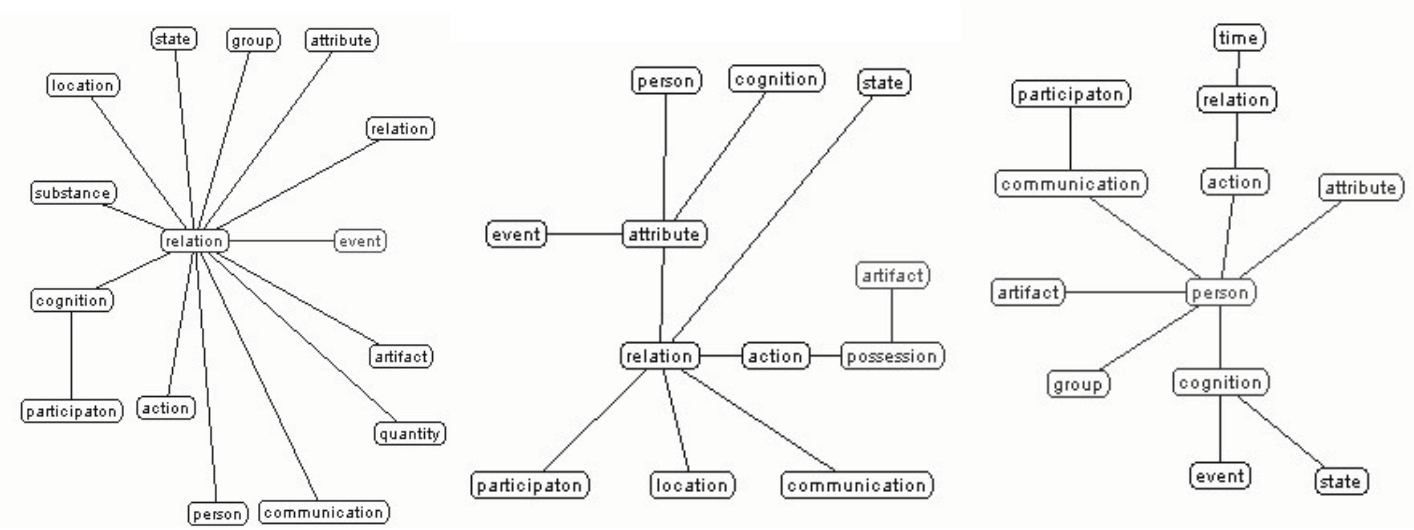


Fig. 3. Example of models in the cognition space for (a) architecture, (b) technical debt, and (c) code style

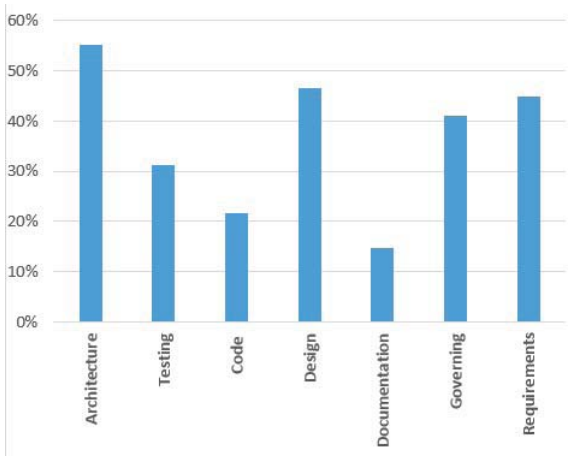


Fig. 4. Distribution of words by rank for each part of sentence type.

relevance of each word in the triplet. The model generated in this example will have four spaces that correspond to the categories of each verb: cognition, communication, motion, and state. Each of these spaces will have five dimensions that correspond to the number of unique categories of subjects and objects: action, artifact ,attribute, body, and cognition. This representation can be easily represented as a matrix and used to compute the correlation between models.

III. Experimental Results and Analysis

We constructed our models using text from IBM Developer Works [24], [25], [26], [27], [28], [29], [30], [31]. Each of these texts contain between 120 and 190 sentences that describe: architecture, code style, testing, design, governance, requirements, documentation, or technical debt. Each of the texts were converted into a model using the methodology described in this article. The models generated varied in the number of spaces used ranging from 11 spaces for architecture, to 12 spaces for documentation, and 15 spaces for design or technical debt. Figure 3 show a comparison of the models that were generated for architecture, technical debt, and code style. For a cleaner representation, these models were pre-processed using PathFinder networks [32]. As seen in this figure, the central dimension for architecture and technical debt is "relation". At the same time the "relation" dimension holds a marginal role for code style. Similarly the "person" dimension is central for code but only marginal for architecture or technical debt. From this visual representation, we can conclude that in the cognitive space, technical debt is more similar to architecture than to code style.

To evaluate the results we calculated the correlation between the "technical debt" model and all the other aspects of software development. The results are shown in Figure 4. As seen in this figure, technical debt is highly correlated with architecture, design, requirements, and governance. Technical debt is least correlated with code style, documentation, and testing. We observed that the correlation is positive for all the aspects and we attribute this to the fact that all these activities belong to the same domain and they overlap. We expect we would return a negative correlation when comparing technical debt with a subject outside software development area. We conclude that the results of our experiments are consistent with the conceptual framework described in Krutchten et al. [2].

IV. Conclusion

In this paper, we developed a model to extract domain-specific knowledge from free test. We applied this model to the software development domain to determine the relevance of a test to several aspects of software development such as architecture, code style, testing, design, governance, requirements, documentation, or technical debt. Our goal was to develop a model that would allow us to determine technical debt indirectly from all other aspects of software development. Our experiments compared the model created for technical

debt with other domain specific models and we showed that the models created are consistent the knowledge frameworks in the domain. Our future work includes much more in-depth experiments using expert-in-the-loop and application of our methodology to other areas such as education. We are also interested in including domain ontologies to replace the WordNet general framework.

REFERENCES

[1] W. Cunningham, "The wycash portfolio management system," *SIGPLAN OOPS Mess.*, vol. 4, pp. 29–30, Dec. 1992.

[2] P. Kruchten, R. L. Nord, and I. Ozkaya, "Technical debt: From metaphor to theory and practice," *IEEE Software*, vol. 29, no. 6, pp. 18–21, 2012.

[3] D. R. Dowty, *Word Meaning and Montague Grammar: The Semantics of Verbs and Times in Generative Semantics and in Montague's PTQ*, vol. 7. Springer, 1979.

[4] B. Levin, *English Verb Classes and Alternations: A Preliminary Investigation*. Chicago, IL: University of Chicago Press, 1993.

[5] G. A. Miller, "Wordnet: a lexical database for english," *Commun. ACM*, vol. 38, pp. 39–41, November 1995.

[6] S. Chambers and S. Chiaretta, "31 days of refactoring," October 2009.

[7] R. T. Fielding, *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, 2000. AAI9980887.

[8] A. Shalloway, S. Bain, K. Pugh, and A. Kolsky, *Essential Skills for the Agile Developer: A Guide to Better Programming and Design*. Addison-Wesley Professional, 1st ed., 2011.

[9] K. Seguin, *Foundations of Programming, Building Better Software*. BetterCode.com, 2008.

[10] T. Kühne and A. Child, "A functional pattern system for object-oriented design," tech. rep., 1999.

[11] W. Agresti, F. Mcgarry, D. Card, J. Page, V. Church, and R. Werking, "Managers handbook for software development." NASA Goddard Space Flight Center, 11 1990.

[12] S. Demeyer, S. Ducasse, and O. Nierstrasz, *Object Oriented Reengineering Patterns*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002.

[13] R. P. Gabriel, *Patterns of Software: Tales from the Software Community*. Oxford Univ Press, 1996.

[14] I. Marsic, *Software Engineering*. Rutgers University, 2012.

[15] "Software testing." Wikipedia, 2014.

[16] S. A. Conger, *The New Software Engineering*. Boston, MA, United States: Course Technology Press, 1st ed., 1993.

[17] J. Zobel, A. Moffat, and K. Ramamohanarao, "Inverted files versus signature files for text indexing," *ACM Trans. Database Syst.*, vol. 23, pp. 453–490, Dec. 1998.

[18] R. Krovetz, "Viewing morphology as an inference process," in *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '93, (New York, NY, USA), pp. 191–202, ACM, 1993.

[19] L. Màrquez and H. Rodríguez, *Machine Learning: ECML-98: 10th European Conference on Machine Learning Chemnitz, Germany, April 21–23, 1998 Proceedings*, ch. Part-of-speech tagging using decision trees, pp. 25–36. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998.

[20] I. Dagan, O. Glickman, and B. Magnini, "The pascal recognising textual entailment challenge," in *Proceedings of the First International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment*, MLCW'05, (Berlin, Heidelberg), pp. 177–190, Springer-Verlag, 2006.

[21] "Textrazor - extract meaning from your text." https://www.textrazor.com. Accessed: 2016-03-22.

[22] J. L. Marichal, "An axiomatic approach of the discrete choquet integral as a tool to aggregate interacting criteria," *IEEE Transactions on Fuzzy Systems*, vol. 8, pp. 800–807, Dec 2000.

[23] P. M. Bentler and K.-H. Yuan, "Tests for linear trend in the smallest eigenvalues of the correlation matrix," *Psychometrika*, vol. 63, no. 2, pp. 131–144, 1998.

[24] "What is software architecture." https://www.ibm.com/ developerworks/ rational/library/feb06/eeles/. Accessed: 2016-05-30.

[25] "Documenting software architecture." https://www.ibm.com/ developerworks/ library/ar-archdoc1/. Accessed: 2016-05-30.

[26] "Test management best practices." https://www.ibm.com/ developerworks/ rational/ library/06/1107_davis/. Accessed: 2016-05-30.

[27] "Writing clean code." https://www.ibm.com/ developerworks/ rational/ library/nov06/pollice/. Accessed: 2016-05-30.

[28] "Design debt economics. a vocabulary for describing the causes, costs, and cures for software maintainability problems." https://www.ibm.com/ developerworks/ rational/ library/ edge/09/jun09/designdebteconomics/. Accessed: 2016-05-30.

[29] "Defining program governance and structure." https://www.ibm.com/ developerworks/ rational/ library/apr05/hanford/. Accessed: 2016-05-30.

[30] "Requirements: An introduction." https://www.ibm.com/ developerworks /rational/ library/4166.html. Accessed: 2016-05-30.

[31] "Technical liability: Extending the technical debt metaphor." https://www.ibm.com/ developerworks/ community/blogs/RationalBAO/entry/technical_liability_ extending_the_ technical_ debt_ metaphor. Accessed: 2016-05-30.

[32] R. W. Schvanevelt, *Pathfinder Associative Networks: Studies in Knowledge Organization*. Norwood, NJ:Ablex, 1990.

Adopting Agile Practices in Maturity Model for Testing

Ana Paula C. C. Furtado
Informatics Center - CIn
Federal University of Pernambuco
Recife, PE, Brazil
apccf@cin.ufpe.br

Ivaldir de Farias Junior, Marcos Wanderley
SOFTEXRECIFE
Recife, PE, Brazil
{ivaldir, marcos}@recife.softex.br

Suzana Sampaio,Ermeson Andrade
Federal Rural University of Pernambuco
Recife, PE, Brazil
{suzana, ermeson}@deinfo.ufrpe.br

***Abstract*— Software engineering comprises various disciplines dedicated to preventing and repairing faults in software development. In this sense, test process improvement, is a widespread validation approach in the industry, however, software development organizations still see the discipline of software testing as a cost rather than an investment to generate benefits. Within this context, the objective of this paper is to present an experience report on the use of agile practices together with MPT.BR to support organizations that want to implement them together. As a result, we propose a group of testing agile practices that can be used together with maturity models on software testing. We conclude that agility in testing is a great field of interest and there are great amounts of contributions that can improve software quality.**

Keywords— software testing, maturity models, agile practices.

I. INTRODUCTION

In today's context of software development, given the increase in the demand for products and the reduction in the number of qualified personnel to develop them, quality is a key concept in strategies for winning a share of this market. In the broad context of software quality seen in Deming [1], Crosby [2] and Juran [3] one of the common characteristics observed is that all of these authors mention that it is essential for the specification given to be adequate and in accordance with clients' needs. Hence, testing software before delivering products to clients is one of the ways to achieve quality. According to Meyrs [4], "*software testing is the process of executing a program with the intent of finding errors*" and he recommends that this activity should be done as early as possible in the software development lifecycle. The earlier it is done, the lower is the cost to fix any faults that are found in the software. In this context, software testing is a tool that supports the development of software which, when used appropriately, adds value to the final product delivered to the customer. It is necessary to plan the introduction of test processes associated with developing software so that both can be run appropriately.

In this context, maturity models under test, such as MPT [5], TMM [6] and TMMI [7], are guides that assist organizations to introduce the essential elements for the development of the discipline of testing, given that it is not always known where to begin to define a testing process.

Agile methods, for their part, appear in the software setting as an alternative to software development which is faster and more readily adaptable to the client´s needs. According to Highsmith [8] "*agility is the ability to both create and respond to change in order to profit in a turbulent business environment; it is the ability to balance flexibility and stability*". The practices arising from this context are also instantiable for testing processes, which should be interpreted as if one can map the concepts of agility for testing activities in the software development scenario.

Therefore the aim of this paper is to present an experience report on how some process areas of MPT.BR were implemented together with agile methods based on data collected from implementing the model in 27 software engineering companies over the last 4 years.

This paper is organized as follows: the next Section gives an overview of the discipline of software testing and its main concepts. Section 3 gives the background of agile software development. Section 4 explains the MPT.BR framework, Section 5 presents the methodology used to implement MPT.BR and Section 6 brings forward an experience report on implementing MPT.BR together with agile practices in Brazilian companies. Section 6 makes concluding remarks and suggests future lines of study.

II. SOFTWARE TESTING

There have been several attempts to define activity testing, ranging from the more empirical insight test to a formal definition [4] and [9]. All statements give a general idea of the definition of software testing and essentially lead to the same overall objective of software testing which is not to find every system/software bug that exists, but to uncover situations that could negatively impact the business. Nevertheless, note that the cost of finding and fixing bugs can rise considerably during the life cycle.

It might cost 100 times more to fix a bug after the product has been released than the fix would have cost during early

development of the product [10]. That is, the earlier a bug is found, the cheaper it costs to fix. The cost of finding and fixing defects rises considerably throughout all the stages in the life cycle, as shown in Figure 1. If an error is made and the consequent defect is detected in the requirements at the specification stage, then it is relatively cheap to find and fix. This is because no rework will be necessary in later stages in the life cycle.

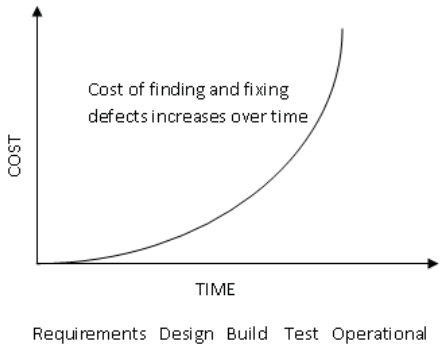


Fig. 1. Cost of defects [11]

On the other hand, if a defect is introduced in the requirement specification and it is only detected in testing or even in the operational stage then it will be much more expensive to fix, since defects in the requirements may propagate themselves into several places in the design and code. It is worth stressing that it is quite often the case that defects detected at a very late stage such as the operational stage, depending on how serious they are, will not be corrected because the cost of doing so would be extremely expensive.

This Section presented the main concepts of software testing used for consolidating the MPT.BR. The Section that follows comments on agile concepts used as reference to implement agile practices on MPT.BR.

III. AGILITY

Agile methodologies and their strategies began with the Agile Manifesto [12], in which experts united to argue that the following values should be applied to software development:

- Individuals and interactions over processes and tools;
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation; and
- Responding to change over following a plan.

One of the justifications for introducing agility into the software development process is that customers' requirements often change [13]. Agile assumes that that change is not only inevitable but also necessary to foster innovation and adaptation [14]. The dominant idea of agile development is that the team can be more effective in responding to these changes [15].

In the context of agile methodologies, it is worth describing both SCRUM [16] and Extreme Programming – XP [17]. The former is a methodology that supports the management of projects as it has the following characteristics:

- Short software development cycles, called *Sprints*;
- Constant delivery of functioning software to the client;
- Multi-disciplinary and self-manageable teams; and
- Daily meetings to monitor the team.

Extreme Programming, for its part, is a discipline of software development based on values of simplicity, communication, feedback, courage, and respect. It works by bringing the whole team together in the presence of simple practices, with enough feedback to enable the team to see where they are and to tune the practices to their unique situation.

According to Dingsoyr [18], the vast majority of published articles talk about Extreme Programming (XP), despite SCRUM being more dominant in the industry. Nevertheless, in recent years, Scrum has been gaining more and more prominence and has already surpassed other methods in annual publications. While XP focuses on agile development, Scrum focuses on agile project management. Both follow agile values and principles but each with its own focus and practices.

According to Beck [19], adopting XP is also related to transforming problems into opportunities: personal growth, deepening of relationships and improving the software produced. The attitude of simply solving the problems is not enough to reach the level of excellence in the production of software. Scrum is not a methodology, it is a framework, an iterative and incremental process, one that is simple to use when handling complex projects [20].

McConnell [21] states that, regardless of project size, some techniques are always valuable, such as: disciplined coding practices, code inspections, good tool support, reviews and use of high-level languages. These techniques are valuable for small projects and indispensable in large projects.

The discussion on the possibility of using or not using agile methods in conjunction with maturity models in software processes is frequent and current [22]. Some authors discuss the benefits of using practices from models such as CMMI to complement agile methods, especially when the size of projects grows. It is hoped to be able, in this way, to take advantage of the control and discipline of these models together with the agility and speed of response to the changes of agile methods [23], [24] and [25].

This Section presented an overview on agility and the development of the discipline together with software development processes and the next Section will present MPT.BR.

IV. MPT.BR

MPT.BR [26] uses best practices from improving the software testing process together with the software lifecycle. The main objectives of the model are:

- To become a reference model for defining and institutionalizing the testing process in organizations;
- To have continuous improvements in the software testing process talked about in accordance with the organizational objectives and desired maturity level;
- To provide a basis for assessing and, consequently, identifying the maturity level which is present in organizations; and
- To collect best practices and structure them in accordance with the levels of complexity and maturity which are associated with them.

MPT.BR consists of two components:

- *Reference model*: this document presents the main structure, the process areas and the practices of the model; and
- *Assessment guide*: this includes the assessment process and instructions on evaluating an organization based on MPT.BR.

A. Reference Model

The MPT.BR reference model presents five maturity levels, representing the stages for the evolution of a test process in the context of an organization. The maturity levels are:

1) ***Partially Managed***: this represents the first maturity level for an organization. It contains the minimal requirements that a company needs to meet in order to demonstrate that the discipline of testing is applied to projects and that this takes place in a planned and monitored manner.

2) ***Managed***: the second maturity level of testing in an organization has a broader visibility in which the scope of the project starts to be controlled by the management of change process. In addition, software testing patterns are defined and processes are monitored and controlled.

3) ***Defined***: at this level, testing becomes organizational. Defined software processes are adopted, quality assurance is institutionalized in order to support process definition, responsibilities for test organization are defined and a measurement program is institutionalized in the organization. At this level, the software testing lifecycle is associated with the development one, where static and acceptance testing are formalized and systematic procedures are applied for test closure.

4) ***Defect Prevention***: the fourth level focuses on preventing defects and systematically improving the quality of the product. At this level, the organization has a process for managing defects, in which defects found are monitored. For these defects, corrective actions are taken to prevent new defects occurring due to the same root cause. A risk analysis of the non-functional attributes of the products is made and non-functional tests are conducted to minimize such risks. It is also, at this maturity level, that an analysis is made to determine the effectiveness of the tests and to determine the quality level of the product objectively.

5) ***Automation and Optimization***: the fifth maturity level sets out to establish a process for testing that continuously improves tests and automates them. Among the characteristics of this level, it is important to mention that there is a systematic approach to automating the conduct of tests and to adopting CASE tools. The testing process is statistically controlled and undergoes continuous improvement.

Each maturity level consists of a group of process areas. A process area is a set of related practices which, when collectively implemented, satisfy a given objective. Each maturity level is also associated with a group of generic practices that need to be applied to each process area that comprises the desired maturity level. A generic practice takes into account process capabilities that need to be met by all process areas of a given maturity level.

For an organization to reach a given maturity level, it should demonstrate through the assessment, that the testing process applied in its project is in compliance with all process areas of that level together with those of the previous levels. The organization needs to demonstrate that generic practices associated with the level are also in compliance. Table I summarizes the maturity levels including MPT.BR process areas.

TABLE I. MPT.BR MATURITY LEVEL AND PROCESS AREAS

Maturity Level	Process Areas
1- Partially managed	GPT – Test Project Management PET – Test Project and Implementation
2 – Managed	GPT – Test Project Management (evolution) PET – Test Project and Implementation (evolution) GRT – Test Requirement Management
3 – Defined	GPT – Test Project Management (evolution) PET – Test Project and Implementation (evolution) FDT – Test Closure GDQ – Quality Assurance MAT – Test Measurement and Analysis OGT – Test Organization TDA – Acceptance Testing TES – Static Testing TRE – Training
4 – Defect Prevention	OGT – Test Organization (evolution) AQP – Product Quality Assessment GDD – Management of Defects TNF – Non-functional Testing
5 - Automation and Optimization	AET – Automating the conduct of Tests CEP – Statistical Control of the process GDF – Management of Tools

Process areas described as "*evolution*" mean that more requirements evolve from the previous level to the following one, these being requirements that were not mentioned before. Table II takes into account generic practices in accordance with the maturity level.

TABLE II. MPT.BR MATURITY LEVEL AND GENERIC PRACTICES

Maturity Level	Generic Practices
1- Partially managed	PG1 – Reach Defined Results PG2 – Establish Organizational Policies PG3 – Plan Process Implementation PG4 – Identify and Provide Resources PG5 – Define Authority and Responsibility PG6 – Provide Training

Maturity Level	Generic Practices
2 – Managed	PG7 – Control Work products PG8 – Monitor and Control the Process PG9 – Provide Senior Management with Visibility of the Process
3 – Defined	No extra generic practice added for this level.
4 – Defect Prevention	No extra generic practice added for this level.
5 - Automation and Optimization	No extra generic practice added for this level.

This Section presented the general structure of MPT.BR with its process areas and generic practices which are being used in the context of software development companies. The following Section will describe the methodology used to implement the model and assess the maturity on the companies throughout the last 4 years.

V. METHODOLOGY

In order to implement and assess the companies on MPT.BR, a methodology was developed to improve the efficiency in implementing the model, increase the expected results and enhance the companies' software testing processes, according to what is presented in Figure 2.

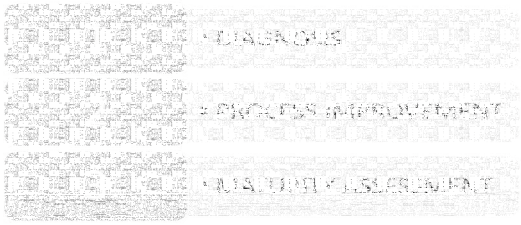


Fig. 2. Process Improvement Methodology | Source: Author

The process improvement methodology was implemented by consultants who gathered practical experience in process improvement, not only on MPT.BR but also other maturity models, following the activities in the sequence display in the methodology, according to what is detailed bellow:

1. Diagnosis: this activity is based on the analysis of the company's current situation, to understand the main problems regarding the software testing process and to plan future actions based on the improvements suggested by the maturity model.

2. Process Improvement: regarding the gaps identified in the diagnosis, a schedule of the improvements is defined and monitored by the consultant along the time in order to improve the companies' process and implement them into software projects.

3. Maturity Assessment: whenever the companies achieves a maturity in a certain level, a formal assessment is conducted by an external appraisal company in order to certify that the maturity level requirement where achieved.

After the maturity assessments were made, formal reports were published and the results on how they implemented agile practices were collected from theses documents.

Therefore, this Section described how process improvements were implemented into the software development companies in order to collect the results of how agile practices were implemented in testing process. Next Section will describe and analyze the results collected.

VI. EXPERIENCE REPORT

Software testing performed in the traditional way is conducted in a separate phase of development, shortly after completing the analysis, design and coding system. Generally, the focus of these tests is only on the graphical interface and occurs at the end of a release or at the end of the project. This is, therefore, the last (or only) sieve of quality. Note that the errors found during these tests will give feedback to the traditional development process, where steps prior to testing (e.g., design, analysis and coding) need to be performed again, and only then will the test be performed. However, if this feedback is performed more than once, it may have a significant impact on the scope, time and cost of the project, which has a direct impact on the quality of the system being developed.

Agile methodologies, such as XP or Scrum are increasingly being adopted by software development companies around the world since they enable results to be obtained in the early stages of software development and are able to add value to the client from the first iteration. For the implementation of agile techniques in the testing environment what happens is that the test occurs on each increment to the product (e.g. a new functionality, increased code etc.) as soon as it is available, and not only when the entire product is concluded. This is, unlike the traditional test, depending on the product that is being developed, tests are performed and nonconformities found are corrected immediately.

A. AGILE PRACTICES IN MPT.BR

Testing maturity models, such as MPT.BR aim to improve the testing process using best practices related to the activities throughout the test life cycle of the product. However, what is still not clearly defined is how test practices should be embedded in the context of agile methods. Thus, this Section describes an experience report based on the experience of implementing MPT.BR in various companies all over Brazil, in which agile practices were adopted in the testing environment. Table III shows the mapping of the process areas (including the practices of each area) in agile implementations adopted in the context of MPT.BR.

TABLE III. AGILE IMPLEMENTATION IN MPT.BR

Process Area	*Practices*	*Agile Implementation*
GPT	GPT4	Scrum Taskboard, Kanban Board, Sprint Backlog, Improvement Backlog
	GPT5	Planning Poker, Ideal Days, Relative Sizing
	GPT6	Short Iterations, Sprints
	GPT9	Daily Meetings to Identify Risks
	GPT13	Agile Metrics, such as Sprint Burndown Chart

Process Area	*Practices*	*Agile Implementation*
	GPT16, GPT18	Daily Meetings, Sprint Review, Retrospective
	GPT17	Continuous Feedback, Client Collaboration
	GPT19, GPT20	Daily Meetings
PET	PET1, PET2	Test Driven Development – TDD Behavior Driven Development - BDD
GRT	GRT1	User Stories, Backlog Item
FDT	FDT3	Sprint Review Restrospective
GDQ	GDQ2	Daily Meetings
MAT	MAT1	Agile Metrics, such as Sprint Burndown Chart
	MAT4	Daily Meetings, Restrospective
TES	TES3	Pair Programming Peer Review
	TES4	Daily Meetings
GDD	GDD1	Daily Meetings, Retrospective
AET	AET1 AET2 AET3	Test Driven Development – TDD Behavior Driven Development - BDD

The details of how agile practices were mapped to the process areas can be observed as described below:

- GPT: agile practices were used to introduce the concept of test sprints, in which the requirements to be tested are arranged in backlog and made visually available using Scrum boards (or Kanban Boards). Moreover, planning poker, relative sizing and ideal days can be used as techniques for estimating the size of the stories to be tested, particularly as a metric so as to construct Sprint Burndown, Team Velocity, Lead Time, etc. Daily Meetings were introduced to monitor the project and as a mechanism to identify and stay abreast of project risks.
- PET: Test driven development (TDD) or Behavior-driven development (BDD) are techniques that could be used to identify the project's test cases and satisfy the demand of the process area.
- GRT: instead of formal requirements, the scope of projects could be organized using user stories that are part of the project backlog.
- FDT: During the Sprint Review, the tested items can be packaged so they can be delivered and the test environment can be clean, thus satisfying part of what the test closure process area requires. In addition, the practice of retrospective adds on the lessons learned, thus bringing an implemented agile practice to FDT.
- GDQ: for this process area, the practice of daily meetings can be implemented to include the reporting of items on the quality of the project.
- MAT: the agile metrics suggested by SCRUM, such as Burndown, Velocity, and Lead Time can be used as an option for the indicators of the test project. In addition, the Daily Meetings and Retrospectives can be used to report on these results.
- TES: the static test can be conducted by pair programming, where not only the revision of the code developed is observed but also the dissemination of knowledge. In addition, the daily meetings can be used as a moment to analyze the data from the reviews and to standardize communication with the team.
- GDD: the daily meetings and retrospectives can also be used to identify the root causes of the defects found.
- AET: the test can be automated based on the BDD and TDD techniques, besides which automating the test itself is already considered the introduction of agile design practices.

Therefore, this section has presented the agile techniques and practices that have been implemented in MPT.BR so far. The Following Section will present the number of agile implementations compared to implementations that did not use such an approach.

B. MPT.BR AGILE CERTIFIED ORGANIZATIONS

Based on the results of the current implementations, the consolidated situation is that, of a total of 16 existing process areas in MPT.BR, so far 10 have been implemented using agile methodologies, i.e. 63% of the model has already been instantiated in an agile way. These data were obtained by analyzing the implementation of MPT.BR in the 27 companies that have been evaluated between 2010 and 2014.

Table IV summarizes the total number of companies evaluated, at their respective levels, and indicates which of them do or do not use agile methods in their testing processes. Based on these numbers, it can be observed that at all levels of companies evaluated until now, MPT has been implemented both by using the traditional approach and agile methods. Level 4 was not evaluated because two companies have opted to go directly to the 5th maturity level.

TABLE IV. NUMBER OF ORGANIZATIONS THAT USE AGILE PRACTICES

Number of Organizations			*Maturity Level*
Agile	*Traditional*	*Total*	
9	3	12	1
4	3	7	2
1	5	6	3
0	0	0	4
2	0	2	5
27 Organizations			**Total**

Moreover, the number of implementations with agile methodologies (16) is greater than the number of companies that have implemented it using the traditional method (11), which is another indication that the current trend of software development is to use agile methodologies. The largest number of evaluations so far is at the first level of maturity and 75% of companies have also used agile methods. It can be observed that of the companies that use agility, just as many are small as are medium-sized or large, which makes us infer that the use of agile methods is not related to the size of the organization.

In general, some positive points were observed starting with using agile methods, such as:

- Increase in the visibility of the process of testing engineering;
- Integration of the development and testing teams;
- Ease of understanding and adapting the company´s processes so as to make them fit for MPT.BR;
- Agility in planning the tests in the project;
- The involvement of testers in the planning activities anticipated the planning of testing and improving the definition of the acceptance criteria of the project ; and
- The tests stopped being made by the developers themselves and the role of the tester was defined in the project.

A negative aspect of the introduction of agile methodologies, which goes out of its way to extol the simplicity of the process and the reduction in the maximum number of artifacts generated during the process, was the difficulty in generating evidence for companies to assess in MPT.BR. As the evaluations are made based on the products generated from projects that use the maturity model, sometimes some artifacts were produced more because of the need to make an evaluation than because of the company's need to have it in its process. The implementers of the model and companies face a paradox between maintaining the agility requirements of a process and adhering to the maturity model and being ready for a formal evaluation.

Therefore, this Section has presented some of the most important aspects concerning the introduction of agile methodologies into a maturity model for software testing. The following Section will present the conclusions obtained from this study as well as future research studies that could be usefully carried out.

VII. Conclusion and Future Research studies

This article was constructed to present the testing maturity model - MPT.BR together with the concepts of testing that were used as the basis for designing it. Moreover, it also presented a theoretical framework for agile methodologies and how these methodologies were instantiated in conjunction with a testing maturity model.

The use of agile methods has been observed as a trend in the area of software development, and this can also be observed when the aspect of software development is directly related to the testing processes of a given organization. The instantiation of the concepts of the agile world for the aspects of the discipline of testing was a demand that came from the Information Technology market and a challenge for the group of implementers of the testing maturity model.

From the data collected from the assessments of the MPT.BR from January 2010 until April 2014, it was observed that most of the companies evaluated made use of agile methodologies when instantiating their processes. It was also observed that the use of agile methods in conjunction with the testing maturity model is not restricted to how large or small a company is because it was conceived in small, medium and large companies.

Therefore, based on the results obtained so far, it is predicted that this study can be complemented by the following future studies:

- Seeking ways to conduct Non-Functional Testing of agile ways to support the implementation of the TNF - Non Functional Testing process area;
- Seeking agile practices to carry out the process area of CEP - Statistical Control of Processes;
- Enhancing the automation of the tests over all process areas and maturity levels of MPT.BR in order to maximize the results obtained;
- Understanding the reasons that lead an organization to choosing either an implementation with a traditional approach or an agile approach; and
- Analyzing the possibilities of evaluating the maturity of companies which use agile practices without requiring documents to be drawn up that are constructed only to prove a given practice has been carried out and which is performed in a lighter way in everyday life.

References

[1] E. Deming (1989) Calidad, Productividad y Competitividad. La salida de la Crisis - Ediciones Díaz de Santos, S.A., Madrid.

[2] P. Crosby (1988). The eternally successful New York: Times Books. organization. New York: McGraw-Hill.,

[3] J. Juran and A. Blanton (1999). Juran's Quality Handbook. McGraw Hill - New York.

[4] J. Glenford Myers (1979). "The Art of Software Testing," John Wiley and Sons, ISBN 0-471-04328-1.

[5] Softex Recife (2011) MPT - Melhoria do Processo de Teste. Available at http://mpt.org.br/mpt/wp-content/uploads/2013/05/MPT_Guia_de_referencia.pdf captured 27/04/2014

[6] E. Veenendaal and R. Swinkels (2002) Guidelines for Testing Maturity. Available at http://goo.gl/0n5d3V captured 26/04/2014.

[7] E. Veenendaal(2012). Test Maturity Model Integration Release 1.0. TMMi Foundation, Ireland. Available at www.tmmifoundation.org/downloads/TMMi/TMMi%20Framework.pdf captured 26/04/2014.

[8] J. Highsmith (2004). Agile Project Management - Creating Innovative Products. Pearson Education.

[9] B. Hetzel (1988). The Complete Guide to Software Testing - Second Edition, John Wiley & Sons.

[10] H. Brian, P. Morgan, A. Samaroo, G. Thompson and P. Williams (2010). "Software Testing – An ISTQB-ISEB Foundation Guide",British Informatics Society Limited.

[11] D. Graham, E. Veenendaal, I. Evans, R. Black (2008). "Foundations of Software Testing: ISTQB Certification," Intl Thomson Business Pr.

[12] J. Highsmith (2001). Jim Highsmith & Martin Fowler. The Agile Manifesto. Software Development Magazine, vol. 9, no. 8, pp. 29–30.

[13] K. Reed, E. Damiani, G. Gianini, A. Colombo (2004). Agile management of uncertain requirements via gen- eralizations: a case study. In QUTE-SWAP '04: Proceedings of the 2004 workshop on Quantitative techniques for software agile process, pp. 40–45, New York, NY, USA, ACM.

[14] V. Vinekar, C. Slinkman and S. Nerur (2006). Can agile and traditional systems development approaches coexist? an ambidextrous view. IS Management, 23(3):31–42.

[15] A. Cockburn, J. Highsmith (2001). Agile Software Development: The People Factor. Computer, vol. 34, no. 11, pp. 131–133.

[16] K. Schwaber and J. Sutherland (2013). The Definitive Guide to Scrum: The Rules of the Game. Available at http://goo.gl/mDyLcM. Captured 26/04/2014.

[17] F. Maurer and S. Martel (2002) Extreme Programming. Available at http://cf.agilealliance.org/articles/system/article/file/1026/file.pdf Captured 26/04/2014

[18] T. Dingsøyr (2012). A decade of agile methodologies: Towards explaining agile software development. J. Syst. Softw. 85, 6 , 1213--1221.

[19] K. Beck (2004). Programação eXtrema aplicada: Acolha as Mudancas.. Bookman.

[20] K. Schwaber (2004). Agile project management with scrum. Microsoft Press, Redmond, WA, USA.

[21] S. McConnell (2004). Code Complete. Microsoft Press

[22] J. Boria, V. Rubinstein, A. Rubinstein. (2013). A História da Tahini-Tahini: Melhoria de Processos de Software com Métodos Ágeis e Modelo MPS" , SBQS 2013, SEPIN.

[23] C. Jakobsen., K. Johnson (2008). Mature Agile with a Twist of CMMI. AGILE '08 Proceedings of the Agile 2008 (pp. 212--217). IEEE Computer Society Washington, DC, USA.

[24] V. Mahnic,N. Zabkar (2008). Measurement repository for Scrum-based software development process. CEA'08 Proceedings of the 2nd WSEAS International Conference on Computer Engineering and Applications (pp. 23--28).

[25] C. Jakobsen, J. Sutherland (2009). Scrum and CMMI ± Going from Good to Great. Agile Conference, 2009. AGILE'09. (pp. 333--337). IEEE Computer Society Washington, DC, USA.

[26] A. Furtado, M. Gomes, E. Andrade, I. de Farias Junior (2012). MPT.BR: A Brazilian Maturity Model for Testing Published in: Quality Software (QSIC).