SOFTENG 2015

The First International Conference on Advances and Trends in Software Engineering

ISBN: 978-1-61208-449-7

April 19 - 24, 2015

Barcelona, Spain

SOFTENG 2015 Editors

Felipe Silva Ferraz, CIN/UFPE / CESAR, Brazil
Jameleddine Hassine, KFUPM, Saudi Arabia

SOFTENG 2015

Foreword

The First International Conference on Advances and Trends in Software Engineering (SOFTENG 2015), held between April 19th-24th, 2015 in Barcelona, Spain focuses on challenging aspects for software development and deployment, across the whole life-cycle.

Software engineering exhibits challenging dimensions in the light of new applications, devices and services. Mobility, user-centric development, smart-devices, e-services, ambient environments, e-health and wearable/implantable devices pose specific challenges for specifying software requirements and developing reliable and safe software. Specific software interfaces, agile organization and software dependability require particular approaches for software security, maintainability, and sustainability.

We take here the opportunity to warmly thank all the members of the SOFTENG 2015 Technical Program Committee, as well as the numerous reviewers. The creation of such a high quality conference program would not have been possible without their involvement. We also kindly thank all the authors who dedicated much of their time and efforts to contribute to SOFTENG 2015. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

Also, this event could not have been a reality without the support of many individuals, organizations, and sponsors. We are grateful to the members of the SOFTENG 2015 organizing committee for their help in handling the logistics and for their work to make this professional meeting a success.

We hope that SOFTENG 2015 was a successful international forum for the exchange of ideas and results between academia and industry and for the promotion of progress in the field of software engineering.

We are convinced that the participants found the event useful and communications very open. We hope Barcelona provided a pleasant environment during the conference and everyone saved some time for exploring this beautiful city.

SOFTENG 2015 Advisory Chairs:

Alain Abran, University of Québec, Canada
Mira Kajko-Mattsson, Stockholm University & Royal Institute of Technology, Sweden
Paolo Maresca, Amadeus IT Group, France
Patricia McQuaid, California Polytechnic State University, USA

SOFTENG Research Liaison Committee
Veena Mendiratta, Bell Labs, Alcatel-Lucent, USA
Michael Perscheid, SAP Innovation Center Potsdam, Germany
Doo-Hwan Bae, Software Process Improvement Center - KAIST, South Korea
Fergal Mc Caffery, Dundalk Institute of Technology, Ireland
Ron Watro, BBN Technologies, USA

SOFTENG 2015

Committee

SOFTENG Advisory Committee

Alain Abran, University of Québec, Canada
Mira Kajko-Mattsson, Stockholm University & Royal Institute of Technology, Sweden
Paolo Maresca, Amadeus IT Group, France
Patricia McQuaid, California Polytechnic State University, USA

SOFTENG Research Liaison Committee

Veena Mendiratta, Bell Labs, Alcatel-Lucent, USA
Michael Perscheid, SAP Innovation Center Potsdam, Germany
Doo-Hwan Bae, Software Process Improvement Center - KAIST, South Korea
Fergal Mc Caffery, Dundalk Institute of Technology, Ireland
Ron Watro, BBN Technologies, USA

SOFTENG Industrial Liaison Committee

Tomas Schweigert, SQS Software Quality Systems AG, Germany
Janne Järvinen, F-Secure Corporation - Helsinki, Finland

SOFTENG 2015 Technical Program Committee

Haider Abbas, Center of Excellence in Information Assurance - King Saud University, Saudi Arabia
Alain Abran, University of Québec, Canada
Magnus Almgren, Chalmers University of Technology, Sweden
Étienne André, Université Paris 13, France
Jocelyn Aubert, Luxembourg Institute of Science and Technology (LIST), Luxembourg
Doo-Hwan Bae, Software Process Improvement Center - KAIST, South Korea
Alessandra Bagnato, SOFTEAM R&D Department, France
Byoungju Choi, Ewha Women's University, South Korea
Sunita Devnani Chulani, Cisco Systems, USA
Paolo Ciancarini, University of Bologna, Italy
Christof Ebert, Vector Consulting Services GmbH, Germany
Mahmoud O. Elish, King Fahd University of Petroleum and Minerals, Saudi Arabia
Anita Finnegan, Dundalk Institute of Technology, Ireland
Francesco Flammini, Ansaldo STS, Italy
Sibylle Fröschle, OFFIS & University of Oldenburg, Germany
Barbara Gallina, Mälardalen University, Sweden
Alessia Garofalo, University of Naples "Parthenope", Italy
Ibrahim Habli, University of York, UK
Qiang (Nathan) He, Swinburne University of Technology, Australia

Guowei Yang, Texas State University, USA
Cemal Yilmaz, Sabanci University, Turkey
Mansooreh Zahedi, IT University of Copenhagen, Denmark

Table of Contents

A Change Impact Analysis Approach to GRL Models

Jameleddine Hassine
Department of Information and Computer Science
King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia
Email: jhassine@kfupm.edu.sa

***Abstract*—Goal models represent interests, intentions, and strategies of different stakeholders in early requirements elicitation process. In a socio-technical context, goal models evolve quickly to accommodate the rapid changes of stakeholders' needs, technologies, and business environments. In order to control and minimize the risks brought by requirements changes, it is important to analyze the effects of modifications in goal models. Given a proposed modification, Change Impact Analysis (CIA) allows for the identification of software artifacts that will be impacted and for the estimation of the effort required to implement the proposed changes. This paper describes an approach to analyze the propagation of changes in goal models specified using the Goal-oriented Requirements Language (GRL). Dependencies in the GRL model are first extracted and illustrated using a GRL Model Dependency Graph (GMDG), describing inter- and intra- actor dependencies. In order to identify model constructs that are impacted by a proposed change, we apply the well-known technique of program slicing to the GMDG model. We illustrate our approach by applying it to a goal model describing undergraduate students' involvement in research activities.**

***Keywords*-Goal models; requirements; Change Impact Analysis; Goal-oriented language(GRL); GRL Model Dependency Graph (GMDG); slicing;**

I. INTRODUCTION

Evolving customer needs is one of the driving factors in software development. Requirements models are the first available artifacts during the software development process. They undergo many changes caused by changing user requirements and business goals or induced by changes in implementation technologies. Hence, there is a need to analyze the impact of requirements changes in order to help detect and solve possible conflicts between stakeholders and to assess the different design alternatives influenced by these changes. Localizing the impact of changes is one of the most efficient strategies for a successful evolution. Change Impact Analysis (CIA) [1] aims at identifying the potential consequences of a change, and estimating what needs to be modified to accomplish a change [1]. Hence, change impact analysis is required for constantly evolving systems to support the comprehension, implementation, and evolution of changes. Change impact analysis has been applied to analyze source code, formal models (architectural and requirements models), or other artifacts (e.g., documents, data sources, configuration files) [2].

Goal models have been introduced as a means to ensure that stakeholders' interests and expectations are met in the early requirements engineering stages. As goal models gain in complexity (e.g., large systems involving many stakeholders and many dependencies), they become difficult to comprehend and to maintain. The main motivation of this research is to manage the complexity of goal models with respect to maintenance tasks. In particular, we are interested in applying change impact analysis techniques to Goal-oriented Requirement Language (GRL) [3]. This paper aims to:

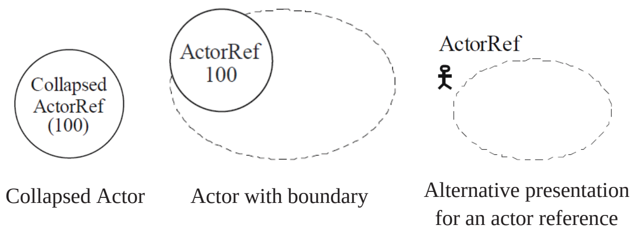
- Provide a GRL-based approach to change impact analysis. The proposed approach allows requirements engineers and projects leaders to ask "what if ...?" questions, and to reason about alternative scenarios with respect to maintain GRL models.
- Provide an insight into how changes propagate within an actor or how they spread across many actors. In fact, dependencies that are embedded within the GRL model are extracted and described as a GRL Model Dependency Graph (GMDG). Dependencies between intentional elements that are within one GRL actor are referred to as intra-actor dependencies, while dependencies involving different actors are referred to as inter-actor dependencies. In case of a modification of the GRL model, it is important to identify whether the changes strech across the actor boundary. This would allow to choosing the correct validation approach (e.g., mediation process or a less formal discussions between the intervening stakeholders).
- Extend the use of the well-known technique of program slicing [4] to goal models. In contrast to traditional program slicing, we apply slicing to GMDG graphs. As a result, the GRL artifacts that are impacted by a given change are identified.

The remainder of this paper is organized as follows. In the following section, we provide a brief introduction to the GRL [3] language. Our proposed change impact analysis approach is presented in Section III. In Section IV, we apply our proposed approach to a GRL model describing the involvement of undergraduate students and professors in research activities. A comparison with related work is provided in Section V. Finally, conclusions and future work are presented in Section VI.

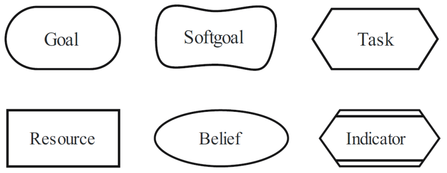
II. GOAL-ORIENTED REQUIREMENTS LANGUAGE (GRL)

GRL [3] is a visual notation used to model stakeholders' intentions, business goals, and non-functional requirements (NFR).

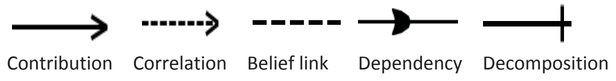
The basic notational elements of GRL are summarized in Figure 1. Actors (see Figure 1(a)) are holders of intentions; they are the active entities in the system or its environment who want goals to be achieved, tasks to be performed, resources to be available, and softgoals to be satisfied. Actor definitions are often used to represent stakeholders as well as systems. A GRL actor may contain intentional elements describing its intentions and capabilities. Figure 1(b) illustrates the GRL intentional elements (i.e., goal, task, softgoal, resource and belief) that optionally reside within an actor.



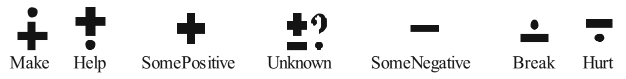
(a) GRL actors



(b) GRL intentional elements



(c) GRL links



(d) GRL qualitative contribution types

Fig. 1. GRL notational elements

Figure 1(c) illustrates the various kinds of links in a goal model. Decomposition links allow an intentional element to be decomposed into sub-elements (using AND, OR, or XOR). Beliefs, used to represent rationales from model creators, are connected to GRL intentional elements using belief links. Contribution links indicate desired impacts of one intentional element on another element. A contribution link has a qualitative contribution type (see Figure 1(d)) and/or a quantitative contribution (an integer value between -100 and 100 in standard GRL). Correlation links are similar to contribution links but describe side effects rather than desired impacts. Finally, dependency links model relationships between actors and between intentional elements (within the same actor). In addition, GRL defines *indicators* as containable elements used to analyze GRL models based on real-world measurements. Since, indicators are used only in converting real-world values into satisfaction levels, they are out of the scope of this research. For a detailed description of the GRL language, the reader is invited to consult the URN standard [3].

III. GRL Change Impact Analysis Approach

In what follows, we present our GRL-based change impact analysis approach.

A. Dependencies in GRL Models

Goal-oriented models describe the actors within a complex socio-technical system, dependencies between system elements, and organizational goals. Dependencies enable reasoning about how actors/elements depend on each other to achieve the planned goals. Dependencies in GRL models can be classified as explicit or implicit [5]. Explicit dependencies are modeled as dependency links $\rightarrowtail$, while implicit dependencies are modeled using contributions $\longrightarrow$, correlations $\dashrightarrow$, and decompositions $\longrightarrow\!\!\!\!|$ [5]. For instance, in Figure 2(a), the satisfaction of *Goal1* depends on the satisfaction of *Goal2* and the type of the contribution *ContributionG2G1*. A change to the satisfaction level of *Goal2* or to the contribution's qualitative/quantitative type/level, will have a direct impact on the satisfaction of *Goal1*. Explicit dependency links can be used in many types of configurations according to the required level of detail [3]. GRL actors (described as collapsed actors) can be used as source and/or destination of an explicit dependency link. Intentional elements inside actor definitions can be used as source and/or destination of a dependency link. Collapsed actors (see Figure 1(a)) are used when an actor is the source/destination of an explicit dependency. In addition, actors cannot be used as source or destination of a contribution, a correlation, or a decomposition link. We further classify dependencies as:

- *Intra-actor* dependencies: describes a dependency (explicit or implicit) having its source and target within the same actor boundary.
- *Inter-actor* dependencies: describes a dependency (explicit or implicit) having its source and target bound to different actor definitions.

It is worth noting that GRL syntax does not allow actors to overlap (i.e., share common GRL elements).

B. GRL Model Dependency Graph (GMDG)

Before we define the GMDG graph, we define formally GRL models.

Definition 1 (GRL Model): We assume that a GRL model *GRLM* is denoted by a 3-tuple (or a triple): (Actors, Elements, Links), where:

- *Actors* is the set of actors in the GRL model. It contains a set of actor references and a set of collapsed actors, denoted by *CollapsedActors*.
- *Elements* is the set of intentional elements (i.e., tasks, goals, softgoals, resources, and beliefs) in the GRL model.
- *Links* is the set of links in the model. A GRL link is defined as a triple (*type*, *src*, *dest*), where *type* is the link type (of type *LinkTypes* = {contribution, correlation,

dependency, decomposition}), *src* and *dest* are the source and destination of the link, respectively (they are either part of *Elements* or *CollapsedActors*).

Definition 2 (Links Access Functions): We define the following access functions over the set of links (i.e., *Links*):

- *TypeLink*: *Links* → *LinkTypes*, returns the type of the link.
- *Source*: *Links* → *Elements* ∪ *CollapsedActors*, returns the intentional element/actor source of the link.
- *Destination*: *Links* → *Elements* ∪ *CollapsedActors*, returns the intentional element/actor destination of the link.

For example, given a contribution *C*=(*contribution*, *Goal2*, *Goal1*) (see Figure 2(a)), then *TypeLink*(*C*)= *contribution*, *Source*(*C*)= *Goal2*, and *Destination*(*C*)= *Goal1*.

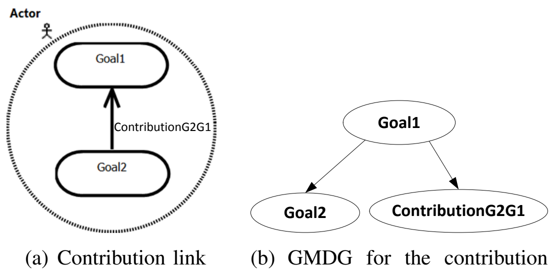
Using the dependency definitions given above, we can now define the GRL Model Dependency Graph (GMDG).

Definition 3 (GRL Model Dependency Graph (GMDG)): A GRL Model Dependency Graph (GMDG) is a directed, connected graph defined as GMDG=(N, E), consisting of:

- *N* is a set of nodes. Each GRL intentional element (i.e., of type *Elements*), each link (i.e., of type *LinkTypes*), and each collapsed actor is mapped to a node *n*∈*N*.
- *E* is a set of edges. An edge *e*∈*E* represents a dependency link (intra- or inter- actor dependency).

C. Constructing the GMDG Graph

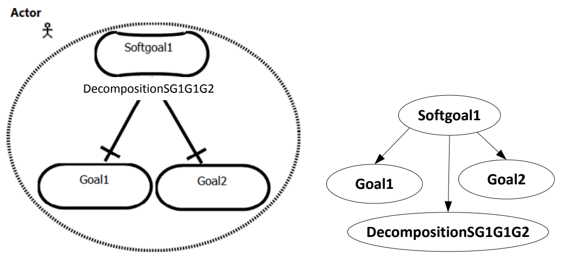
In a GMDG graph, intra-actor dependencies are illustrated as solid arrows (see Figure 2(b)), while inter-actor dependencies are illustrated as dashed arrows (see Figure 4(b)). Figure 2(b) illustrates the GMDG corresponding to a single contribution link (Figure 2(a)). GMDG nodes are created for *Goal1*, *Goal2*, and the contribution link *ContributionG2G1*. Two intra-actor dependency links are created from node *Goal1* to both *Goal2* and *ContributionG2G1* nodes.



(a) Contribution link (b) GMDG for the contribution link

Fig. 2. Contribution relationship and its Associated GMDG

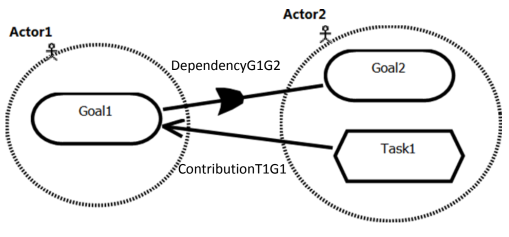
In case of a decomposition relationship (see Figure 3(a)), each intentional element is mapped to a GMDG node (i.e., *Softgoal1*, *Goal1*, *Goal2*). The decomposition is mapped to one single node (i.e., *DecompositionSG1G1G2*). Intra-actor dependencies are created between *Softgoal1* and *Goal1*, *Goal2*, and *DecompositionSG1G1G2* (see Figure 3(b)).



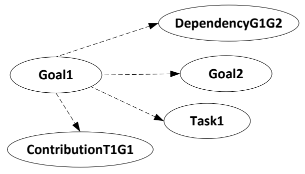
(a) Decomposition Relationship (b) GMDG for the Decomposition Relationship

Fig. 3. Decomposition relationship and its associated GMDG

Figure 4(a) illustrates a GRL having two actors (i.e., *Actor1* and *Actor2*) and two inter-actor dependencies (i.e., *DependencyG1G2* and *ContributionT1G1*). Explicit dependencies are mapped to GMDG nodes as follows: nodes are created to map the depender (e.g., *Goal1*), the dependee (e.g., *Goal2*), and the dependency link (e.g., *DependencyG1G2*). Two inter-actor dependency links are created from node *Goal1* to *Goal2* and *DependencyG1G2* nodes. The contribution link is mapped to two inter-actor dependencies, one between *Goal1* and *Task1*, and the other between *Goal1* and ContributionT1G1 as shown in Figure 4(b).



(a) Inter-actor Dependencies



(b) GMDG with Inter-actor Dependencies

Fig. 4. Inter-actor dependencies and their associated GMDG

It is worth noting that beliefs are always associated with one single intentional element. Consequently, we don't create a separate GMDG node for representing a belief. The algorithm in Figure 5 is used to construct the GMDG graph.

D. Slicing the GRL Model Dependency Graph

The notion of program slicing was originated in the seminal paper by Weiser [4]. Weiser defined a slice *S* as a reduced independent program guaranteed to faithfully represent the original program within the domain of the specified subset of behavior. Informally, a static program slice consists of those parts of a program that potentially could affect/affected by

```
Input  : A GRL Model
Output: A GMDG Graph
foreach element e ∈ Elements ∪ CollapsedActors do CreateNode(e);
foreach link l ∈ Links do
    if (TypeLink(l)=contribution or TypeLink(l)=correlation or
    TypeLink(l)=dependency then
        CreateNode(l);
        if (Source(l) and Destination(l) are within the same actor)
        then
            Create an intra-actor dependency link from Destination(l)
            to Source(l);
            Create an intra-actor dependency link from Destination(l)
            to l;
        else
            Create an inter-actor dependency link from Destination(l)
            to Source(l);
            Create an inter-actor dependency link from Destination(l)
            to l;
        end
    end
    if (TypeLink(l)=decomposition) then
        if (no node is created for l) then
            CreateNode(l);
        end
        if (Source(l) and Destination(l) are within the same actor)
        then
            Create an intra-actor dependency link from Destination(l)
            to Source(l);
        else
            Create an inter-actor dependency link from Destination(l)
            to Source(l);
        end
    end
end
```

Fig. 5. Constructing the GMDG graph

the value of a variable V at a point of interest, called the *slicing criterion*. The application of slicing has been extended to other software artifacts such as requirements and design models, formal specifications, software architectures, etc. In what follows, we extend the application of slicing to GRL models. In our CIA approach, we are interested in obtaining the GRL elements that are impacted by a given modification represented by the *slicing criterion*.

Definition 4 (GRL Slicing Criterion): Given a GRL model, a slicing criterion may be either a GRL intentional element, a GRL link, or a collapsed actor.

The proposed slicing algorithm (see Figure 6) is based on a backward traversal of the GMDG. It starts with the localization of the node corresponding to the slicing criteria. Next, it follows all incoming links and marks all encountered nodes. The procedure stops when all reached nodes have no incoming edges. Encountering inter-actor dependency links is an indication that the proposed change would affect other actors.

The GMDG marked nodes are then mapped back to the original GRL model, representing all GRL constructs impacted by the proposed change.

```
Input  : A GMDG + a slicing criterion (SC)
Output: A set of marked GMDG nodes + whether the impact is spread
        to other actors (ImpactingOtherActors)
Locate SC;
Mark node SC;
currentNode= SC;
ImpactingOtherActors = false;
forall the incoming links to SC do
    if (incoming link is inter-actor) then
        ImpactingOtherActors = true;
    end
    Follow the link;
    Mark the reached node;
    currentNode ⟵ reached node;
    recursively call the algorithm with GMDG and currentNode as
    input
end
```

Fig. 6. GMDG Slicing Algorithm

IV. Case Study: Undergraduate Student Involvement in Research Activities

In this section, we apply our proposed CIA approach to a GRL model describing undergraduate student involvement in research activities (see Figure 7). The model involves two actors (Professor and Undergraduate Student) and describes one explicit dependency stating that “In order to achieve student satisfaction with their participation in university research projects, undergraduate students depend on professors to ensure active involvement of students in their research projects”. Research opportunities may take one of the following two forms: (1) programming duties and (2) experiments and data collection. These duties are described as two professor tasks (i.e., “Assign programming duties to undergraduate students” and “Assign experiments and data collection to undergraduate students”) contributing positively (i.e., using two GRL *help* contributions) to the softgoal “Active involvement of undergraduate students in research projects”. The latter softgoal hurts (i.e., using the *hurt* contribution type) the completion of critical projects having tight deadlines. Student satisfaction with research activities is modeled as a softgoal “Students satisfied with their participation in university research projects” and it is subject to getting an academic credit for their performed tasks (i.e., goal “Receive academic credits for research activities”) and getting a financial compensation (i.e., goal “Receive financial compensation for research activities”) from professors.

Figure 8 illustrates three changes to be implemented in the original GRL model:

1) *Change1*: Addition of a new task “Assign mechanical and electrical assembly tasks to undergraduate students” contributing positively (i.e., *SomePositive* type) to the softgoal “Active involvement of undergraduate students in research projects”.
2) *Change2:* Replace the *And* by an *Or* decomposition.
3) *Change3:* Deletion of task “Assign programming duties to undergraduate students” and its corresponding *help* contribution.

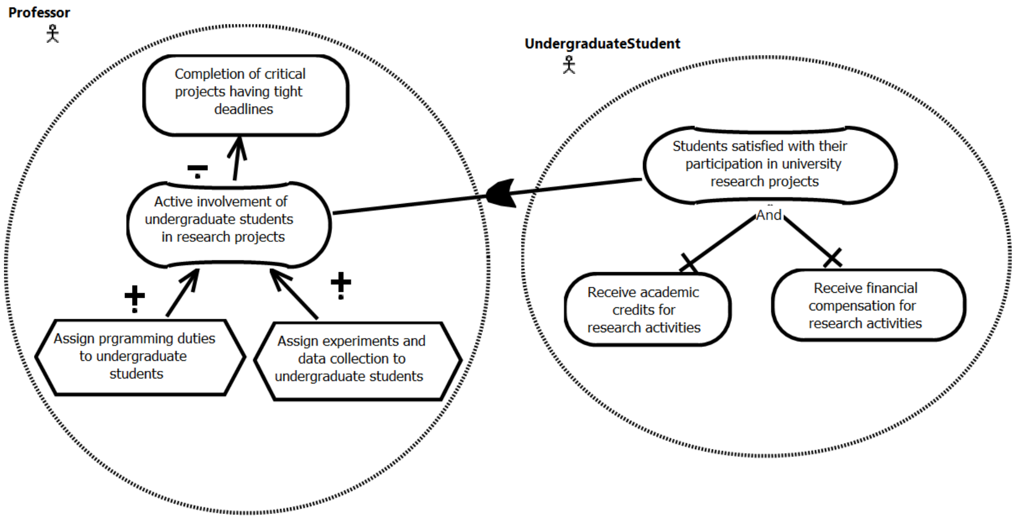
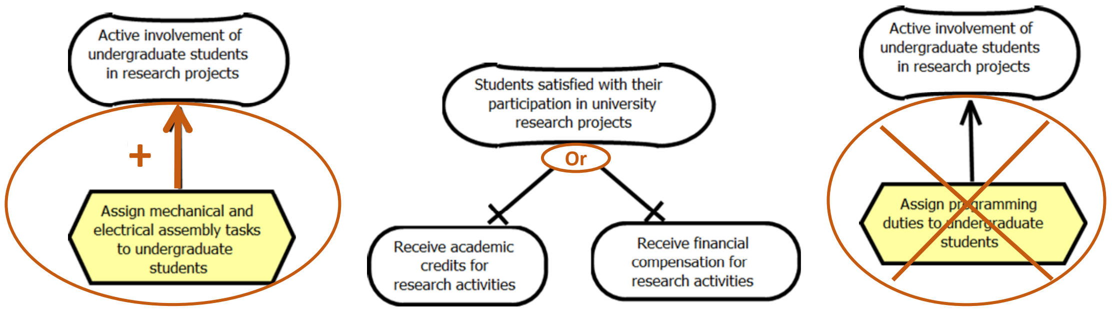

Professor
UndergraduateStudent
Completion of critical projects having tight deadlines
Active involvement of undergraduate students in research projects
Assign prgramming duties to undergraduate students
Assign experiments and data collection to undergraduate students
Students satisfied with their participation in university research projects
And
Receive academic credits for research activities
Receive financial compensation for research activities
+
+

Fig. 7. GRL model describing undergraduate student involvement in research activities


Active involvement of undergraduate students in research projects
+
Assign mechanical and electrical assembly tasks to undergraduate students
Students satisfied with their participation in university research projects
Or
Receive academic credits for research activities
Receive financial compensation for research activities
Active involvement of undergraduate students in research projects
Assign programming duties to undergraduate students

(a) Change1: Add a new task and a new help contribution

(b) Change2: Replace the *And* by an *OR* decomposition

(c) Change3: Delete a task and its corresponding contribution

Fig. 8. Three planned changes to the GRL model

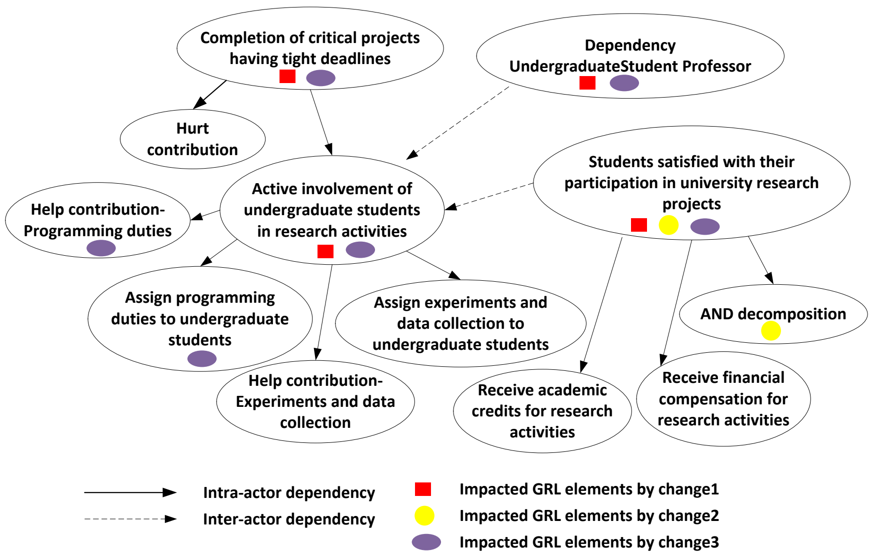

Completion of critical projects having tight deadlines
Dependency UndergraduateStudent Professor
Hurt contribution
Active involvement of undergraduate students in research activities
Students satisfied with their participation in university research projects
Help contribution-Programming duties
Assign programming duties to undergraduate students
Assign experiments and data collection to undergraduate students
AND decomposition
Help contribution-Experiments and data collection
Receive academic credits for research activities
Receive financial compensation for research activities
Intra-actor dependency
Inter-actor dependency
Impacted GRL elements by change1
Impacted GRL elements by change2
Impacted GRL elements by change3

Fig. 9. GRL Model Dependency Graph corresponding to the GRL model in Figure 7

Figure 9 illustrates the marked GMDG corresponding to the GRL model in Figure 7 and the three proposed changes of Figure 8.

V. Comparison with related work

There have been a number of studies on goal models that establish traceability links between the requirements and the system design. Lamsweerde [6] has proposed an approach to derive software architectures from a system goal model using heuristics. The approach assigns responsibilities for achieving goals to their corresponding components and establish connections among them. Yu et al. [7] have proposed a technique for generating a highly versatile software design from goal models. Their technique transforms goals into components and determines component connections from AND/OR-refinement links. Lee et al. [8] have proposed a change impact analysis approach using a goal-driven traceability-based technique. The authors have used traces among goals and use cases to analyze requirements changes. Goals and use case are connected via three different traceability relations (evolution, dependency, and satisfaction), which are stored in a design structure matrix. Impacted entities can then be determined by applying a reachability analysis on the matrix. While these approaches establish traceability with other artifacts, our proposed approach focus on the impact of changes in the goal model itself. Furthermore, our approach can be combined with approaches like the ones presented in [6] [7] [8] by adding traceability links from the GRL elements to other artifacts.

Some studies have proposed ways to support developers in making requirements changes in goal models. Ernst et al. [9] have introduced the notion of a Requirements Engineering Knowledge Base (REKB) for maintaining a requirements model. The authors explore the case where unanticipated changes occur to the requirements of an operational system, such as a new law coming into effect, or adding new features suggested by the marketing team. Our proposed approach is different from this respect as we conduct change impact analysis on goal models once requirements changes have been identified. Cleland-Huang et al. [10] have introduced a probabilistic approach to manage traceability links for non-functional requirements. Non-functional requirements and their dependencies are modeled with a Softgoal Interdependency Graph (SIG). Designers can then analyze the impact of changes by retrieving links between classes affected by changes in a softgoal interdependency graph. While Cleland-Huang et al. use the interdependence of non-functional requirements, our proposed approach is based on the intrinsic structure of the goal model and does not consider the type of requirements. Nakagawa et al. [11] proposed a process of elaboration for goal models, expressed in KAOS [12], that extracts a set of control loops from the requirements descriptions. These control loops are considered to be independent components, hence, preventing the impact of a change from spreading outside them. Our proposed approach allows for the identification of effects spreading between GRL actors.

VI. Conclusions and future work

We have proposed an approach to change impact analysis that allows requirements engineers to assess the possible impact of early changes in goal-oriented models. The proposed approach identifies whether the proposed changes are propagated to external actors. Furthermore, we have extended the use of the well-known technique of program slicing to GRL Model Dependency Graphs, that are derived from GRL models to describe model dependencies. As a future work, we plan to combine our approach with existing GRL goal satisfaction evaluation strategies, in order to have a precise assessment of the magnitude of a given change.

References

[1] R. S. Arnold, Software Change Impact Analysis. Los Alamitos, CA, USA: IEEE Computer Society Press, 1996.

[2] S. Lehnert, “A taxonomy for software change impact analysis,” in Proceedings of the 12th International Workshop on Principles of Software Evolution and the 7th Annual ERCIM Workshop on Software Evolution, ser. IWPSE-EVOL ’11. New York, NY, USA: ACM, 2011, pp. 41–50. [Online]. Available: http://doi.acm.org/10.1145/2024445.2024454

[3] ITU-T, “Recommendation Z.151 (10/12), User Requirements Notation (URN) language definition, Geneva, Switzerland,” Genève,Switzerland, 2012. [Online]. Available: http://www.itu.int/rec/T-REC-Z.151/en

[4] M. Weiser, “Program slicing,” in Proceedings of the 5th International Conference on Software Engineering, ser. ICSE ’81. Piscataway, NJ, USA: IEEE Press, 1981, pp. 439–449. [Online]. Available: http://dl.acm.org/citation.cfm?id=800078.802557

[5] J. Hassine and M. Alshayeb, “Measurement of actor external dependencies in GRL models,” in Proceedings of the Seventh International i* Workshop co-located with the 26th International Conference on Advanced Information Systems Engineering (CAiSE 2014), Thessaloniki, Greece, June 16-17, 2014., ser. CEUR Workshop Proceedings, F. Dalpiaz and J. Horkoff, Eds., vol. 1157. CEUR-WS.org, 2014. [Online]. Available: http://ceur-ws.org/Vol-1157/paper22.pdf

[6] A. van Lamsweerde, “From system goals to software architecture,” in Formal Methods for Software Architectures, ser. Lecture Notes in Computer Science, M. Bernardo and P. Inverardi, Eds. Springer Berlin Heidelberg, 2003, vol. 2804, pp. 25–43.

[7] Y. Yu, A. Lapouchnian, S. Liaskos, J. Mylopoulos, and J. C. S. P. Leite, “From goals to high-variability software design,” in Proceedings of the 17th International Conference on Foundations of Intelligent Systems, ser. ISMIS’08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 1–16. [Online]. Available: http://dl.acm.org/citation.cfm?id=1786474.1786476

[8] W.-T. Lee, W.-Y. Deng, J. Lee, and S.-J. Lee, “Change impact analysis with a goal-driven traceability-based approach,” International Journal of Intelligent Systems, vol. 25, no. 8, 2010, pp. 878–908. [Online]. Available: http://dx.doi.org/10.1002/int.20443

[9] N. Ernst, A. Borgida, and I. Jureta, “Finding incremental solutions for evolving requirements,” in 19th IEEE International Requirements Engineering Conference (RE), Aug 2011, pp. 15–24.

[10] J. Cleland-Huang, R. Settimi, O. BenKhadra, E. Berezhanskaya, and S. Christina, “Goal-centric traceability for managing non-functional requirements,” in Proceedings of the 27th International Conference on Software Engineering, ser. ICSE ’05. New York, NY, USA: ACM, 2005, pp. 362–371. [Online]. Available: http://doi.acm.org/10.1145/1062455.1062525

[11] H. Nakagawa, A. Ohsuga, and S. Honiden, “A goal model elaboration for localizing changes in software evolution,” in Requirements Engineering Conference (RE), 2013 21st IEEE International, July 2013, pp. 155–164.

[12] A. van Lamsweerde, “Requirements engineering: from craft to discipline,” in Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE 2008), Atlanta, Georgia, USA, 2008, pp. 238–249.

Information Security in Smart Cities

Using OpenID, SAML and OAuth to increase security in urban environment

Felipe Silva Ferraz[1,2], Carlos Candido Barros Sampaio[1,2], Carlos André Guimarães Ferraz[1]

[1]Informatics Center
Federal University of Pernambuco
Recife, Brazil
emails: {fsf3, ccbs, cagf }@cin.ufpe.br

[2]CESAR
Recife Center for Advanced Studies and Systems
Recife, Brazil
emails: {fsf, ccbs}@cesar.org.br

***Abstract* — As the population living in cities and metropolis grows, the need for the transformation of a city into a Smart City grows, too. The Smart City concept refers to a broad range of definitions and technologies; among those it is possible to identify topics such as Internet of Things, ubiquity, empowering citizens, interoperability of services/systems, green systems, and open data. Whatever is the explored concept, all of them try to reach a single goal, namely, turn the city into a better place to live through the use of Information Technology. In this context, more specifically, interoperable environments play an important role as it provides means to connect different services by creating new systems that are able to provide a bigger variety of information to its users. In this matter, a new range of challenge rises; among those challenges, information security is one of the most important to be discussed. This paper presents a group of security issues that raises a dangerous concern in Smart City solutions, discusses how identity standards, such as OpenID, SAML and OAuth, impact on those issues.**

Keywords — Smart City; Information Security; Security Standards.

I. INTRODUCTION

The evolution of cities has been taking a toll both on the environment, as well as on the shape of human population. Its needs have been increasing, and so the pressure on the ecosystem of systems has been constantly escalating.

In the modern world [1], sustainability is a major issue; if we keep on exploiting resources and services without any thought about the following generations, the future of human race may no longer have enough resources to survive. But also, industrial development is as important as environmental issues; safety is as important as saving time; continuous availability of resources is as important as not exploiting them, and so on [2]–[4].

To match this kind of situation, the citizens of major cities around the world must become more informed, responsible and efficient, in order to gain and provide faster and more continuous access to information [3][5][6]. Every structure, whether its intended use relates to resources, health, government, transportation, education, or public safety systems was designed, built and maintained with advanced, integrated materials, sensors, electronics, and networks to provide those citizens with the means to attend those needs [7]–[9].

In this area, new solutions are absolutely necessary not only to improve the quality of daily-life with innovative, sustainable, long term and efficient protocols but also in terms of security/reliability. Security and/or reliability are important paths mostly because the solutions will be exposed to an extensive range of attacks. Internal and external parties are not trusted, and privacy, integrity and availability will be a vital prerequisite for the approval of the citizens. In addition to it, since the assumptions and requirements for smart critical infrastructures are very different, implying that networks for smart cities should be engineered quite differently, it also raises a problem of integration or interoperability [10].

With cities progressing towards smarter societies, worldwide Information and Communication Technologies (ICT), a class of software is also advancing and ushering itself to the IT sector, nowadays, called ‘green software’ or ‘smart software’ [11]. The primary role of a smart software is to enable the functioning of the devices, running them in such a way that the device is eco-friendly and aids the smart behavior of a city [9][12]. For a smart software to increase the functioning and sustainability of a Smart City as a whole, the devices with smart software must contribute to the entire system. As software systems are vulnerable to threats, the Smart City should be prepared for such attacks and breaches of security [11].

Web-based protocols, for instance, protocols as OAuth [13], Security Assertion Markup Language (SAML) [14], and OpenID [15], play an important role in web and cloud platforms [16]. They present means to guarantee access to specific services in order to provide authorized access to its private data to the users. Furthermore, they present a major contribution to single signing needs.

This paper presents an analysis on 9 security issues proposed in previous works [14][15], facing three different security standards. This paper is divided as follows: after a brief Introduction, Section II will introduce the concepts of Smart Cities, followed by Section III dealing with security analysis on smart cities; Section IV will depict 9 security issues in the role of smart cities solutions. Section V will explore differences and strengths on OpenID, SAML and OAuth. Finally, Sections VI and VII will finish this paper analyzing the impact of the mentioned standards and presenting some conclusions, respectively.

II. SMART CITIES

A long and exhausting talk has been taking part in research areas – solutions of Smart Cities, studies and implementation play an important role in solving a visible problem. How to deal with the unprecedented level of citizens living in cities? Differently from other ages, large cities have now most part of the world population and an increasingly share of the world's most skilled, educated, talented, creative and entrepreneurial personalities. For the first time in human history, more than 50 percent of the population of the world now lives in large cities, and what is more alarming is that, according to the United Nations, this number will increase to 70 percent in less than 50 fifty years [19]. This *city growth* or emerging of urban life is taking the city infrastructure to a stress level that has never been seen before, since the demand for basic services increased and are exponentially overloaded [7].

According to a research called *Smarter Cities and Their Innovation Challenges* [20], there is an urgent need for urban scenarios and cities to be smarter in the management of their infrastructure resources and interactions [20]. The urban performance must not rely only on its hardware infrastructure, or the physical concepts of infrastructure, but it must start taking into account the social interactions and a faster deployment of information and services.

Cities are becoming increasingly technologically empowered as their core systems; e.g., Education, Public Safety, Transportation, Energy and Water, Healthcare and Services are instrumented and interconnected, enabling new ways to deal with massive, parallel and concurrent usage. In the same pace, new challenges in the field of information security rise and must be properly addressed.

III. SMART CITIES SECURITY ANALYSIS

Despite the number of studies and protocols related to information security, the amount of vulnerabilities in connected applications has increased in the past few years. In this matter, Smart Cities systems will demand a specific treatment in order to address its specific security challenges information.

According to [9][21]–[23], Smart cities solutions rely on a high degree on connectivity, so that their systems (such as Education, Government, Traffic, Security, Resources and Health) can create an interoperable network, providing more powerful, accurate and unique services to the citizens. For this reason, one of the biggest challenges facing Smart City development is related to Information Security of Interoperable Systems [1]. Information security is a critical issue due to the increasing potential of cyber attacks and incidents against critical sectors in a Smart City. Information Security must address not only deliberated attacks, such as from disgruntled employees, industrial espionage, and terrorists, but also inadvertent compromises of the information infrastructure due to the user errors, equipment failures, and natural disasters. Vulnerabilities might allow an attacker to penetrate a network, gain access to control the software and alter load conditions to destabilize the system in unpredictable ways. To protect the Smart City in a proper way, a number of security problems have to be faced according to a specific design/plan.

It would be too simplistic, and probably a lapse, to believe that traditional security approach based only on privacy and authorization concepts can simply be added into a critical infrastructure of a city to make it safer as much as it becomes smarter. New architectures are necessary not only to improve the security services, but the interoperability and the security in general.

This class of services is fundamental to the success of the future city, and represents a topic of such complexity that it is beyond the scope of this paper to cover in details. As an illustration, let us explore the design of identity services for the future city – which is required to maintain privacy while maintaining security.

The integration of the identity of the citizen across multiple systems and services and the ability to provide a joined-up response to life events needs, comprises the goal of allowing the citizens to manage their own identity and what information is released about them to who or when, while anonymous, aggregate data is made more widely available.

So, Identity Management is an essential key for future cities. A unified identity system, one that can integrate itself with multiple identity providers and different forms of authentication and identification, is needed to handle the extensively wired nature of the city and the density of data transactions, systems and diversity of solutions.

Citizens or entities will use their identities to get access to services and systems, and through that the benefits offered by those. This way, to integrate several solutions (systems and services), entities and services will eventually repeat their identification artifact in different moments and situations.

Ideally, every citizen and/or entity shall have a number of identities, each is made of a number of attributes, which are either exposed, or used to validate a request without exposing the information. The use of multiple identities limits the exposure of truly important credentials, minimizing risk of abuse and identity steal, while allowing the exposure of less critical information that is helpful for participants in the city ecosystem such as retailers, building operators, service providers, and governments.

Not only the citizens will be in charge of their identities, but also the information that constitutes them, and when this information can be exposed. The proposed solution is proposed to build a trustworthy relationship between the city, the services/systems and citizens, allowing the acquisition and flow of information that is helpful to all participants without compromising their identities.

IV. SECURITY ISSUES IN THE ROLE OF A SMART CITY

Based on previous studies [14][15], which brought to attention the need to make further improvements, related to information security on Smart Cities environment, this session will depict a set of 9 Security issues that an urban system and a city may be under the risk of.

It is important to complement that, even though the technical solutions applied in those environments handle with matters, such as Code Injection [24], Cross Site

Scripting [24], Cross Site Request Forgery [24] and Buffer Overflow [24], and others, the issues presented in this work explore concepts related to the nature of a Smart City system. Our proposal is to present a set of issues that, regardless of the technical solution applied, may be a threat to urban cities (or a Smart City system) in a different level.

Figure 1 presents an overview of this difference.

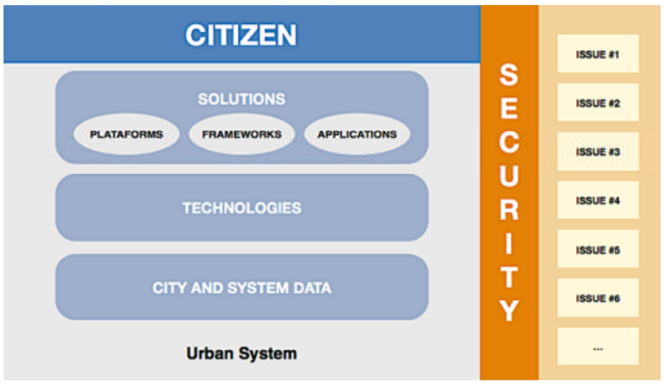


Figure 1. Security issues overview

Urban Systems are composed by *Citizens* using *Solutions,* which could be *Platforms, Frameworks and Applications;* All of those built on *Technologies* to receive and use *Data*. Urban System Security Issues or Security Issues in the role of a Smart City are situations that can be put as problems to the infrastructure as a whole.

In the following section, those issues will be depicted, in order to illustrate the impact on OAuth [13], Security SAML [14], and OpenID [15].

A. *Access to information from applications*

According to Sen et al. [25], packet transfer must be addressed, in order to apply efforts to add security improving data privacy and integrity. Looking from a network and access perspective, devices have the meanings to access a packet, or a set of packets, in different ways and locations and with different efforts. For instance, to reduce latencies during data transfer, local copies or cache values of those packets could be created.

Let us assume that a sensor connects to a server to identify and authenticate user A and retrieves its permissions. During this process, a user B could intercept the packet, in different points of the network or of the device, and gain a set of information from user A and the service it is accessing.

B. *Information Tracking*

It is important to have an interoperable and interconnected environment for systems to interact with one another, as in [5][11]–[15]. It is also extremely important that, for instance, the information used by system B and, that are originally created by system A, cannot be tracked back to its origin; it means that even though system A has provided a set of information to B, a user from system B should not realize that this information is from another part.

As an example, let us assume that system A provides information to a solution B.

Let us supposed that A is a system of criminal reports B is another solution that uses those criminal reports to define the most suitable place to open a new commercial building, based on criminal records. The information used by B, which was provided by A, must not be unveiled. This situation could destroy the anonymity in A and compromise witnesses and citizens victims of crimes, for example.

C. *Citizen Tracking*

Solutions for smart cities use different sensors (physical or social), those sensors are used to collect data from several city systems, and, based on this, it is possible for urban systems to have a better city management.

In order to avoid further problems, such sensors must be under the control of a responsible entity in order to preserve its functionality and generated data.

Among the possible problems raised by this topic, it can be appropriate: Unauthorized citizen tracking, discovery of movement patterns and flood of directional advertisement/ merchandising.

D. *User/Citizen data loss*

This issue deals with the concept that applications are saving precious data in the device and, if are not well treated, those data could be lost or compromised creating significant problems to the citizen.

This could be achieved by adding mechanisms related to client cryptographic storage [7][22], system isolation and even solutions related to authorization and authentication mechanisms.

E. *Crossed access to information in data centers*

For this scenario, we deal with situations related to unauthorized access to information by exploiting flaws on the server side.

For example, when accessing information related to students educational systems, a given entity (application) can recover criminal records, from a non-specific connection, related to this citizen even though the solution should only be using Educational Services. This situation may occur since both systems share a common area or permissions that must be respected in order to avoid this kind of behavior.

F. *Crossed access in client side*

Description #E, *Crossed access to information in data centers*, details a situation related to unauthorized access in server side.

Issue #F, this current topic, brings forth a subject related to unauthorized access on the client side, for instance, in a mobile device that holds sensitive information.

Different from issue #D, which the concern is about *every* information saved and that are NOT properly stored, and liable to undesired access within the context of a device.

If, for instance, system A saves in a device values related to paid fees, and system B uses the same mechanism to store information regarding the user financial account. If the device does not provide A and B with the correct isolation, it is possible that through A an attacker gains access to values presented in B, and even more, it is possible that a malicious third part system may be installed and, then, gains access to both systems information [28].

G. Lack of Security in Depth

According to OWASP TOP 10, one of the Top Ten risks to WEB application is related to code injection. Also according to OWASP, sanitize inputted values and remove undesired texts is one of the measures to avoid that and other security flaws [24].

This flaw, #G, is related to systems that do not validate data in different layers, and are compromised in any level, by data originated from other services.

In other words, if system A provides a rich UI environment to the user and has several validations and sanitization in this part, if the back-end structure does not apply the same criteria, whenever a system C sends data to system A, system A may use this data. It means that, if C has a malicious code inputted, it might be transferred to A once A misses defense in depth concepts.

H. Viral effect in urban environment

A Smart City uses an interoperable environment to provide solutions with the opportunity to interact with other system, exchanging data and creating more value to its citizens [2].

If the border of these relations is not well defined, the systems may face a scenario where a value is changed in system A and when system B uses this changed value, it may corrupt the information used in system B.

In issue #G, our main concern is with the lack of protection in every layer, and how this could be a problem that an urban scenario is highly connected. In the present issue, our concern is with the consequences of issues like issue #G, if the system is highly connected and lacks protections in several parts, the consequences of an attack can be exponentially increased, infecting the entire solution through the infection of a small part.

I. Infection traceability and recovery

The amount of data used and stored by a Smart City has reached unprecedented levels. Moreover, the connection between systems has created a System of Systems structure that provides those solutions with data coming from different services.

Issue #H presents a viral threat related to a set of data that can share or provide another service with different data, creating, at some level, a self-sustained system. From that point of view, this issue presents a consequence for issue #I. Due to the amount of data and interconnected system, it is possible for an infection to maintain its origin undetected and beyond data recovery.

Using as an example, System A, with terabytes of data, exchanging values with a System B, that feeds Systems C and D with updated and new values, processed from A data. D, on the other hand, keeps passing some fine-grained data back to System A. If A suffers an infection having its data compromised, B will be fed with infected data, spreading the infection to other systems, like C and D. As soon as the infection could be detected, recovery processes may not be an option since the amount of data compromised is too big to be restored to a previous form. In addition, due to the relation between systems, the infection source may not be detectable.

V. EXPLORING OAUTH, SAML AND OPENID

To address some issues mentioned on the previous section, identification management will play an important role [28]; therefore, this section will present architectural solutions that addresses security issues, specially related to identification, authorization and authentication across an interoperable environment. This section will depict three different approaches that offer a set of functionalities that could aid mitigating previous issues.

According to approaches related to, or making use of, OAuth 2.0 [27][28], SAML [29][30] and OpenID [31][32], it appears as the bigger responsible for security assurance in interoperable environments. For this reason, OAuth, SAML and OpenID will be depicted in the following section, and compared with the mentioned issues in order to understand if there are positive impacts on a Smart City environment.

A. OAuth

Open Authorization (OAuth) is an authentication standard used by service providers to store protected resources in a way that a resource owner do not have to hand out their credentials to gain access to the protected assets. It means that through OAuth is possible to authorize another website, with access to the user information stored within another service provider, without the need to share their access permissions.

The basic structure of OAuth is composed by a Resource Owner, that is an entity responsible for storing protected assets and is capable of granting access to the assets under its control, an Authorization Server is responsible for handling authentication and authorization of different entities involved and a Resource Server that is a server that hosts the client asset [27][28][33].

Figure 2 shows the basic flow of an OAuth structure:

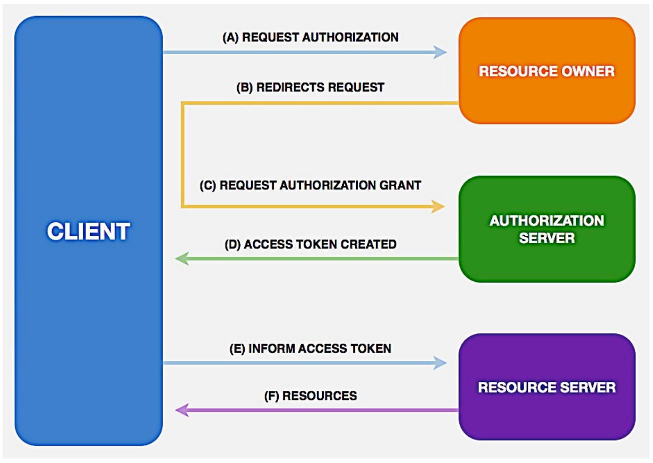


Figure 2. OAuth basic flow

(A) In the first step, a **client** requests an authorization from the **resource owner**.

(B) The **resource owner**, replies to the **client** and redirects the request to **authorization server**.
(C) The **client** requests an authorization grant from the **authorization server** by presenting the client credentials.
(D) The **authorization server** validates the **client** credentials and the authorization grant, and if valid issues an access token.
(E) The **client** requests the protected resource from the **resource server** and authenticates by presenting the access token.
(F) The resource server validates the access token, and if valid, serves the request.

B. SAML

Security Assertion Markup Language (SAML) defines an XML based framework used to describe and exchange information related to security between secure web-based entities [1][34]–[37].

SAML is a reference standard that implements identity provider that has the capability to address several security scenarios and technologies. The main strength of a SAML based system is that it can create a trust relationship using entities that relies on different security mechanism. Different from other security systems SAML approach is to express assertions about an entity that other application within the same network or environment can trust

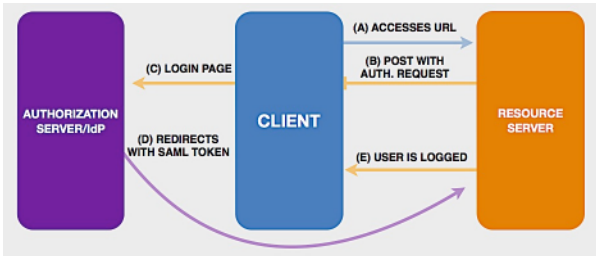


Figure 3. SAML basic flow

(A) A **client** tries to access an asset in **resource server**.
(B) The **resource server**, redirects the request to the **authorization server**.
(C) The **client** informs login and password.
(D) Once the authentication is made, the **authorization server** redirects the SAML token to the **resource server**.
(E) The asset access is granted to the **client**.

C. OpenID

OpenID is a distributed open standard technology, used to identify users with URL typed ID. Any type of system can use an OpenID protocol without any kind of fee.

The final user also does not need to depend on a specific site or domain to keep their ID controlled. It means that they do not need to enter any of its personal information such as email, name, address or other identifiers to have an ID and password for every site, instead all that is necessary is to lot in using their OpenID in a site that adopts and OpenID system [21]. Due to this property, a user do not need to have a separate ID and Password for each site further OpenID creates the effect of a outsourced user authentication service.

OpenID basic flow is composed of a Client, which represents and entity using the OpenID system, Relying Party (RP) that is the service provider and the OpenID Provider that holds the logic related to IDs and Passwords [29][38], as presented in Figure 3.

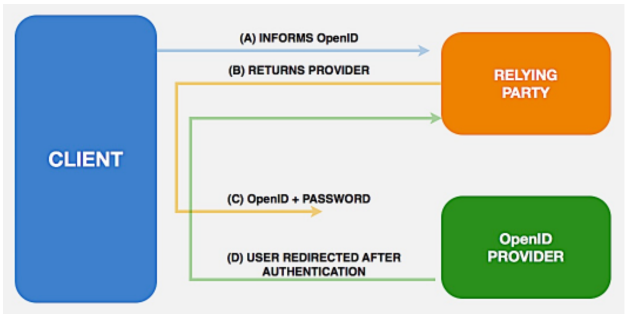


Figure 4. OpenID basic flow

(A) The client informs the OpenID to a **RP**.
(B) The **RP** normalizes the clients OpenID, identify the OpenID and redirects it to the client.
(C) The **client** informs credentials for the ID.
(D) After authentication, access is granted.

VI. Towards Issues Analysis

In this section, the impact of OAuth, SAML and OpenID, under the vision of the security issues mentioned in Section III, will be analyzed.

The three-depicted protocols have a direct relation with both creation and maintenance of identifiers and with authorization and authentication in an interoperable environment.

A. Access to information from applications

Once the token is generated, and somehow stored within the client, only the OpenID presents means to avoid this threat, due to its characteristic of asking for a password once the ID is presented. This way even if the ID is compromised, the attacker needs to have extra security information about the ID. OAuth and SAML, on the other hand, it does not request other verification after the token is created.

B. Information Tracking

This issue is related to the concept of an information source not being able to be discovered. About this idea, the three mentioned protocols have no ways to avoid the issue if the correct authorization attribution is not made. In other words, the protocol can address the problem, but if not well used, or by any human misconfiguration it can still be explored.

C. *Citizen tracking*

Considering that the token generated by OAuth and SAML are compromised, it is possible to track information from different systems. To both mentioned protocols, even if it is possible to explore the flaw, it also is unlikely that every system used by the citizen could be reachable by the same token. In OpenID relies the trust that, even though the ID is compromised, the user needs to achieve also the password of the citizen.

D. *User/Citizen data loss*

Since the main focus of the studied protocols are related to the interoperability of systems, it has no direct relation with protection regarding the client side. That way OAuth, OpenID and SAML are marked with no positive impacts with issue number #4.

E. *Crossed access to information in data centers*

Similar to issue number #2, this issue is partially addressed by OpenID, SAML and OAuth, because they have the strengths to solve this kind of scenario, but due to incorrect use of the protocols or by human mistakes, when adding the permissions and authorizations, data could be compromised even in the server side.

F. *Crossed access in client side*

Similar to issue #4, issue #6 deals with the concept of client side been compromised, the difference in this case is about the notion that a data in application A could be wrongly accessed from another application B, causing data leakage from one app to another, whereas for issue #4 it is related to data loss on the client side by any other means, for example, week client storage. The same explanation for issue #4 is applied for #6 and the three protocols have no impact on this scenario.

G. *Lack of Security in Depth*

As previously mentioned, Security in Depth is a concept that suggests adding several layers of protection within a system scope. OAuth, SAML and OpenID, deals with the concept of providing few tokens to every user making it simpler to log and gain access to the proper asset. Due to that fact it is feasible to realize that the three protocols make the use of a set of services easier, but lose some security by repeating the same checks for different services.

H. *Viral effect in urban environment*

Issue #8 is potentially solved by the three protocols since they present means to avoid actions coming from unauthorized parts. Even though they are susceptible to human flaws, it is still highly unlikely that for every system using the protocols, they present bad settings or operational flaws.

I. *Infection traceability and recovery*

Finally, issue number #9 deals with the idea that if some point of a broad system is compromised, it is improbable to identify where the infection or the flaw was first initialized and also to recover the state of the system to a previous version. Since OAuth, SAML and OpenID, deal with a single ID for a set of systems, this will make it impossible to track which system the flaw came from.

Table I uses the following subtitle, to summarize the impacts of each one of the three protocols: ✗ for no positive impacts, ✗/✓ partially address the scenario. ✓ directly addresses the situation.

TABLE I. OAuth, SAML AND OpenID IMPACTS AGAINST ISSUES

ISSUES	COVERAGE		
	OAuth	SAML	OpenID
1. Access to information from applications	✗	✗	✓
2. Information Tracking	✗/✓	✗/✓	✗/✓
3. Citizen Tracking	✗/✓	✗/✓	✓
4. User/Citizen data loss	✗	✗	✗
5. Crossed access to information in data centers	✗/✓	✗/✓	✗/✓
6. Crossed access in client side	✗	✗	✗
7. Lack of Security in Depth	✗	✗	✗
8. Viral effect in urban environment	✓	✓	✓
9. Infection traceability and recovery	✗	✗	✗

VII. CONCLUSION AND FUTURE WORK

For the first time in human history, humanity is facing a unique situation where more than 50% of the population lives in big cities. To work it out, there is an urgent need to evolve information technology systems to solutions that provide the citizens more and detailed information about different subjects of its daily usage.

At the same time that new solutions rise, new challenges are also developed. Among those, information security plays an important role, and not only due to the privacy issues of the citizens; it is a subject that may go beyond citizens and impact the entire system.

Solutions like OpenID, SAML and OAuth are fundamental to guarantee the safety of the single sign on users. Unfortunately, all the expectations rely on one of those 3 standards and it may not address and solve all the problems. Most of this concern is related to the fact that those standards are under authentication and authorization purposes, which, based on previously described issues are not enough. As a future work, is expected to go deeper in the analyses of the impact of OpenID, SAML and OAuth and to proposes and extension to those technologies to focuses more in smart cities identification management.

REFERENCES

[1] F. Gil-Castineira, E. Costa-Montenegro, F. Gonzalez-Castano, C. López-Bravo, T. Ojala, and R. Bose, “Experiences inside the Ubiquitous Oulu Smart City,” *Computer (Long. Beach. Calif).*, vol. 44, no. 6, pp. 48–55, Jun. 2011.

[2] A. Bartoli, M. Soriano, J. Hernandez-Serrano, M. Dohler,

A. Kountouris, D. Barthel, Security and Privacy in your Smart City , in Proceedings of Barcelona Smart Cities Congress 2011, 29-2 December 2011, Barcelona (Spain).

[3] M. Batty, K. W. Axhausen, F. Giannotti, a. Pozdnoukhov, a. Bazzani, M. Wachowicz, G. Ouzounis, and Y. Portugali, "Smart cities of the future," *Eur. Phys. J. Spec. Top.*, vol. 214, no. 1, pp. 481–518, Dec. 2012.

[4] J. Bélissent and S. Analyst, "What's new in Smart Cities ?," pp. 1–20, 2011.

[5] O. Haubensak, "Smart cities and internet of things," *Bus. Asp. Internet Things, Semin.* pp. 33–39, 2011.

[6] A. Frost, "Moving Citizens in the Smarter City — Using a Framework Approach to Plan Intelligent Transportation Systems Strategies and Implement Solutions."

[7] F. Ferraz, C. Sampaio, and C. Ferraz, "Towards a Smart City Security Model Exploring Smart Cities Elements Based on Nowadays Solutions," *ICSEA 2013*, pp. 546–550, 2013.

[8] D. Washburn, U. Sindhu, and S. Balaouras, "Helping CIOs Understand 'Smart City' Initiatives," *Growth*, 2009.

[9] W. M. da Silva, A. Alvaro, G. H. R. P. Tomas, R. a. Afonso, K. L. Dias, and V. C. Garcia, "Smart cities software architectures," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing - SAC '13*, 2013, p. 1722.

[10] J. Ko, N. Tsiftes, S. Dawson-haggerty, and M. Durvy, "Industry : Beyond Interoperability – Pushing the Performance of Sensor Network IP Stacks," pp. 1–11.

[11] M. Sen, A. Dutt, S. Agarwal, and A. Nath, "Issues of Privacy and Security in the Role of Software in Smart Cities," *2013 Int. Conf. Commun. Syst. Netw. Technol.*, pp. 518–523, Apr. 2013.

[12] M. Chen, "Towards smart city: M2M communications with software agent intelligence," *Multimed. Tools Appl.*, vol. 67, no. 1, pp. 167–178, Feb. 2012.

[13] OAuth, "OAuth 2.0." [Online]. Available: www.oauth.net.

[14] F. Nie, F. Xu, and R. Qi, "SAML-Based Single Sign-On for Legacy System," no. August, pp. 470–473, 2012.

[15] J.-H. You and M.-S. Jun, "A Mechanism to Prevent RP Phishing in OpenID System," *2010 IEEE/ACIS 9th Int. Conf. Comput. Inf. Sci.*, pp. 876–880, Aug. 2010.

[16] J. Sendor, Y. Lehmann, G. Serme, and A. Santana de Oliveira, "Platform-level Support for Authorization in Cloud Services with OAuth 2," *2014 IEEE Int. Conf. Cloud Eng.*, pp. 458–465, Mar. 2014.

[17] F. S. Ferraz and C. A. G. Ferraz, "More Than Meets the Eye In Smart City Information Security: Exploring security issues far beyond privacy concerns," in *IEEE computer science, UFirst-UIC 2014*, 2014.

[18] F. S. Ferraz and C. A. G. Ferraz, "Smart City Security Issues: Depicting Information Security Issues in the Role of an Urban Environment," in *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, 2014, pp. 842–847.

[19] S. Dirks and M. Keeling, "A vision of smarter cities: How cities can lead the way into a prosperous and sustainable future," *IBM Inst. Bus. Value. June*, 2009.

[20] M. Naphade, G. Banavar, C. Harrison, J. Paraszczak, and R. Morris, "Smarter Cities and Their Innovation Challenges," *Computer (Long. Beach. Calif).*, vol. 44, no. 6, pp. 32–39, Jun. 2011.

[21] C. Harrison, B. Eckman, R. Hamilton, P. Hartswick, J. Kalagnanam, J. Paraszczak, and P. Williams, "Foundations for Smarter Cities," *IBM J. Res. Dev.*, vol. 54, no. 4, pp. 1–16, Jul. 2010.

[22] C. Balakrishna, "Enabling Technologies for Smart City Services and Applications," *2012 Sixth Int. Conf. Next* Gener. Mob. Appl. Serv. Technol., pp. 223–227, Sep. 2012.

[23] N. Mitton, S. Papavassiliou, A. Puliafito, and K. S. Trivedi, "Combining Cloud and sensors in a smart city environment," EURASIP J. Wirel. Commun. Netw., vol. 2012, no. 1, p. 247, 2012.

[24] OWASP, "OWASP Top 10 - 2013 : The the most critical web application security risks," 2013.

[25] M. Sen, A. Dutt, S. Agarwal, and A. Nath, "Issues of Privacy and Security in the Role of Software in Smart Cities," in 2013 International Conference on Communication Systems and Network Technologies, 2013, pp. 518–523.

[26] O. Garcia-Morchon, S. L. Keoh, S. Kumar, P. Moreno-Sanchez, F. Vidal-Meca, and J. H. Ziegeldorf, "Securing the IP-based internet of things with HIP and DTLS," Proc. sixth ACM Conf. Secur. Priv. Wirel. Mob. networks - WiSec '13, p. 119, 2013.

[27] I. Verbauwhede, "Efficient and secure hardware," Datenschutz und Datensicherheit - DuD, vol. 36, no. 12, pp. 872–875, Nov. 2012.

[28] M. Schumacher, E. Fernandez-Buglioni, D. Hybertson, F. Buschmann, and P. Sommerlad, Security Patterns : Integrating Security and Systems Engineering (Wiley Software Patterns Series). John Wiley & Sons, 2006.

[29] M. Noureddine and R. Bashroush, "A provisioning model towards OAuth 2.0 performance optimization," 2011 IEEE 10th Int. Conf. Cybern. Intell. Syst., pp. 76–80, Sep. 2011.

[30] B. Leiba and H. Technologies, "OAuth Web Authorization Protocol," pp. 0–3, 2012.

[31] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, "Security and Cloud Computing: InterCloud Identity Management Infrastructure," 2010 19th IEEE Int. Work. Enabling Technol. Infrastructures Collab. Enterp., pp. 263–265, 2010.

[32] H. Wang, C. Fan, S. Yang, J. Zou, and X. Zhang, "A New Secure OpenID Authentication Mechanism Using One-Time Password (OTP)," in 2011 7th International Conference on Wireless Communications, Networking and Mobile Computing, 2011, pp. 1–4.

[33] F. Yang and S. Manoharan, "A security analysis of the OAuth protocol," 2013 IEEE Pacific Rim Conf. Commun. Comput. Signal Process., pp. 271–276, Aug. 2013.

[34] T. T. A. Dinh, W. Wenqiang, and A. Datta, "City on the Sky: Extending XACML for Flexible, Secure Data Sharing on the Cloud," J. Grid Comput., vol. 10, no. 1, pp. 151–172, Mar. 2012.

[35] S. Dirks and M. Keeling, "A vision of smarter cities," IBM Inst. Bus. Value, 2009.

[36] M. Al-Hader, A. Rodzi, A. R. Sharif, and N. Ahmad, "SOA of Smart City Geospatial Management," 2009 Third UKSim Eur. Symp. Comput. Model. Simul., pp. 6–10, 2009.

[37] A. Aldama-Nalda, H. Chourabi, T. a. Pardo, J. R. Gil-Garcia, S. Mellouli, H. J. Scholl, S. Alawadhi, T. Nam, and S. Walker, "Smart cities and service integration initiatives in North American cities," Proc. 13th Annu. Int. Conf. Digit. Gov. Res. - dg.o '12, p. 289, 2012.

[38] T. Tran and C. Wietfeld, "Approaches for optimizing the performance of a mobile SAML-based emergency response system," 2009 13th Enterp. Distrib. Object Comput. Conf. Work., pp. 148–156, Sep. 2009.

Early & Quick Function Point Method

An empirical validation experiment

Roberto Meli
Data Processing Organization Srl
Rome, Italy
email: roberto.meli@dpo.it

***Abstract*— The "Early & Quick Function Points Approach (E&QFPA)" is a mean to approximate the results of some standard Functional Size Measurement Methods like IFPUG, SiFP or COSMIC. E&QFPA is a set of concepts and procedures that, even when applied to non-detailed functional specifications of a software system, maintains the overall structure and the essential principles of standard functional size measurement methods. The E&QFPA combines different estimation approaches in order to provide better approximations of a software system functional size: it makes use of both analogical and analytical classification of function types (transactions and data). Moreover, it allows the use of different levels of detail for different branches of the system (multilevel approach). This paper illustrates the basic concepts of the method which is mature and well established in the Italian market, as well as the results of an empirical validation experiment conducted on a real business data set of IFPUG function point measures. The usage of such a method may contribute to the rapid quantification of user requirements very early in the production life cycle.**

Keywords: function; point; estimation; approximation.

I. Introduction

A Functional Software Measurement Method (FSMM) is a mean to measure Functional User Requirements (FUR) of a software application according to the rules of an international ISO/IEC standard [1]. The measurement process required by the most diffused FSMM is often perceived by the ICT personnel as excessively time consuming, expensive and difficult to apply in business contexts where the details needed for FSMM standard application are not always available or stable enough. Therefore, several simplified approximation processes have been proposed [3]-[7]. The Early & Quick Function Points Approach is one of these.

The paper is structured as follows: Section II summarizes the E&QFP method; Section III reports on the advantages in using the method in business practices; Section IV presents the criteria that were used to assess the accuracy of the method at various levels of application; Section V presents the conditions of the experiment and its results; Section VI contains a "qualitative" comparison of several types of approximation methods; Section VII states the conclusions.

II. Basic concepts

This Section introduces the origin and the basic concepts of the E&QFPM needed to understand the framework used for the empirical experiment. The contents presented here are not sufficient to allow an in dept comprehension of the method in itself. The reader interested in mastering the method should refer to the standard documentation [19].

The Early & Quick FP method was created in 1997 by the author in order to facilitate the approximation (also called estimation) of the IFPUG Function Points values [8][1]. It was presented for the first time at the ESCOM 97 conference [10] and later at the IFPUG conference [11]. Since then, the original approach has evolved and its usage is increased [14]-[18]. The method has been reported in 2009 as the best choice in approximating methods by the CNIPA Italian Government Authority [12]. In 2000, the approach was extended, experimentally, to the COSMIC Functional Size Method [12]. In 2006, the Early & Quick Function Points Method - E&QFPM (IFPUG version) - has become a registered trademark but the method is available in the public domain since it is managed as a "Publicly Available Method", subject to the Creative Commons license, attribution-non derivative works. The E&QFP development team has opened the doors to external contributions and the technique evolves considering feedbacks from actual users in the market. DPO continues to support, improve and customize the method publishing a new version any time it is needed by the technical community. The method is not a commercial product. A certification program has been created to guarantee that the method is used consistently among different practitioners. After 15 years from the initial formulation, the latest evolution of the method, identified as version 3.1 [19] was released in April 2012 integrating the new Simple Function Point FSMM [20]-[22].

The Early & Quick Function Points Approach (E&QFPA) is a set of concepts and procedures that, even when applied to non-detailed functional specifications of a software system, maintains the overall structure and satisfies the essential principles of standard functional size measurement methods. It may be applied to approximate different types of Functional Size Measurement Methods (FSMM).

The E&QFPA combines different estimating techniques in order to provide better approximations of a software system functional size: it makes use of both analogical and analytical classification of function types (logical transactions and data). Moreover, it allows the use of different levels of detail for different branches of the system (multilevel approach): the overall global uncertainty level in the estimate (which is a range of values, i.e., a set of minimum, most likely, and maximum values) is the weighted sum of the individual components' uncertainty levels. The "core driver" of the approach is an analytically and statistically originated table of FP (Function Points) values to be used in making functional size estimation.

The E&QFPA is based on the fundamental principles reported in TABLE I.

TABLE I - E&QFP FUNDAMENTAL PRINCIPLES

Principle	**Explanation**
Classification by analogy	Similarity in the overall functionality between new and existing known software objects.
Structured aggregation	Grouping of a certain number of lower level software logical objects in one higher level logical object.
Estimation flexibility	Data and transactional components are assessed autonomously. No predefined and fixed function/data ratio is assumed.
Multilevel approach	No discard of existent details, if available – no need of "invented" details, if unavailable.
Use of a derivation table	Each software object at each detail level is assigned a size value, based on an analytically / statistically derived table.

From now on we will restrict our interest in the IFPUG variant of the approach that we will simply call Early & Quick Function Points Method - E&QFPM.

The values in the derivation table for the IFPUG approximation model were originally stated by expert judgment and later on by the ISBSG data set analysis [23]. Once the values are determined for any specific version of the method they are not changed anymore in order to use the method in a consistent way across practitioners, environments, organizations etc. Local calibration of the E&QFPM is always possible in order to better the results for a specific context but it should be clearly reported by the practitioners as a variation of the standard version.

Figure 1 shows the estimation process starting with the Functional User Requirements interpretation and ending with the FP estimation.

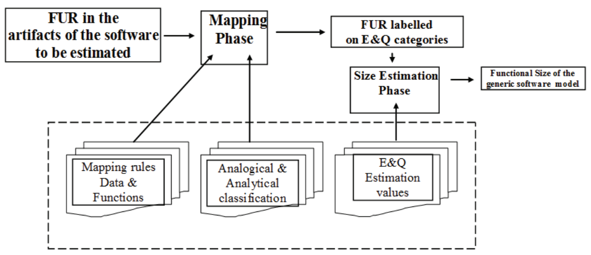

FUR in the artifacts of the software to be estimated
Mapping Phase
FUR labelled on E&Q categories
Size Estimation Phase
Functional Size of the generic software model
Mapping rules Data & Functions
Analogical & Analytical classification
E&Q Estimation values

Figure 1. Early & Quick FP Estimation Process

The starting point of the process is the logical product breakdown structure of the system being estimated, and the mapping of FURs on the E&QFP elements. The basic E&QFP elementary components are the following software objects:

- logical data groups, and
- elementary functional processes,

that is, the same Base Functional Components (BFC) types of the IFPUG measurement method. Further aggregations, as depicted in Figure 2, are provided:

- data BFC can be grouped into general data groups;
- transactional BFC can be grouped into "general" logical processes;
- general processes can be grouped into "macro" logical processes.

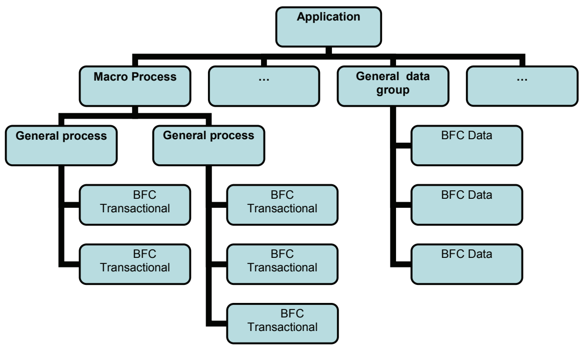


Figure 2. Functional hierarchy in the E&Q estimation method (for sake of simplicity, only one instance of macro process and one instance of general data group are shown)

Each "software logical object" is assigned a set of FP values (minimum, most likely, maximum) based on an analytical/statistical table, then the values are summed up to provide the overall estimation result (minimum, most likely, maximum). To obtain the estimated size of the software application being considered, a "structured" list of its processes and data groups is the only required item, even comprising non-homogeneous levels of detail. Knowledge of similar software objects will make it easier to assign the right level of classification to each element on the list, and therefore to derive its contribution to the overall size.

Obviously, if any particular object in the functional tree is estimated assigning directly an FP value then no *contained object* (i.e., in a "father"-"son" relationship) must be considered to contribute directly to the grand total: as a matter of fact, all the "son's" values are already included into the "father's" values and must not be added to it. The estimation uncertainty (represented by the minimum-maximum range) is dependent on the level of object in the hierarchy and will be greater for higher levels of software objects aggregation, due to the higher lack of details.

Estimation using E&QFP technique may be done at three levels of detail depending on the granularity of the components used for estimation: summary, intermediate and detailed.

III. ADVANTAGES IN USING E&QFPM

E&QFPM is an approximation method which is:

- Fast
- Cheap
- Adaptable
- Reliable
- Documentable
- Easy to learn

Fast: the empirical study described in Section V showed a productivity from 1.9 to 6.3 times better than a standard measurement. This result is aligned with informal experience derived by daily use of the method by practitioners in several Italian organizations. The reason is that much less elements should be identified and evaluated in the User Requirements documentation than for the IFPUG measurement.

Cheap: because it is fast and does not involve more specialized people than a standard measurement.

Adaptable: because it is applicable either when the standard technique is not a possible choice - due to missing detailed information –either when the details are available.

Reliable: when we consider the "technical accuracy" (see later).

Documentable: since the approximation is based on elements extracted from the FUR that may be described, discussed, shared and tracked by the practitioners. Direct estimation techniques are only based on personal intuition.

Easy to learn: because the rules are much simpler than the ones in the standard reference measurement manual. This does not mean that the method is always easy to apply, actually, since when the FUR are not enough detailed to identify BFC (Base Functional Component – the smallest measurable part of FUR – see [2]), a strong experience is needed to practice the analogy, which is essential for a good estimation.

Of course, any estimation method (and E&QFPM is not different) has a unavoidable uncertainty in its usage that makes it incomparable with a measurement method in terms of accuracy but we must accept that if we use an estimation method it is because we are not able (due to missing information) or do not want (due to missing measurement resources) to use a measurement method and the cost of doing that is a higher potential error in sizing.

IV. EVALUATION RULES AND CRITERIA USED TO ASSESS THE RELIABILITY OF THE E&QFP METHOD

In this Section, we will clarify the criteria used to represent the outcomes of the E&QFP method when applied to a real life sample of software applications, which was not used to calibrate the method in itself.

First of all, we have to work out the meaning of the quality attribute named "accuracy" of the method. By "accuracy" we intend the absence of systematic and random errors. According to [1] it is the "closeness of agreement between a measured quantity value and a true quantity value of a measurand". In our case the "*true quantity value" is the IFPUG FP value* for a software application and *the E&Q value for the same measurand is the "measured quantity value"* (measured with approximation, of course) that we want to analyse. In addition, we use the term "measurement error" or simply "error" to intend the "measured quantity value minus a reference quantity value" whereas the "relative error" is the "error divided by the reference quantity value". Again, in our case, the *measured quantity value is the E&QFP value* determined for a software application and the *reference quantity value is the IFPUG FP value* for the same application.

Estimation is a human intensive process which involves personal capabilities in addition to technical tools and rules. In evaluating the accuracy of an approximation method (its capability to predict exact values) we should be able to separate the judgement on the human capabilities from the judgement of the method itself. This is usually not easy to do since methods are used by human beings: their knowledge of the software requirements, of the application domain and of the method itself are directly related to the final accuracy of results, in real specific situations. Nevertheless, for FP approximation, we believe it is possible to assess separately the **"technical accuracy"** and the **"operational accuracy"**.

As we have seen, the E&QFP method is based, mainly, on the classification of logical requirements with respect to a standard or customized assignment table provided by the method itself. If we apply the estimation method on an already built application we have the capability to construct the exact hierarchy of logical objects to be used in the estimation at various levels of detail. Furthermore, since we have all the detailed information, we are able to assign each aggregated object exactly in the right class of the table. For example, if we aggregate some detailed requirements into a general requirement and we identify it as a General Process we are able to count exactly how many BFCs are grouped into the GP and we may exactly classify it as a small, medium or large GP, as required by the E&QFPM.

In this way, we may not be wrong with classifications and the final accuracy will be purified by the human classification errors. The residual error is given by the difference between the actual measurement of the BFCs included in the GP and the weight that the method assigns to it: this is what we call "technical accuracy". In real life, estimators will not experience this simplified situation since the requirements will not be generally known at the most detailed level and in addition to the eventual "technical" error it is possible to have a classification error which leads to an "operational" error which we expect to be greater than the "technical" one, unless underestimations compensate overestimation in the overall exercise. In addition to this aspect, it is also essential to understand that the method may be used at three different levels of detail each of one characterized by a different technical accuracy, so that it is impossible to assess the accuracy of the method, generally speaking, but it is possible to assess the accuracy of the method *at a specific level of application*.

Ignoring these essential aspects may lead to inconsistent results and false conclusions on the reliability of the method in itself as reported in [24], where the capability of freshly and quickly trained practitioners was measured together with the "technical accuracy" and estimation levels of detail were mixed up comparing incomparable results and deriving an overall accuracy for the method, which simply doesn't exist. If we want to draw conclusions on the accuracy of the method and not on the capability of practitioners then we have to depurate the experiment from the participant's bias using the approach outlined here. In real life contexts, the quality of the operational estimation will depend on the expertise of the estimator, on the available FUR details, on the level of application of the method. If we do not fix all these parameters in an experimental situation it is impossible to conduct adequate empirical observations.

Before introducing the indicators that we used to assess the "technical accuracy" we want to state that in this paper we assume that given a specific software application the exact value of FP is the value measured using the standard IFPUG rules by a certified FP Specialist (CFPS) and the estimated values to be compared with the measured value are those expressed by a certified E&QFP specialist for the same application. The error is the difference between the approximated value and the standard measured value, the relative error is the difference between the approximated value and the measured value all divided by the measured value.

A. *Portfolio error*

This is the error derived for the entire sample conceived as if it was a portfolio of applications, in other words a set of software applications managed as a whole for business reasons. In real cases, it is important to know what is the general behaviour of the portfolio in addition to the behaviour of single components. Economical resources are distributed over individual software applications but it is important to know if underestimations of portfolio components might be compensated by overestimations of other components or if the errors are of the same nature and sum up to unacceptable levels. "Portfolio error" is calculated adding up all the measurements and estimations and calculating the difference between them as if the set of applications was actually only one bigger application (its measure is the sum of the individual measurements and its approximation is the sum of the individual approximations). This indicator is only a "business" indicator, useful when associated to the other indicators like the following (more traditional) ones.

B. *Prediction at level X*

This indicator – pred (X) - is simply the relative number of estimations (%) that fall inside the range of "actual value" +/- X%. Usually X is set at 25% for model's evaluation. Since the E&QFP method is a "good performer" with respect to the technical accuracy, we decided to lower this threshold to 10%.

C. *Mean Relative Error / Mean Absolute Error*

The mean relative error is the average of the relative errors (with their sign). The mean absolute error is the average of the absolute errors.

D. *Median Relative Error / Median Absolute Error*

The median is the value that is roughly in the middle of the data set. If n is odd, the median is the single value in the middle, namely the value with rank (n + 1)/2. If n is even, there is not a single value in the middle, so the median is defined to be the average of the two middle values, namely the values with ranks n/2 and n/2 + 1. The median value is less sensible to the influence of extreme values with respect to the arithmetic mean. The median could be calculated over the relative values or over the absolute values.

E. *Reliability Indicator*

The reliability indicator *RI* provides a numerical evaluation of the accuracy of the estimation with respect to the corresponding measurement method. This indicator does not express the variability range, but rather the (a posteriori) deviation between the actual measured size value *M* and the estimation range (S_{min}, S_{ml} and S_{max}) – where *ml* means *most likely*. The indicator is defined for non-zero ranges ($S_{min} \neq S_{max}$) – for the estimation of a single system/project *i* – by the following formula:

$$RI_i = \frac{(S_{\max} - S_{\min}) - |M - S_{ml}|}{(S_{\max} - S_{\min})} \qquad (1)$$

The indicator has the following features:

RI_i has a threshold value for *M* equal to one of the range extremis (the smallest value of the two);

RI_i gets worse for M going externally of the estimation range, and vice versa;

RI_i gets better for smaller ranges (S_{max} - S_{min});

RI_i gets better for smaller differences between M and S_{ml} and yields the best value for $M = S_{ml}$.

Figure 3 shows an example of the shape of RI_i *(y-values)*, with fixed values $S_{min} = 10$ FP, $S_{ml} = 13$ FP, $S_{max} = 20$ FP and actual measured value M varying on that range (x-values in Function Points). In the best case ($M = S_{ml}$), we find $RI_i = 1$ (maximum); in the extremis ($M = S_{min}$ o $M = S_{max}$), we find the threshold $RI_i = 0.3$ (=min(0,3;0,7)); for bad estimations (M external to the range), $RI_i < 0.3$. Hence, in this case *the expected value of the reliability indicator, for a satisfactory estimation technique, is between 0.3 and 1*. The closest to 1 is the RI and the better is the estimation.

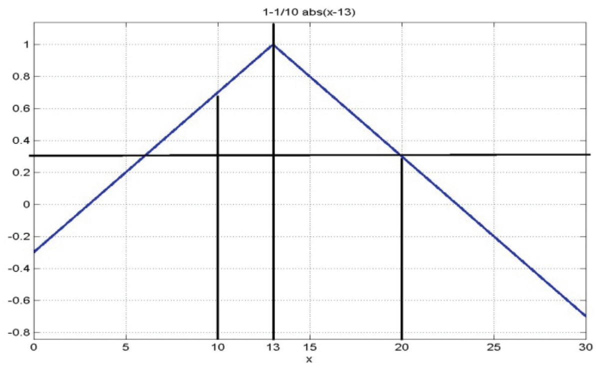


Figure 3. Shape of RI_i with respect to the actual measured size M

The overall reliability indicator of the estimation method is given by the average *RI* over *N* cases. Thus, the average reliability indicator provides, for future estimations, an evaluation of the associated "risk".

V. A VALIDATION EMPIRICAL EXPERIMENT

A validation empirical experiment has been conducted in order to assess the accuracy of the estimation results, using a real life data set of 65 IFPUG FP measurements ranging from 113 FP to 1601 FP (51 baselines; 7 new developments; 8 enhancements). The measurements were taken by Certified Function Point Specialists of the organization originating the data set.

This data set was not used to calibrate the version of the model, it was used to test the "technical accuracy" of the standard model on an independent data set. Details on the data set are reported in the Supplemental Material Section.

A. Measurement and Approximation productivity

TABLE II shows the IFPUG average measurement productivity registered for the 65 cases compared to the approximation productivity of the three level of application of the E&QFP method.

TABLE II - SIZING PRODUCTIVITY

	IFPUG	Detailed	Intermediate	Summary
avg(hours/FP)	45.2	85.7	177.0	284.2
ratio	1.0	1.9	3.9	6.3

B. Universe description

The following TABLE III shows the universe description.

TABLE III – UNIVERSE DESCRIPTION

Universe description	
Average	513.6
Median	403.0
Moda	170.0
Kurtosis	0.88179
Asymmetry	1.18710
Interval	1488
Minimum	113
Maximum	1601
Sum	33384
Count	65

C. Portfolio error

The portfolio error is extremely low as it is shown by the following TABLE IV. This means that using the estimation technique at any level, the total portfolio is estimated in an extremely precise way.

TABLE IV – PORTFOLIO ERRORS

	IFPUG value	Estimation	Difference	%	Abs(%)
Detailed	33384	32466,5	-917,5	-3%	3%
Intermediate	33384	33821,7	437,7	1%	1%
Summary	33384	33732,2	348,2	1%	1%

D. Prediction at level X

The most part of the estimations are beneath the 10% of absolute error as shown in TABLE V. We used an improved version of the typical Pred(25%) due to the high quality of estimations available.

TABLE V – PREDICTION AT LEVEL 10%

	total	<=10%	Pred(10%)
Detailed	65	51	78%
Intermediate	65	55	85%
Summary	65	46	71%

E. *Error magnitudes*

In TABLE VI, we show the minimum, median, average, maximum errors with their sign and as absolute values for the three level of details.

TABLE VI – ERROR MAGNITUDES

	Detailed		Intermediate		Summary	
	error%	abs(error%)	error%	abs(error%)	error%	abs(error%)
min	-18%	0%	-17%	0%	-17%	0%
median	-2%	4%	-2%	5%	-1%	6%
avg	-3%	6%	-1%	6%	-1%	7%
max	20%	20%	27%	27%	24%	24%

F. *Average Reliability Indicator*

The average RI indicator (shown in TABLE VII) is very high in the case of intermediate and summary estimation and low in the case of detailed estimation, due to the very tight range of confidence interval for the detailed estimation. In 15 cases out of 65, the RI was negative since the measured FP value was outside the min-max estimation range, but in none of those cases the absolute error was higher than 20%.

TABLE VII - AVERAGE RELIABILITY INDICATOR

	avg(RI)
Detailed	0,33
Intermediate	0,81
Summary	0,86

G. *Scatterplot diagrams*

Figure 4, 5 and 6 just report, for a visual check, the IFPUG values plotted against the approximated values for the same cases using the three different levels of application of the method.

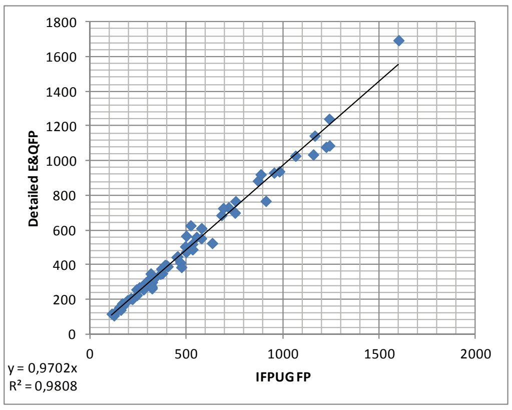


Figure 4. IFPUG FP vs Detailed E&QFP

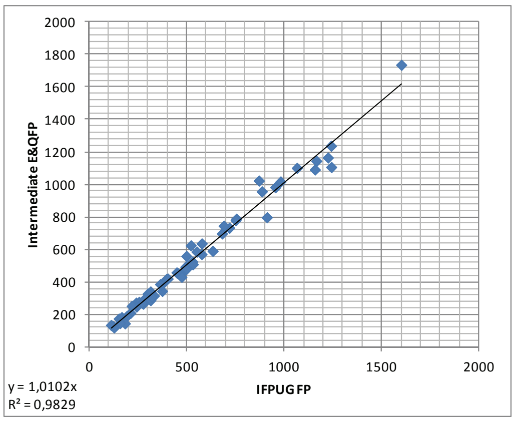


Figure 5. IFPUG FP vs Intermediate E&QFP

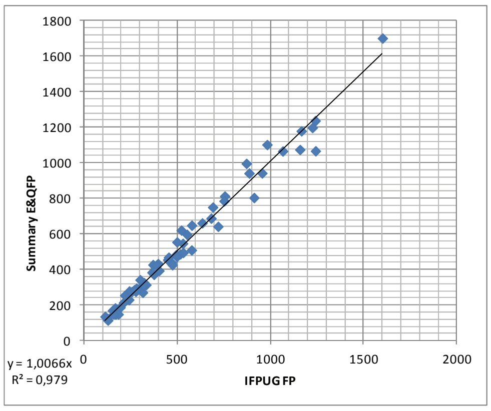


Figure 6. IFPUG FP vs Summary E&QFP

VI. COMPARISON WITH OTHER APPROXIMATION METHODS

In [5], there is a classification of the approximation methods useful to compare E&QFPM to other available methods.

These methods, also known as Algorithmic Model Methods, provide one or more transformation algorithms, which produce a software size estimate as a function of a number of variables, which in turn relate to software attributes. Generally, these methods are correlated to a decomposition process. By decomposing an application into its major functions, estimation can be performed in a step-by-step fashion. This category is further specialized into three main subclasses: Technology Driven Methods, Logic Driven Methods (also called Architecture Driven Methods) and Hybrid Methods.

Technology Driven Methods - This term denotes the derivation of the FP value, for a given software system, from its technical elements, for example Lines of Code or from the number of classes or objects (in an OO environment), physical tables, screen forms, widgets and the like. Using this kind of method, it is not necessary to develop a logical model of the application in terms of functionalities or data entities since the derivation algorithm is a based on statistically derived ratios, i.e., it is based on a statistically proven (hopefully) correlation between a technical measure and a logical measure (the FP count). This kind of method is not reliable due to the significant differences between a physical model and a logical model of software and to the modern programming technologies. An additional problem is that the ratio between technical and logical measures might be easily manipulated in order to "drive" the final FP estimation in a pre-defined direction.

Logic Driven Methods - There are many Logic Driven Methods for estimating FP size, mostly because of statistical research on benchmarking data sets. Not every method is quoted in technical literature, but many of them are widely known in practice. The main characteristic of these methods is that they are based on a "logical model" of the application to be estimated. This means that the model is totally compliant with the Functional Size Model of the IFPUG method. One subset of Logic Driven Methods may be called **Extrapolative Counts**: this kind of methods assumes that we count only one or two FP components (typically the number of Logical Files) of the application, and derive the rest of the count on a statistical or theoretical basis. All of these models should be carefully analyzed, in order to understand their applicability in a particular domain. Very often this method may be customized to reflect the FP distribution of a particular environment instead of a global public available database. This method is simple to use but quite error prone, since a "missing" object may involve a lot of derived FP to be disregarded and the vice versa. As a final consideration, this method needs a very accurate identification of the estimation driver (i.e., ILFs), which is not usually possible at a moment in time when the requirements are still uncertain, vague, approximate and instable. A second type of Logic Driven Methods may be called **Sampled Counts:** using this method, the IFPUG standard count of a part of the entire system is carried out and from this partial count, the rest of the system can be estimated. While in the previous situation the whole system is investigated with respect to some FP components (EI, EO, EQ, ILF or EIF), using the Sampled Count method only a portion of the system is investigated with respect to all the FP components. This method is simple as the previous one but it could be even more error prone since the assumptions about the proportion of the known part over the rest of the system are not really reliable. In addition, there is still the problem of obtaining very detailed information on the "known" part to be counted using the standard procedures and rules in situations where these data might be unavailable. A third type of Logic Driven Methods may be called **Average Complexity Estimation:** this type of method consists in identifying all the IFPUG Base Functional Components – BFC (EI, EO, EQ, ILF, EIF) and assuming an average or most likely complexity for each one of them. This is often a quite precise method but it needs a detailed insight in the software logical requirements as if it was a standard count.

Hybrid Methods – These methods merge technical driven aspects with logical modelling and might accelerate the estimation process but further research is needed to demonstrate their value.

The **E&QFPM** is the most flexible method, since it is based on a mix of approaches including Sampled Counts, Average Complexity Estimation but introduces analogy and multilevel approach. Any of the quoted methods may be considered as specific cases of use of the E&QFPM.

VII. CONCLUSIONS

The analysis of the empirical experiment has confirmed and improved results shown by other studies and current practice. The Early & Quick FP estimation technique is a competitive mean to approximate IFPUG FP values in such a precise way that in many organizations it is used as a primary way to evaluate assets and projects. The effect of expert analogy and domain experience (a potential source of errors in the field) becomes less important as much as the technique is used at the intermediate and detailed level. In these cases, the technical accuracy becomes very close to the operational accuracy. It is important to highlight that the eventuality of committing a large error, in using the E&Q method at a low level of detail of requirements, is largely compensated by the fact that no other approximation technique may be used on summary requirements which are missing the needed details.

ACKNOWLEDGMENT

The experimental data set (IFPUG measures) has been supplied by an Italian public agency and the estimations were made by Luciano Luciani (DPO) IFPUG Certified Function Point Specialist (CFPS) and Certified Early & Quick Function Point Specialist (CEQ-FPS).

REFERENCES

[1] JCGM, JCGM 200:2008 - International vocabulary of metrology — Basic and general concepts and associated terms (VIM), http://www.iso.org/sites/JCGM/VIM/JCGM_200e.html, [retrieved: March, 2015]

[2] ISO/IEC, "14143-1:1998 'Information technology - Software measurement - Functional size measurement - Part 1: Definition of Concepts'", JTC 1 / SC 7, ISO/IEC, 1998.

[3] R. Meli and L. Santillo, "Function Point Estimation Methods: a Comparative Overview", FESMA '99 Conference proceedings, Amsterdam, October 1999.

[4] H. van Heeringen, E. van Gorp, and T. Prins, "Functional size measurement - Accuracy versus costs - Is it really worth it?", SMEF 2009, Rome, Italy, May 2009.

[5] R. Meli, "Functional size approximation: why bother with details ?", IWSM Metrikon MENSURA, Amsterdam, The Netherlands, 2009.

[6] C. Jones, "A new business model for Function Point metrics", August 2009.

[7] L. Lavazza and G. Liu, "An Empirical Evaluation of Simplified Function Point Measurement Processes", Int.

Journal On Advances in Software, vol. 6, n.1/2, pp. 1–13, 2013.

[9] ISO/IEC 20926: 2003, “Software engineering – IFPUG 4.1 Unadjusted functional size measurement method – Counting Practices Manual”, Geneva, 2003.

[10] R. Meli, "Early Function Points : a new estimation method for software projects" - ESCOM 97 - May 1997 - Berlin (Germany).

[11] R. Meli, "Early and Extended Function Point: a new method for Function Points estimation" - IFPUG - Fall Conference - 15-19 September 1997 - Scottsdale, Arizona USA.

[12] CNIPA, Linee guida sulla qualità dei beni e dei servizi ICT per la definizione ed il governo dei contratti della Pubblica Amministrazione - Strategie di Acquisizione delle Forniture ICT v3.4, pp. 205/249, 2005.

[13] R. Meli, A. Abran, V. T. Ho, and S. Oligny, “On the applicability of COSMIC-FFP for measuring software throughout its life cycle”, ESCOM-SCOPE 2000, Munich, April 18-20, 2000.

[14] R. Meli, "Early and Quick Function Point Analysis from Summary User Requirements to Project Management", IFPUG, "IT Measurement - Practical Advice from the Experts", Chapter 26, pp. 417-440, Addison-Wesley, 2002.

[15] M. Conte, T. Iorio, R. Meli, and L. Santillo, "E&Q: An Early & Quick Approach to Functional Size Measurement Methods", SMEF2004, Roma, Italia, January 2004.

[16] L. Santillo, M. Conte, and R. Meli, "Early & Quick Function Point: Sizing More with Less" , 11th IEEE International Software Metrics Symposium, 19-22 September, 2005 Como, Italy.

[8] International Function Point Users Group, “Function Point Counting Practices Manual - Release 4.3.1”, January 2010.

[17] R. Ellafi and R. Meli, "A Source Code Analysis Function Point Estimation Method Integrated with a Logic Driven Estimation Method", SMEF2006, Roma, Italy, May 2006.

[18] T. Iorio, R. Meli, and F. Perna, "Early & Quick Function Points® v3.0: enhancements for a Publicly Available Method", SMEF2007, Italy, May 2007.

[19] DPO, Early & Quick FP Reference Manual, http://www.dpo.it/en/eqfp/risorse.htm, [retrieved: March, 2015].

[20] R. Meli, "Simple Function Point: a new Functional Size Measurement Method fully compliant with IFPUG 4.x", SMEF2011, Roma.

[21] R. Meli, "Simple Function Point! A new method for functional size measurement fully compatible with the IFPUG method 4.x", UK Software Metrics Association & COSMIC International Conference on Software Metrics and Estimating, London , October 2011, London.

[22] SiFPA, "Simple Function Point Functional Size Measurement Method - Reference Manual SiFP-01.00-RM-EN-01.01", http://www.sifpa.org/en/index.htm, [retrieved: March, 2015].

[23] International Software Benchmarking Standards Group, Worldwide Software Development: The Benchmark, (from rel 5, 1999 to rel 12, 2013).

[24] K. Almakadmeh and A. Abran, “Experimental Evaluation of an Industrial Technique for the Approximation of Software Functional Size”, International Journal Of Computers & Technology Vol 10, No 3, 2013.

SUPPLEMENTAL MATERIAL : THE EXPERIMENTAL DATA SET

			E&QFP								
		IFPUG	Detailed			Intermediate			Summary		
N	Type	STD	Min	ML	Max	Min	ML	Max	Min	ML	Max
1	Baseline	577	525.3	552.5	578.3	466.9	571.1	676.4	378.2	509.5	640.3
2	Baseline	1601	1614	1695	1778	1445	1735	2032	1255.2	1702	2148
3	Baseline	755	733.1	766.4	805.7	702	788.6	882.4	588.6	812.7	1037
4	Development	375	334.5	351	367.7	303.9	344.8	387.7	272.1	372.1	472.1
5	Baseline	1225	1027	1080	1133	950.3	1165	1383	895.1	1198	1501
6	Baseline	1241	1188	1242	1304	888.8	1237	1586	888.8	1237	1586
7	Baseline	322	286.1	300.1	315.1	265	306.1	348.9	242.2	325.7	409.1
8	Baseline	634	500.3	524.9	550.9	497.7	591	686.6	490.5	661.7	832.9
9	Baseline	301	295	309.3	324.3	279	328.5	379.1	252.8	342.3	431.8
10	Baseline	212	197.9	206.6	217.2	189.9	209.9	232.6	152.5	213.8	275.1
11	Baseline	241	246.2	257.9	270.9	236.8	272.6	309.9	172.4	230.2	288
12	Baseline	1066	980.8	1028	1080	942	1101	1266	802.8	1066	1330
13	Baseline	256	256.9	269.2	282.9	242.6	277.4	313.3	216.2	275.2	333.8
14	Baseline	955	888.5	931	978	845.6	982.4	1125	693.6	942.3	1191
15	Baseline	1166	1092	1144	1202	1092	1144	1202	933	1179	1425
16	Baseline	370	360.2	376.7	395.8	338.2	381.4	427.8	313	427.3	541.5
17	Baseline	281	269	276.6	284.2	254.5	277.7	301.6	226.1	294.2	362.9
18	Baseline	682	664	685.7	711.5	610.6	698.4	791.8	528	688.1	849.7
19	Baseline	449	424.8	443.9	466.1	414.1	458.6	507.9	329.4	454.9	580.3
20	Baseline	578	580.6	610.4	639.7	519.5	636.6	755.7	480.6	648.5	816.3
21	Baseline	243	215.3	225.2	236.5	221	248.8	279.3	202.9	278.7	354.6
22	Baseline	324	305.2	318.8	335.1	290.9	324.1	360.6	228.5	309.7	391.1
23	Baseline	153	148.9	155.8	163.1	144.8	174.7	205.2	130.6	169.8	209
24	Baseline	170	160.8	167.8	176.6	158.2	172.4	188.7	107.6	148.7	189.8
25	Baseline	290	286.1	298.2	314	284.4	303.1	325.8	204.9	288.7	372.5
26	Baseline	982	895.1	938.7	983.9	864.7	1018	1174	819.9	1103	1385
27	Baseline	315	273.5	286.3	299.9	245.3	288.2	332.8	208.3	271	334.1
28	Baseline	124	102.5	107.6	112.9	106.5	128.3	150.5	91.8	125.5	159.1
29	Baseline	494	480.7	505.5	529.2	404.8	493.4	583.4	360.9	483	604.9
30	Baseline	170	154.2	161.9	169.7	149.3	179.3	210.2	129.9	171.7	213.7
31	Baseline	469	392.9	412.6	432.3	364.1	445.9	529.6	333.3	439.1	545.3
32	Baseline	530	494.2	519.5	543.7	435.6	525.3	616.6	412.2	547	681.4
33	Baseline	454	426.8	448.1	469.5	388.5	455.5	524.6	343.9	469	594.1
34	Baseline	277	245.5	257.6	270.5	226.4	266.3	307.7	210.7	276.5	342.6
35	Baseline	719	695.8	730.7	766.1	626.5	733.5	842.9	481.8	642.4	802.9
36	Baseline	871	842.1	884.5	927.3	855.7	1023	1193	758.8	996.5	1234
37	Baseline	1242	1035	1088	1139	924.9	1107	1292	811.5	1067	1322
38	Baseline	397	378.2	396.9	415.8	355.7	418.7	484	328.3	432.8	537.6
39	Baseline	167	167	176.2	183.8	141	183.2	225.2	141	183.2	225.2
40	Baseline	912	728.6	768	802.3	652.6	797.7	945.1	600.3	804.3	1008
41	Baseline	128	116.9	122.6	128.5	104.2	119.8	136.3	87	114.5	142.2
42	Baseline	194	186	195.1	204.9	154.6	191.7	229.7	138	187.1	236.2
43	Baseline	752	666.9	700.1	733.4	623.2	779.4	938.1	588.4	785.8	983.6
44	Baseline	516	482.5	503.4	525.9	417.7	506.2	595.8	375.1	487.1	598.6
45	Baseline	532	465	488.6	511.8	426.4	508.7	593.2	389	495	601.7
46	Baseline	321	253.4	264	271.8	255.5	316	376.5	236.3	320.3	404.3
47	Baseline	113	112.2	117.4	123.3	97.7	135.5	173.3	97.7	135.5	173.3
48	Baseline	184	175.2	183.6	193.1	124.8	145.2	166.9	107.6	148.7	189.8
49	Baseline	389	380.7	398.7	418.5	351.2	406.4	464.1	327.7	425.1	523.1
50	Baseline	315	332.7	349.2	365.2	290.4	341.4	394	241.4	311.2	380.8
51	Baseline	159	133.9	139.8	147	127	147.5	169.3	107.6	148.7	189.8
52	Development	1158	988.7	1036	1088	917	1091	1272	798.4	1075	1352
53	Development	886	880	921.1	967.6	842.1	957.1	1079	717.5	941.9	1167
54	Development	691	693	726.1	761.8	655	745.5	841.8	579	750.9	923.9
55	Development	552	534	560	587.8	501.1	586.1	673.7	463.3	597.2	731.2
56	Development	522	597.1	626.5	657.9	525.2	624.8	727.8	468.4	621.1	773.8
57	Enhancement	499	539.4	566.5	594.8	478.4	559.7	643.9	449.2	554.5	661.4
58	Development	498	451	473.1	496.2	401.6	479	558.7	349.4	472.4	595.3
59	Enhancement	474	368.1	386.4	405.1	350.3	431.5	513	338.5	426.1	513.8
60	Enhancement	403	374.1	390.8	410.8	382.3	420.8	463.1	297.9	394	490.4
61	Enhancement	365	329.5	347.1	362.9	293	386.4	479.8	287.4	384	480.5
62	Enhancement	333	306.8	321.8	338	271	316	361.6	231.6	314.4	397
63	Enhancement	321	259.3	272.3	285.7	236.5	307.6	378.9	238.9	313.1	387.3
64	Enhancement	219	192	202.1	211.7	191.7	252.4	313.3	189.9	254.6	319.3
65	Enhancement	270	263.4	276.6	290.1	229.7	271.6	314.6	214	286.1	358.1
		33384		32467			33822			33732	

Software Component Allocation on Heterogeneous Embedded Systems Using Coloured Petri Nets

Issam Al-Azzoni

Department of Software Engineering
College of Computer and Information Sciences
King Saud University
Riyadh, Saudi Arabia
Email: ialazzoni@ksu.edu.sa

***Abstract*—In this paper, we present a new approach to component allocation in heterogeneous embedded systems using Coloured Petri Nets (CPNs). While several techniques in optimization exist to solve the component allocation problem, this is the first paper to develop a corresponding CPN model and outline a technique to find an optimal and feasible allocation. The CPN model represents an advancement towards a model-driven engineering view of the problem allowing to subject the model to other types of non-functional analysis. We also exploit the use of CPN Tools, a well-known tool for analyzing CPNs, in generating the state spaces and finding optimal allocations.**

Keywords–Component allocation; Coloured Petri Nets; Model-driven engineering; Embedded systems; Heterogeneous systems.

I. Introduction

Embedded systems have recently become ubiquitous. These systems contain multiple integrated software components and hardware computational units. An embedded system is a computer system, and its associated software, built into some piece of equipment [1]. There are numerous examples where embedded systems are being exploited, including telecommunication systems, household appliances, robots, automobiles, and airlines. These systems are also becoming more heterogeneous with the advent of several types of processors including Central Processing units (CPUs), Graphical Processing Units (GPUs), and Field Programmable Gate Arrays (FPGAs).

This heterogeneity has created new challenges for software architects and designers who decide the placement of the different software components on top of the hardware computational units. While there are many ways to place the components while meeting the functional requirements, the problem becomes much more complex when considering the non-functional (quality) aspects of the placement. For example, a particular mapping of the components may result in better performance than other mappings. The component allocation problem aims to find an allocation (mapping) of the software components such that a certain cost function is optimized. Strategies to solve the problem provide software architects with the necessary tools to make decisions on the allocation of components for heterogeneous embedded systems.

Model-driven engineering (MDE) advocates the use of models in systems analysis and design [2]. The use of models permits various types of analysis to be performed on the models before the actual system is implemented. This can be done at a high level of abstraction and in an automated fashion. While there exist several techniques for solving the component allocation as an optimization problem, this paper presents a new model-based approach for modeling and solving the component allocation problem. The new approach uses Coloured Petri Nets (CPNs) as the modeling language. CPNs have a very rich set of supporting theory and automated tools for model analysis [3]. By modeling the component allocation problem in CPNs, we not only can find an optimal allocation that optimizes a cost function, but also can subject the optimal allocation for other types of non-functional analysis including security and dependability analysis. CPNs have been applied extensively in analyzing non-functional aspects of systems [4][5]. In addition, different approaches exist to transfer other standard software modeling language models into Petri Net models (see the work of [6]).

The contributions of the paper are summarized as follows:

1) We describe a new approach to model the component allocation problem in CPNs.
2) We describe the use of CPN Tools [7] in analyzing the CPN model and solving the component allocation problem.

The organization of the paper is as follows. In Section II, we define the component allocation problem more formally. We illustrate our approach in Section III. In Section IV, we evaluate our CPN based approach using a realistic case study. The related work is discussed in Section V. Section VI concludes the paper and outlines future work.

II. Problem Definition

Consider a software system consisting of n components. Every component needs to be assigned to a computational unit on a hardware platform consisting of m computational units. The computational units offer a number of resources l (for example, computation, memory, and energy resources).

The Component Resource Consumption Matrix $T = [t_{ijk}]_{(n \times m \times l)}$ defines the amount of resources each component requires. The element t_{ijk} represents the necessary amount of the $k-th$ resource required by the $i-th$ software component when allocated on the $j-th$ computational unit.

The Computational Unit Resource Capacity Matrix $R = [r_{jk}]_{(m \times l)}$ defines the amount of resources that each computational unit can provide. The element r_{jk} represents the $k-th$ resource capacity of a $j-th$ computational unit.

An allocation of the components maps each software component to one of the m computational units. Two or more components can be allocated on the same computational unit. From a mathematical viewpoint, an allocation represents a permutation with repetition which assigns one computational unit for each software component. Note that there are m^n possible allocations which implies that the search space increases exponentially with the number of components and computational units.

The component allocation problem is to find an allocation $(p_1, \cdots, p_n)$, where component i is assigned to computational unit p_i, such that it is both feasible and optimal. A feasible allocation means that the resources consumed by the software components allocated on any computational unit do not exceed the resource capacities that the computational unit provides. Thus, the feasibility condition can be stated as follows: given an allocation $(p_1, \cdots, p_n)$, for any computational unit j:

$$\sum_{i,p_i=j} (t_{ip_ik}) \leq r_{jk} \tag{1}$$

for all resources k.

In addition to satisfying (1), we might consider additional constraints that need to be satisfied by a feasible allocation. In this paper, we consider the system architectural constraint that in a feasible allocation a particular component should be (or should not) be allocated on a particular computational unit. There could be several of such architectural constraints that a feasible allocation needs to satisfy.

Given an allocation $(p_1, \cdots, p_n)$, its cost can be computed using the following cost function:

$$w = \sum_{k=1}^{l} f_k \sum_{i=1}^{n} t_{ip_ik} \tag{2}$$

Here, f_k represents a trade-off factor whose purpose is to specify the weights of each resource in the cost function. This allows to differentiate the importance of different resources. An optimal allocation has the smallest w (greater than 0). The component allocation problem is to find an optimal and feasible allocation. Thus, the chosen allocation needs to satisfy (1) (in addition to possibly additional constraints) and has the smallest cost w (greater than 0) which is defined by (2).

The component allocation problem can be formulated as a 0-1 integer linear programming problem which is NP-complete [8]. For exact solutions and small problem sizes (the problem size is based on the number of components and computational units), one can use traditional integer programming techniques. However, for large problem sizes, one needs to resort to heuristics which find good approximations through large space search methods.

III. Approach

In this section, we apply the CPN based approach to solve a component allocation problem using parameters of a realistic system borrowed from [9]. Section III-A gives a brief description of the system. In Section III-B, we develop a CPN model of the system and in Section III-C we describe the generation and analysis of the state space using CPN Tools.

$$\begin{bmatrix} 10 & 90 & 90 & 55 \\ 50 & 20 & 20 & 72 \\ 30 & 20 & 20 & 72 \\ 10 & 40 & 40 & 72 \\ 20 & 40 & 40 & 72 \\ 20 & 50 & 50 & 55 \\ 90 & 20 & 20 & 15 \\ 20 & 10 & 10 & 70 \\ 20 & 10 & 10 & 70 \\ 20 & 15 & 15 & 70 \\ 90 & 10 & 10 & 33 \end{bmatrix} \qquad \begin{bmatrix} 48 & 256 & 256 & 128 \\ 128 & 256 & 256 & 148 \\ 64 & 256 & 256 & 148 \\ 48 & 168 & 168 & 148 \\ 64 & 168 & 168 & 148 \\ 64 & 168 & 168 & 64 \\ 168 & 128 & 128 & 64 \\ 148 & 96 & 96 & 148 \\ 48 & 32 & 32 & 148 \\ 48 & 32 & 32 & 148 \\ 168 & 64 & 64 & 96 \end{bmatrix}$$

(a) (b)

$$\begin{bmatrix} 2 & 18 & 18 & 11 \\ 10 & 4 & 4 & 14 \\ 6 & 4 & 4 & 14 \\ 2 & 8 & 8 & 14 \\ 4 & 8 & 8 & 14 \\ 4 & 10 & 10 & 11 \\ 18 & 4 & 4 & 3 \\ 4 & 2 & 2 & 14 \\ 4 & 2 & 2 & 14 \\ 4 & 3 & 3 & 14 \\ 18 & 2 & 2 & 7 \end{bmatrix}$$

(c)

Figure 1. The component resource consumptions.

A. Case Study

To demonstrate our approach, we borrow the same parameters used to develop a component allocation problem from [9]. The system considered is a software system that handles and interprets vision data on an autonomous underwater vehicle (AUV) while simultaneously interacting with them in real time. That system is being developed as a part of RALF3 project [10].

The system consists of $n = 11$ components. These are: **1-UI** User Interface, **2-CH** Communication Handler, **3-MP** Message Parser, **4-MD** Manual Drive, **5-MM** Mission Manager, **6-MC** Movement Control, **7-V** Vision, **8-AC** Actuator Control, **9-SI** Sensors Layer 1, **10-S2** Sensors Layer 2, and **11-SF** Stream Filtering components. The hardware platform consists of $m = 4$ computational units. These are: **1-mCPU** Mulicore CPU, **2-FPGA** FPGA I, **3-FPGA** FPGA II, and **4-GPU** GPU. There are $l = 3$ resources: average execution time (measured in milliseconds), memory (measured in megabytes), and average energy consumption (measured in milliamperes per hour).

Figure 1 shows the component resource consumptions (*i.e.*, the elements of the matrix T). Since T is three-dimensional (components, computational units, resources), we use three matrices to display three different resources (*i.e.*, the third dimension): (a) average execution time, (b) memory, and (c) average energy consumption. The computational unit resource capacity matrix is given by:

$$R = \begin{bmatrix} 100 & 256 & 50 \\ 150 & 640 & 25 \\ 150 & 640 & 25 \\ 100 & 256 & 15 \end{bmatrix}$$

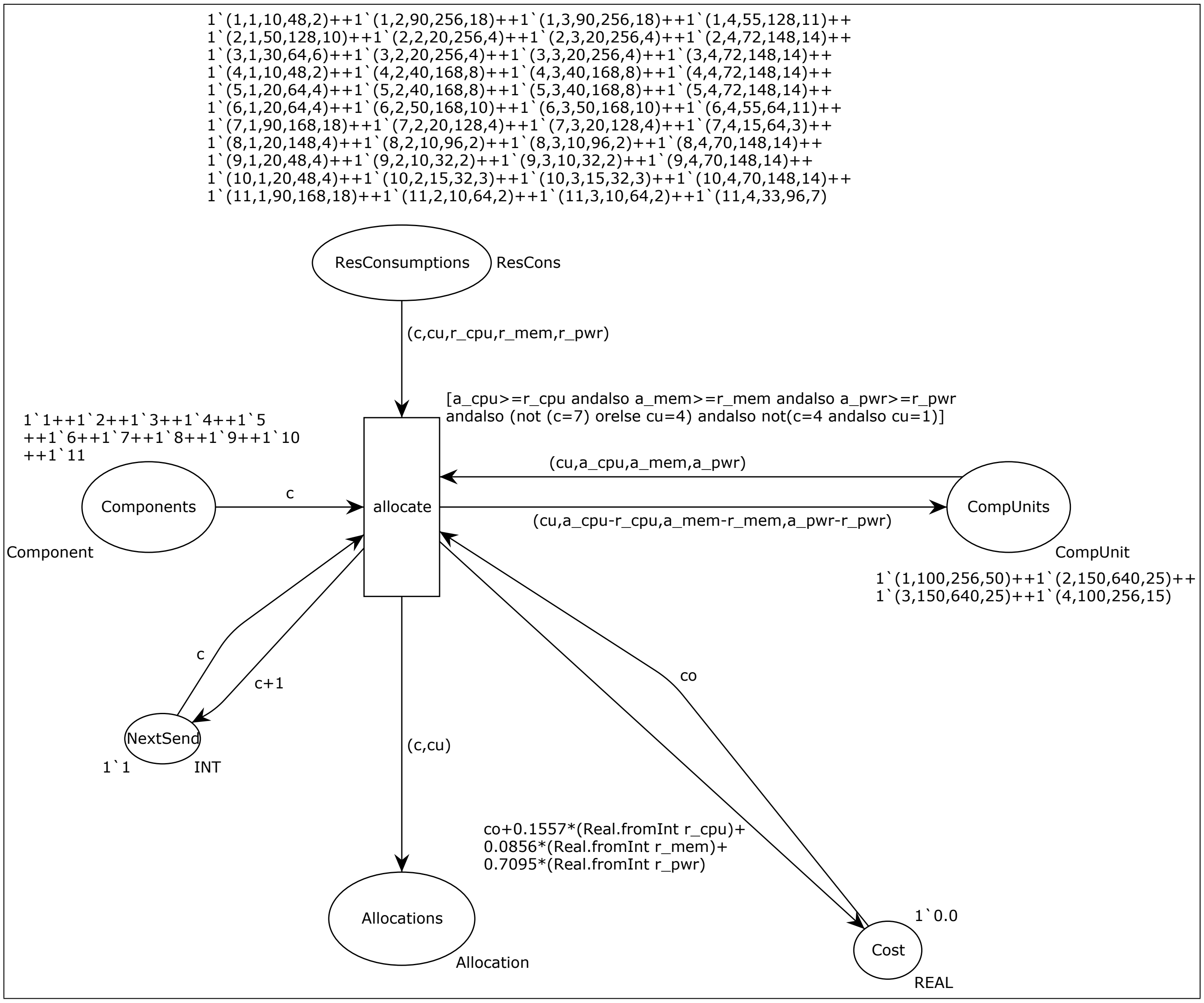


Figure 2. The CPN model for the system of the case study.

To compute the cost of an allocation in (2), we use the trade-off vector:

$$F = [\ 0.1557 \quad 0.0856 \quad 0.7095\ ]$$

Here, the k-th element in vector F represents the trade-off factor f_k. The trade-off factors are computed using Analytic Hierarchy Process (AHP) [11]. The details are given in [9].

We will consider two additional constraints:

- **Constraint I:** Component **7-V** should be allocated on **4-GPU**.
- **Constraint II:** Component **4-MD** should not be allocated on **1-mCPU**.

B. The CPN Model

The CPN model is shown in Figure 2. The CPN contains six places. The place $Components$ holds tokens which represent the components. The place $CompUnits$ holds tokens representing the computational units. Each token records the available resources that the corresponding computational unit currently has. The place $ResConsumptions$ holds tokens which encode the component resource consumption matrix T. The place $Allocations$ holds tokens which represent the allocations of components to computational units. The place $NextSend$ is used to control which component is to be allocated next. The place $Cost$ holds a single token which records the total cost of the allocated components. There is only one transition in the CPN. Firing the transition $allocate$ corresponds to assigning a component to one of the computational units.

The colour sets are defined as follows:

```
colset UNIT = unit;
colset INT = int;
colset REAL = real;
colset BOOL = bool;
colset STRING = string;
```

```
val max_val: real = 2000.0;


fun tot_cost n =
let
val accCostsToken = Mark.model'Cost 1 n;
in
hd(accCostsToken)
end;


fun DesiredTerminal1 n = (Mark.model'Components 1 n == empty);
val x = SearchNodes(EntireGraph, DesiredTerminal1, NoLimit, tot_cost, max_val, Real.min);
fun DesiredTerminal2 n = DesiredTerminal1(n) andalso (Mark.model'Cost 1 n == 1`x);
val y = SearchNodes(EntireGraph, DesiredTerminal2, NoLimit, fn n => n,[],op ::);



CalculateOccGraph();
```

Figure 3. The CPN ML queries used to generate and search through the state space for the CPN model of Figure 2.

```
  colset Component = int;
  colset CompUnit = product INT * INT * INT *
INT;
  colset Allocation = product INT * INT;
  colset ResCons = product INT * INT * INT * INT
* INT;
```

The variables are declared as follows:

```
  var c,cu: INT;
  var co:REAL;
  var a_cpu,a_mem,a_pwr: INT;
  var r_cpu,r_mem,r_pwr: INT;
```

The place $NextSend$ is used to reduce the state space by allocating the components in order of their numbers. This is valid since the order of assigning components to computational units does not matter with respect to the feasibility condition (see (1)).

The constraints are included in the CPN model by using the guard of transition $allocate$. For example, in Constraint I, Component **7-V** should be allocated on **4-GPU**. Thus, a feasible allocation of components should satisfy the condition that $(c = 7) \to (cu = 4)$ which is logically equivalent to $\neg(c = 7) \lor (cu = 4)$. For Constraint II, Component **4-MD** should not be allocated on **1-mCPU**. Thus, a feasible allocation of components should also satisfy the condition that $\neg((c = 4) \land (cu = 1))$. Both conditions are added to the guard of transition $allocate$.

When a component is allocated to a computational unit, the corresponding cost needs to be added to the total cost (the colour of the token in place $Cost$). This is modeled by using the arc from transition $allocate$ to place $Cost$. Note the trade-off factors f_k in the arc expression.

C. State Space Generation and Analysis

We use the state space tool of CPN Tools Version 4.0 to find an optimal and feasible component allocation. CPN Tools Version 4.0 adds the support for real colorsets. Figure 3 shows the query functions used to generate and search through the state space. These queries are written in the CPN ML programming language (presented in Chapter 3 in [3]). For a given marking represented by n, the function tot_cost returns the total cost of the assigned components which is equal to the value (colour) of the token in place $Cost$.

To find the optimal allocations, we use the CPN ML defined function $SearchNodes$ twice. First, we use it to find the minimum value for the total allocation cost over all markings which satisfy the predicate $DesiredTerminal1$. The predicate $DesiredTerminal1$ returns true if and only if the marking represented by n satisfies the condition that there is no token in place $Components$ (hence, all components have been assigned). Thus, the variable x stores the minimum total component allocation cost. The constant max_val is a large real number useful in the start of applying the combination function $Real.min$ of $SearchNodes$. The constant max_val can be set to any large real number, but one should ensure that it is larger than the cost of a single allocation chosen at random. Second, we use $SearchNodes$ to find the markings which satisfy $DesiredTerminal2$. The predicate $DesiredTerminal2$ returns true if and only if the marking represented by n satisfies $DesiredTerminal1$ and that the total allocation cost is equal to x. Thus, the output of the second $SearchNodes$ (stored in variable y) is the list of all markings corresponding to the optimal allocations. The optimal allocations are determined by examining the tokens in place $Allocations$ in any of such markings.

IV. EVALUATION

In this section, we show results of applying our approach on the case study presented in Section III-A. We use CPN Tools to create the corresponding CPN model as developed in Section III-B and analyze the generated state space as outlined in Section III-C.

Table I shows the evaluation results. The table includes the cost of an optimal and feasible component allocation computed by an exhaustive search. In addition, the table shows the optimal and feasible component allocation computed using the

CPN based approach, its cost w, and the time (in seconds) it took CPN Tools to generate the state space. Note that the state space generation was done on a Dell desktop computer equipped with a 3.00GHz dual-core processor and 2GB RAM. The table validates the CPN approach in the case study since the returned component allocation is optimal (its cost is equal to that of the optimal allocation returned by the exhaustive search) and feasible.

Table II shows the evaluation results for the same component model, but excluding **Constraint II**. To exclude this constraint, we remove the corresponding condition from the guard of transition $allocate$. The optimal cost found by the CPN approach is equal to that found by the exhaustive search. Note that the state space search time is almost three times worse than the previous result. This is explained by the increase in the size of the state space due to the exclusion of the constraint.

TABLE I. EVALUATION RESULTS.

Optimal Cost - Exhaustive Search	141.01
Optimal and Feasible Allocation - CPN Approach	(1,3,1,2,1,1,4,3,3,2,3)
Optimal Cost - CPN Approach	141.01
Number of Seconds - CPN Approach	44

TABLE II. EVALUATION RESULTS EXCLUDING CONSTRAINT II.

Optimal Cost - Exhaustive Search	132.23
Optimal and Feasible Allocation - CPN Approach	(1,3,1,1,1,4,4,2,2,2,3)
Optimal Cost - CPN Approach	132.23
Number of Seconds - CPN Approach	128

V. Related Work

The authors of [9] apply a genetic algorithm to find feasible, optimal solutions to the component allocation problem. Our model that defines the component allocation problem is based on the model presented in [9]. However, we do not consider communication costs between components. The authors also apply analytical hierarchical process to deal with the problem of different measurement units in calculating the trade-off factors. Genetic algorithms usually find good solutions; however, generally speaking there is no guarantee that these solutions are the optimal solutions. Compared to the CPN based approach presented in the paper, genetic algorithms scale well for large systems.

Another method for solving the component allocation problem is presented in [12]. The method uses branch-and-bound and forward checking mechanism. The method was implemented in the Automatic Integration of Reusable Embedded Software (AIRS) toolkit [13].

A generic framework aimed at finding the most appropriate deployment architecture (mapping of software components onto hardware resources) for a distributed software system is presented in [14]. The framework formally defines the allocation problem and provides a set of applicable algorithms for solving the problem. In addition, a tool suite is developed to enable the use of the proposed framework. The component allocation problem presented in this paper can be though of as a particular instantiation of the framework. In addition, the CPN based approach can supplement the solution algorithms presented in [14].

The authors of [15] present a formal model for allocation optimization of embedded systems which contains a mix of CPU and GPU processing nodes. The authors use mixed-integer nonlinear programming as the optimization model. In addition, the authors translate the model into a solver using a standard format called MPS (Mathematical Programming System) that can be interpreted using most solvers. The authors make the observation that the mixed-integer nonlinear programming solvers do not scale well for medium and large size problems.

Several approaches exist for component allocation in real-time embedded systems [16][17]. In real-time embedded systems, components (tasks) have additional attributes such that completion time, period, and deadline. The allocation problem for real-time embedded systems needs to ensure that tasks complete before their deadlines. Our CPN based approach uses a different component model which does not take these timing properties into account.

VI. Conclusion and Future Work

This paper has presented a new approach to component allocation using CPNs and CPN Tools. One potential limitation that needs to be considered in the future work is the exponential increase in the generated state space for larger systems. Techniques to scale the applicability of the CPN approach are needed. One approach is to determine an upper bound on the cost and only generate states having cost less than this upper bound. The upper bound can be guessed or can be determined using other optimization methods including genetic algorithms. Also, part of our future work should concentrate on automated methods for model transformation to/from other modeling languages, including the UML Profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE) [18]. Finally, the CPN models need to be analyzed in terms of other non-functional properties such as security and dependability. Future work should apply the new approach and its techniques on several realistic case studies.

References

[1] J. Carlson, J. Feljan, J. Mäki-Turja, and M. Sjödin, “Deployment modelling and synthesis in a component model for distributed embedded systems,” in Proceedings of the Conference on Software Engineering and Advanced Applications, 2010, pp. 74–82.

[2] B. Selic, “Model-driven development: Its essence and opportunities,” in Proceedings of The Symposium on Object and Component-Oriented Real-Time Distributed Computing, 2006, pp. 313–319.

[3] K. Jensen and L. M. Kristensen, Coloured Petri Nets - Modelling and Validation of Concurrent Systems. Springer, 2009.

[4] I. Al-Azzoni, D. G. Down, and R. Khedri, “Modeling and verification of cryptographic protocols using coloured petri nets and Design/CPN,” Nordic Journal of Computing, vol. 12, no. 3, 2005, pp. 201–228.

[5] L. Wells, “Performance analysis using Coloured Petri Nets,” in Proceedings of the Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems, 2002, pp. 217–221.

[6] S. Distefano, M. Scarpa, and A. Puliafito, “From UML to Petri Nets: The PCM-based methodology,” IEEE Transactions on Software Engineering, vol. 37, no. 1, 2011, pp. 65–79.

[7] CPN Tools. http://cpntools.org/ [Accessed December 2014].

[8] R. M. Karp, “Reducibility among combinatorial problems,” in Proceedings of the Symposium on the Complexity of Computer Computations, 1972, pp. 85–103.

[9] I. Švogor, I. Crnković, and N. Vrček, “An extended model for multi-criteria software component allocation on a heterogeneous embedded platform,” Journal of Computing and Information Technology, vol. 21, no. 4, 2013, pp. 211–222.

[10] RALF3 Project Web. http://www.mrtc.mdh.se/projects/ralf3/ [Accessed Aug 2014].

[11] T. L. Saaty, Fundamentals of Decision Making and Priority Theory with the Analytic Hierarchy Process. RWS Publications, 1994.

[12] S. Wang, J. R. Merrick, and K. G. Shin, “Component allocation with multiple resource constraints for large embedded real-time software design,” in Proceedings of the Real-Time and Embedded Technology and Applications Symposium, 2004, pp. 219–226.

[13] AIRES. http://kabru.eecs.umich.edu/bin/view/Main/AIRES [Accessed December 2014].

[14] S. Malek, N. Medvidović, and M. Mikic-Rakic, “An extensible framework for improving a distributed software system’s deployment architecture,” IEEE Transactions on Software Engineering, vol. 38, no. 1, 2012, pp. 73–100.

[15] G. Campeanu, J. Carlson, and S. Sentilles, “Component allocation optimization for heterogeneous CPU-GPU embedded systems,” in Proceedings of the Conference on Software Engineering and Advanced Applications, 2014, pp. 229–236.

[16] J. Fredriksson, K. Sandström, and M. Åkerholm, “Optimizing resource usage in component-based real-time systems,” in Proceedings of the Symposium on Component-based software engineering, 2005, pp. 49–65.

[17] I. Bate and P. Emberson, “Incorporating scenarios and heuristics to improve flexibility in real-time embedded systems,” in Proceedings of the Real-Time and Embedded Technology and Applications Symposium, 2006, pp. 221–230.

[18] OMG, UML Profile for MARTE: Modeling and Analysis of Real-Time and Embedded Systems, version 1.1, formal/11-06-02; June 2011.

Feature Mining for Product Line Construction

Yutian Tang
and Hareton Leung

Department of Computing
The Hong Kong Polytechnic University
Hong Kong SAR, China
Email: {csytang,cshleng}@comp.polyu.edu.hk

***Abstract*—Software product line engineering is a promising approach to generate software assets with systematic reuse property resulting in a significant decrease in development cost. Numerous studies and practical work have proved the reliability, reusability, productivity and the reduced R&D cost attributes of product line engineering. Whereas, the adoption rate of the product line is still relatively low considering the complexity and risk of the task given. It is crucial to have effective approaches to migrate legacy software into product line by mining features in the legacy. Nevertheless, common approaches in feature mining are mainly designed for general systems instead of product line. Therefore, in this paper, we firstly highlight characteristics that a well-designed feature-mining algorithm should contain and pinpoint the shortcomings of existing methods. To enhance the performance of existing approaches for product line, we proposed two feasible directions of research in terms of feature mining for product line.**

Keywords*–*Software product line; Feature mining; Feature location; Reference checking.

I. INTRODUCTION

Software Product Lines (SPL) [1] provides tailored software artifacts with reusable property to stakeholders with minimal R&D effort, considering their general process allows common assets to be shared rather than developing individual systems separately. Among all approaches in generating software product lines, a promising low cost approach should be migrating and refactoring legacy software into a product line, since many fundamentals of legacy could be reused. For the migration, the legacy software has to be reorganized by feature base and then adjusted to a certain pattern to fit the product line. In software product line, variant is a proprietary term for feature and it is designed and specified by domain experts or developers to be either mandatory or otherwise. It assists normalizing the product lines feature model. For instance, in a database product line, a transaction feature should be mandatory for all sub-systems in the product line; however, each sub-system may have its own definition of ranking approach (ranking feature) to process data items.

As reported that successful adoption of the software product line will greatly reduce the cost of generating software products, provide timely service and products to market with reusable characteristics, and decrease the effort for quality assurance. Despite the benefit brought by SPL, the adoption still stays at a low level when compared to other new techniques, including Service Oriented Architecture, Aspect Oriented Programming, and so forth. The underlying reason could be the initial investment, including constructing variants and common assets, in the product line is relative high, and it also costs a lot for developers/software suppliers in terms of risk and complexity [2]. To simplify this process, the variants and common assets could be built by transferring legacy software to product line. The main challenge for migrating is that features in a product line could range from coarse to fine granularity with common and unique purposes like generating various products for different users. For example, in the *MobileMedia* product line, the feature play video is embedded in 24 classes in the system [3]. Therefore, the first and essential step in migrating legacy software into a product line should be locating features and extracting the source code related to the feature concerned given the condition that the code fragments implementing a feature could be scattered in the system instead of concentrating in a single component or file.

In this paper, we will review several existing approaches in feature mining and pinpoint the potential research gaps and the main challenges. Later, we will discuss the lessons learned from our work and other researchers. To focus the research work in feature mining, we will propose two potential directions to consider and investigate with underlying motivations. The contribution of this paper includes following aspects: (1) presenting prospective research gaps in feature mining and shortcomings of current approaches; and (2) proposing some feasible directions for further research.

The rest of this paper is organized as follows: in Section II, we present the essential parts that should be considered in providing feature mining work in product line. Several potential research directions will be presented in Section III. The conclusion will be covered in the last section.

II. FEATURE MINING

Generally, a feature could be represented/defined by a set of code segments, which implement functions of the feature, and assist interaction with other modules (conjunction part). For example, the feature *transaction* in a bank product line cope with a business transaction, which could be a billing or a transfer. Functions belonging to this feature could be *pay_bill*, and *generate_receipt*. In addition, the conjunction part could be functions in charge of acquiring customers bank account information.

Feature mining in product line engineering is highly related to feature location, concern location, and other related fields, within the specific context and constraints in a product line. Developing a feature mining approach for product line is different from designing one for general software products, given the particular circumstance and restriction in product line. Here, we identify several conditions and constraints that will restrict the algorithmic design:

1) Fine granularity design: one significant distinction is that, in product line engineering, a follow-up step is to reconstruct and rewrite legacy code annotated by features into variants for reuse purpose instead of investing on an individual feature. With this constraint, approaches for traditional software products may not be suitable for product line use, as they are designed at the coarse granularity rather than a fine granularity level. Furthermore, the coarse granularity cannot guarantee the consistency of code segments extracted and is not fully adaptable for reconstruction of a product line;
2) Type checking: as reported in [4], type errors are more prone to occur in product lines rather than traditional software. If a product line is ill typed, it may introduce several potential errors during compilation and runtime. These errors include [4]:
    a) *Method invocation*: If a function in class *A* invokes another function in class *B* and the invoked method in *B* is annotated as a variant, deletion of class *B* (removes a variant from a product line, which is allowed in the product line context) will incur undeclared invocation in class *A* as shown in Figure 1;

```
class Painter{
void setPainter(Painter pt,Color col,
    Background bacg){
 canuvs.set(pt,col,bacg.getCurrScope());
}
}
class Canuvs{
#ifdef CANUVS_SET
 boolean set(Painter pt,Color col,Scope scope)
     {....}
#endif
}
```

Figure 1. An example of method invocation type error.

    b) *Referencing types*: Similarly, in referencing-type error, if a class is referenced as a return type or a customized type, when the class is annotated and serves as a variant, the referencing object still remains to be resolved, since the dangling class object will point to null. For the case depicted in Figure 2, if class *Background* is annotated and removed, the object *bacg* will incur a compilation error, since it refers to null;
    c) *Parameters*: Similar to prior case *referencing types*, if an object, which refers to the annotated class, is also annotated, this variant will still fail, since removing this referring object will leave a missing part in the original code slot, which will lead to a compilation error. As shown in Figure 3, in method *set*, the object *bacg* is also annotated as part of variant BACKGROUND, and if variant BACKGROUND is removed from the product line, object *bacg* should be removed as well, which will lead to insufficient parameters for method *set* (three instead of two).

```
class Painter{
 void setPainter(Painter pt,Color col,
     Background bacg){
  canuvs.set(pt,col,bacg.getCurrScope());
 }
}
#ifdef BACKGROUND
class Background{....}
#endif
}
```

Figure 2. An example of referencing type error.

```
class Painter{
 void setPainter(Painter pt,Color col,#ifdef
     BACKGROUND Background bacg #endif){
  canuvs.set(pt,col,bacg.getCurrScope());
 }
}
class Canuvs{
  boolean set(Painter pt,Color col,Scope scope
      ){...}
}
```

Figure 3. Parameter type error.

    d) *Feature interaction*:In general, there are multiple features embedded in a product line. In addition, the connections between features in the feature model show how features can interact and conjunct. Further, in product line, feature dependency will show how features are used and organized. As explained in[5], a practical example of feature dependency is that feature *A* can be selected if and only if feature *B* or *C* is selected without feature *D*, which shows the dependency relation between feature *A*, *B*, *C* and *D* in the feature model.

Next, we will introduce the lessons learned from our study and other related studies. By investigating previous studies in feature mining in a product line, our previous research in feature mining and other approaches in feature location (non product line use) [6], [7], we find following important considerations have been overlooked when preparing a well-defined approach for feature mining:

1) Attention should be given to analyze variability and feature model: Traditional approaches focusing on feature location merely extract features and code segments attached to these features without sufficient effort on feature interaction detection. As known, features do not just interact with each other. Therefore,

detecting interdependencies and interactions from legacy software is still a pending challenge requiring additional work, as most feature location technique neglects this aspect, which is critical for product line engineering. Here, by detecting feature interaction, we mean exploring the interaction from source legacy automatically rather than just defining it in a feature model.

2) Quality assurance: In migrating legacy software into product line, focuses are often on the procedure of migrating without considering the change of quality in this continuous process. Therefore, we believe providing a set of quality metrics to measure the change in quality, such as reusability, reliability, and readability will assist the practitioners in evaluating the results of feature mining.

III. Proposed Directions of further study

As presented above, several conditions and constraints must be considered when developing feature-mining approaches in product line engineering. We identify two possible directions, reference detection and data mining approach, to cope with feature mining in the context of product line engineering. The reason we recommend these directions are twofold: firstly, both could explore and mine features at a fine granularity; and secondly, these approaches are currently not well investigated for this context. The essential idea of these methods is to construct a standard graph to represent the system. Furthermore, the problem needed to be solved, namely, finding code segments for feature concerned, could be altered to discovery a set of nodes in the graph. Reference detection applies a searching strategy to the graph, whereas data mining groups programming elements using the concept of similarity.

Prior to providing concrete description of two potential research directions, we will first present the general procedure of feature mining, which is the infrastructure of all feature-mining approaches on legacy code. The general process of feature mining should consist:

- Defining and describing the features concerned and relationship among them in a model;
- A domain export or developer should identify seeds as starting point for each feature. Seed should be a single or a set of representative programming elements (methods, variables) that stand for the feature;
- For a single feature, the seed chosen along with the feature-mining approach will iteratively inspect related code segments and mark the code segments, which belonging to the same feature, with the same label;
- In the last step, the developers or tools will reorganize or rewrite the code fragments to variants.

In this paper, our work only focuses on the third step, which is providing competitive frameworks to cope with feature-mining task.

A. Reference detection

Reference detection is originally motivated by a simple type of relation in a program named *define-use*, which represents a link between a definition of an instance variable and its later reference to the object [8]. For instance, in a normal object-oriented (OO) language, a single variable could be defined by a statement "*Timer localtimer* = new *Timer*()", which defines a new instance *localtimer* under class *Timer*. A reference statement could be "*this.launch(localtimer)*", which sets the object localtimer as a parameter of launch method. For this scenario, a link should be built between the define-statement and reference-statement in the program. A statement could be set as an attribute with two possible values:*def*, and *ref*. To simplify this procedure, we treat a program as a set of values and operations upon them. Next, we will introduce different types of variables and associated information that should be detected besides the reference relation.

1) *Local variable reference*:A local variable could be declared inside a method or as an argument of a method. For this kind of variable, it could be referenced by other programming elements inside the method and also reference other programming elements inside and outside the method. Specifically, some local variables even have smaller valid boundaries. For example, if a local variable is defined inside a *for* statement block, its valid scope will be within this *for* statement. By checking the location, at which a variable is defined/declared, and the context, it is feasible to obtain the valid scope of the variable.
2) *Instance variable reference*:Instance variables normally hold all attributes of classes and employed as local variables or fields for the class. There are two issues related to the scope concern: the first is the location that this type of instance variable is declared, and the second is whether hierarchical relations exist, which means for a certain class or interface, its super-class, interface, and sub-classes, should also be extracted if any.
3) *Class variable reference*: Class variable is global for all instances of the class and generally its life cycle will finish when the class is destroyed. For a class variable, we should consider the same issues as those of instance variable.
4) *Class/interface field reference*:Fields are the inherent attributes of a class or interface, and are allocated memory when the class/interface is created. The following information should be collected to build the *def-use* link when inspecting a class/interface field: (1) detect the scope of current reference, since a field of a class could be used outside the class/interface in which the field is declared; (2) explore the location where the original class/interface and the field are declared; and (3) build the link between this field reference and instance variable declaration.
5) *Reference variable reference*:When a variable is assigned directly to another variable, these two variables share the same memory location. Thereby, we can use a joint node with two identifiers to represent these two variables, and all reference links to any of them will be redirected to this joint node instead.

After all *def-use* relations are extracted from the source code, we can re-construct and re-build the original program into a standard graph, in which programming elements under reference or defined relations are connected. A simplified example is shown in Figure 4 with some critical information hidden, such as, location information, type information and so forth. After constructing this infrastructure, the original code segments are annotated with the reference/define information. Then, numerous algorithms could be proposed based on this framework by checking the fan-in, fan-out, graph topology structure, ranking connected items by granularity and so forth. This framework offers the following advantages: both fine and coarse-granulated programming elements are encapsulated in the graph and only fine granularity information is required. The rational for keeping both fine and coarse granularity information is to ensure completeness of the program and ease to locate a specific programming element. We propose two approaches to resolve this. One is separating the graph into sub graphs by granularity; and another is setting searching conditions, for instance, statement, expression, variable and so forth, when searching related programming elements in the graph. That is, Abstract Syntax Tree (AST) node could be adapted to work as the node in this framework and the type of AST node could be used to search nodes under a specific granularity.

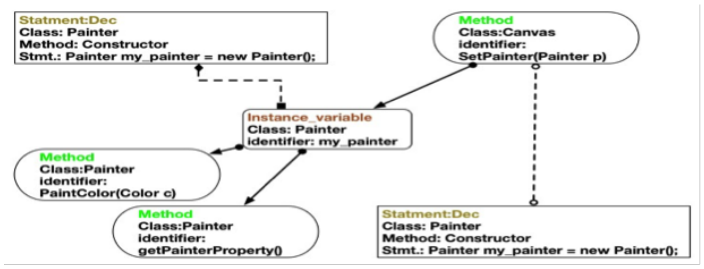


Figure 4. An example of referencing model

Determining the feature boundary under reference detection. Technically, features could be connected by a reference link (e.g. call a function), or physically embedded in the same compiling unit from the code base perspective. Specifically, the reference link could be any type listed above. For this case, algorithm *referenceTrack* can be employed to assist in finding the boundary of a feature:

Algorithm 1 referenceTrack(threshold)

```
for single seeds $s$ do
    create $P_s = \{p|p\, references\, s\}$
end for
for $p \in P$ do
    compute uniqueness value $uv_p$ by $uv_p = \left(\frac{\#\, links\, p\, refereces\, s}{all\, p\, reference\, links} - \frac{\#\, links\, s\, references\, p}{all\, s\, reference\, links}\right) * g(p)$
end for
Rank elements in $P$, according to $uv_p$, and if the max $uv_t > threshold$, it will be annotated into the feature.
The definition statement $dt$ of $t$ will be annotated.Then set $dt$ as $s$.And recall this algorithm with $dt$.
```

Specifically, "# links p references s" represents the number of links from p to s; "# p reference links" shows the number of links p reference others. $g(p)$ shows the granularity of p and its value is designed for showing the interference from the granularity aspect. The algorithm *referenceTrack* will stop in STEP 3, if uv_t is below the threshold assigned.

B. *Data mining based approach*

Apart from the program analysis techniques, machine learning techniques, especially data mining approaches are also recommended for feature mining. Inspiration of using data mining for feature mining comes from the common trait that both attempt to group items into distinct clusters. One basic idea in data mining is to measure the distance between different items to show their "closeness", which is applied to indicate their similarity. In feature mining, this idea should be adjusted to grouping programming elements into different clusters, and each cluster could represent a unique feature. Here, we provide several potential relations that could be used to detect the distance among programming elements:

1) textitReference distance(call graph based):In our model, the reference distance is defined as the number of jumps from one programming element to another and all links connecting intermediate elements are reference relation links.
2) textitControl distance(control graph based):Control graph could be extracted from a program to indicate its control structure. Control distance, in our model, is defined as the jump in a control graph from one programming element to another with the restriction that the distance between two programming elements in the same control branch should be 1.
3) textitText comparsion:Similarity could also be detected from the textual aspect with the assistance of text comparison algorithms[9]. Using different text comparison algorithms, the text distance could be described using the value computed by the token frequency, common strings and so forth.

On the contrary to *def-use* model, the data mining based approach determines closeness from various aspects by building different types of graphs, including call graph and control graph. By utilizing these potential relations, the similarity among programming elements can be determined. Finally, programming elements could be classified into different clusters with each cluster representing a unique feature. In this framework, the seeds selected are used to represent distinct features and other related programming elements are explored iteratively. Here the seed merely serves as the trigger of the algorithm, and is used to detect other potential programming elements.

Determining feature boundary under data-mining approach. Different from reference detection, the mining process for a single feature will stop when one of following stopping criteria is met:

1) For a single node in the graph, if all neighbors (data or control) of this node are annotated by other features, the mining process for this node will stop;
2) A threshold could be set to restrict the selection of programming elements and could be used to stop the mining process for the current node. That is, if and only if the distance computed higher than the set threshold, it will be considered to be added to this feature. If scores of all neighbors (data or control) of the current node are lower than the threshold, the mining procedure for this node will stop.

IV. Conclusion

Despite many research works have investigated feature location and variability detection in product lines, the task of feature mining still poses great challenges, considering the particular circumstance of software product line, complexity of the task, and other special constraints as discussed previously. Mostly, current approaches are insufficient to cope with the fine granularity concern in product line and further detection is required to guarantee feature-mining work flows [8] in legacy software migration. Based on our previous work and other studies in feature mining, we identify key traits that a well-defined product line feature mining approach should contain. The conditions mentioned are highly recommended prior to proposing a well-performing algorithm. With the research gap identified, we proposed two feasible directions to investigate the feature mining. We strongly believe that with general rules and constraints specified in the product line engineering, new approaches can be defined for feature mining in product line and will lead to reduction of risk and effort in transferring legacy software into the product line.

References

[1] K. Pohl, G. Bockle, and F. v. d. Linden, “Software product line engineering foundations, principles, and techniques,” 2005.

[2] C. Catal, “Barriers to the adoption of software product line engineering,” ACM SIGSOFT Software Engineering Notes, vol. 34, no. 6, 2009, pp. 1–4.

[3] E. Figueiredo, N. Cacho, Sant, C. Anna, M. Monteiro, U. Kulesza, A. Garcia, S. Soares, F. Ferrari, S. Khan, F. C. Filho, and F. Dantas, “Evolving software product lines with aspects,” pp. 261–270, 2008.

[4] amp, C. stner, S. Apel, Th, T. m, and G. Saake, “Type checking annotation-based product lines,” ACM Transactions on Software Engineering and Methodology (TOSEM), vol. 21, no. 3, 2012, pp. 1–39.

[5] M. Mendonca, A. Wsowski, and K. Czarnecki, “Sat-based analysis of feature models is easy,” in Proceedings of the 13th International Software Product Line Conference. Carnegie Mellon University, 2009, Conference Proceedings, pp. 231–240.

[6] M. A. Laguna and Y. Crespo, “A systematic mapping study on software product line evolution: From legacy system reengineering to product line refactoring,” Sci. Comput. Program., vol. 78, no. 8, 2013, pp. 1010–1034.

[7] C. Kastner, A. Dreiling, and K. Ostermann, “Variability mining: Consistent semi-automatic detection of product-line features,” Software Engineering, IEEE Transactions on, vol. 40, no. 1, 2014, pp. 67–82.

[8] E. Sderberg, T. Ekman, G. Hedin, and E. Magnusson, “Extensible intraprocedural flow analysis at the abstract syntax tree level,” Science of Computer Programming, vol. 78, no. 10, 2013, pp. 1809–1827.

[9] B. Cleary, C. Exton, J. Buckley, and M. English, “An empirical analysis of information retrieval based concept location techniques in software comprehension,” An International Journal, vol. 14, no. 1, 2009, pp. 93–130.

Detecting Disruption Periods on TCP Servers with Passive Packet Traffic Analysis

Iria Prieto,
Mikel Izal,
Eduardo Magaña
and Daniel Morato

Public University of Navarre
Navarre, Spain
Email: iria.prieto, mikel.izal, eduardo.magana, daniel.morato @unavarra.com

***Abstract*—This paper presents a simple passive algorithm to monitor service availability. The algorithm is based on packet counting over a passive traffic trace of a population of clients accessing servers of interest. The major advantage of the algorithm is that it is passive and thus not invasive while usual monitor systems that can be found on Internet are active probing agents. The proposed system does not communicates to actual servers. It is easy to build as an online monitoring system with no big constraints in software or hardware. It does not relay on a distributed number of network placements for probing agents but works on a single network observing point near network edge. Initial proof of work of the algorithm is presented by analyzing unavailability problems for popular servers at an academic network at Public University of Navarre.**

Keywords–Availability service; network; traffic

I. Introduction

As networks constantly evolve, network application servers are improved in software and hardware in order to cope with the growth of client's demand. In spite of this rapid development, sometimes, clients can not gain access to the servers due to communication problems or server saturation, due to flash crowd demands, human errors, updates, routing failures, etc.

Nowadays, even few minutes unavailability can be critical. For an enterprise offering products to clients through a web server, an interruption of this service means loss sales. Another example which shows the threat of service interruption is the use of an antivirus update server. In case of banks, or other organization where security is a priority, an interruption of the update server entails possible infection problems.

In order to detect when the clients of a network are not being able to successfully use a server application, a wide range of monitoring clients, such as Nagios [1], Zabbix [2], Cacti [3], Munin [4], have been developed. These systems warn the network administrator that a given server of interest is unavailable. These kind of systems work based on active probes, such as ICMP (Internet Control Message Protocol) ping or automatically requesting a server web page in case of monitoring HTTP (Hypertext Transfer Protocol) server. They are required to be installed and configured in monitoring client machines or at the server.

In cases where problems need to be detected at different client networks, at least one client has to be installed on each network. Otherwise, some problems will not be detected, like cases of routing problems in the path from clients to the servers of interest, if the monitoring client may use other route to reach the server.

As it is shown by Liu et al. [5] depending of the location of the system resources the application will achieve more effectiveness. Therefore, depending on where our monitoring clients will be located we would have only the vision of this location. Also, checking the configuration of these monitoring clients can be a problem for multi-tier system where the number of them will be high. In the literature, some papers explore how to face up testing the configuration in these scenarios, [6]. Another problem of taking active measurement across an entire network is that for wide networks it will not be scalable and some paths should be chosen and the rest of statistics inferred through predictive algorithms [7].

On the other hand, active probing can be a problem in high loaded systems or when monitoring third party servers which may not react well to external continuous requests. Nowadays, more and more enterprises rely on public services on Internet that would need to be monitored. In these cases, firewalls and intrusion detectors may deny probes or even ban future normal requests as response to continuous monitoring.

Configuring and using these kinds of distributed monitoring systems is not trivial as shown by different studies on how to approach the problem of monitoring for distributed programs, [8]–[14].

Another disadvantage of active availability monitoring comes from cases where the clients access servers through proxy-caches. In that case, the monitoring client may be requesting a webpage from the server and receiving a response just because it is cached at the proxy system even if the final server is unreachable or has some problem. Thus, the active measurement does not actually check for server availability and other clients in different networks or served by different proxies may be experiencing access problems for the same server. In these cases the system would not detect the problem until the timeout of the cached object. This situation can be addressed by proxy configuration (may not be an option depending on proxy ownership) or crafting requests so they are not cached.

In some cases, due to misconfiguration or network issues, the monitoring client may experience problems to reach the server while actual client access is working, thus giving rise to false positive alerts to the network administrator. The cause of this failures may be things as memory problems or CPU or network overload of the monitoring client. This is often due to the fact that the same agent is probing a large number of servers. Therefore the dimensioning of these clients has to be considered carefully.

Another issue to consider is the reaction time of the monitoring system. The minimum and maximum acceptable time for problem detection has to be decided. Longer times imply slower reaction, smaller times may generate higher overhead and interference to normal clients.

Currently the majority of cloud services available on the Internet offer services over TCP protocol for communications with clients [15], [16]. It has been observed that some servers, due to overload, start refusing new TCP connections by answering with RST packets to clients for some time. In many cases the observed time of these kind of events is on the order of seconds, but usually less than half a minute. After this event the server recovers its normal behaviour and accept again new clients. As stated before, even if it only lasts for seconds this problem may be critical for some businesses, causing user complaints and bad server reputation.

There have been proposals to cope with the downsides of active monitoring. Schatzmann et al. [17] proposed a method to detect temporary unreachability based on flow-level analysis by capturing traces from different routes. Although their method was able to work online the main disadvantage was the need to monitor in different points of a network. Besides, it should be taken into account that the setup of these kinds of measurements is not an easy task [18].

The goal in this work is the development of a simple online disruption detection method for TCP servers. This method avoids active measurement and work just by passive observing network traffic. The proposed method is based on simple packet level counting such as the number of RST and data packet received. It does not require large amounts of memory or CPU power and it is able to detect problems for clients in different networks and for different services without using distributed agents. It will be shown that it is able to detect micro-access-failures with a configurable granularity in the reaction time.

The paper is organized as follows. First of all, the algorithm and configuration parameters are introduced. Section III describes the network scenario used to check the proposed algorithm. Section IV presents the results, comparing it to active detection of popular public services. Finally Sections V and VI present conclusions and future work.

II. Proposed algorithm

As stated in the introduction, the method is based on passive traffic capture. By capturing traffic close to the clients in a given network it will detect when some services will not be available to this community (in this work, the sample community will be the clients at Public University of Navarre network). The main target of the proposed algorithm is to find when a service disruption event has occurred, that means that the clients on the monitored network can not successfully use the service. The server may be down or may just be unreachable from this point due to network or some other problem. In any case this local unavailability is what the network administrator wants to detect more than the global server state. The objective is to detect availability problems, including the case where clients are able to reach the servers but not to use their services. To achieve this, a simple algorithm has been proposed which does not require big hardware or software constraints.

The flow of traffic from the clients to the servers of interest is captured and some simple counters are evaluated every fixed time interval. The counters used are the number of data packets and reset packets sent by the full group of clients and target servers seen during a given (i.e 5 seconds) time interval. Reset packets are TCP protocol packets with RST flag activated. They are used by a TCP endpoint to reject incoming connections and also whenever an abnormal packet is received by a TCP endpoint, to signal to the other side that it should abort the connection. The algorithm bases on the fact that a server sending just TCP RST packets and not any other valid packet to a group of clients during even a small period of time is an indicator of unavailability. Although sometimes it has been observed that the servers finish their connections in an unexpected way such as, sending RST packets to the clients after a client has sent a Fin packet, the algorithm will not show false positives since it will have a high probability that another client will be sending or receiving data packets in the same period. The mechanism consists on dividing time in fixed sized intervals. On every interval the number of packets seen from clients and servers are considered and related to previous interval. When a client sends packets to servers which do not send anything back to it, a server issue is suspected.

If in subsequent seconds the servers keep silent but send reset packets the servers are confirmed as not working. Also, if the client keep sending packets and the servers keep silent it is confirmed as not working. The previous identification idea is built with two simple filter for every interval. On each time interval, counters for clients and servers are updated in order to describe the situations explained before. On the side of the client the counter is the number of packets sent to the servers, regardless if they are data packets or not, $packet\_cli$. On the other hand, on the side of the server, two counters are taken into account: The number of data bytes sent, $bytes\_servers$, and the number of packets with the reset flag activated, $reset\_packets$.

If during a given interval the counters show the client was sending packets but the servers did not send any data packet (even they may send reset packets) the result of the first filter for that time slot is 1. Also the result is 1 when there are no packet sent by the client and the server only sends RST packets. That indicates the server is not answering requests. The second filter would be 1 whenever the result of the first filter of the interval being analazying is 1 and the result of the first filter of the previous interval was also 1. The process can be easily explained through two membership functions, like the ones used in fuzzy logic [19], which are applied in each period. Firstly the used variables are defined:

- x= Number of client packets sent in an interval
- y= Server Bytes sent by the servers in an interval
- z= Number of RST packets sent by the servers in an interval
- i = i^{th} Interval to be analazyed.
- $\psi_i(x, y, z)$= First pass of the compound filter applied in each interval i.

- $\varphi_i(\psi_i, \psi_{i-1})$= Second pass of the compound filter applied in each interval i, it takes into account the result of the first pass.

The two membership functions are described in the equation 1.

$$\psi_i(x,y,z) = \begin{cases} 1 & if \quad ((x>0) \quad and \quad (y=0)) \quad or \\ & ((x=0) \quad and \quad (y=0) \quad and \\ & \quad (z>0)) \\ 0 & Otherwise \end{cases}$$

$$\varphi_i(\psi_i, \psi_{i-1}) = \begin{cases} 1 & if \quad (\psi_i = 1) \quad and \quad (\psi_{i-1} = 1) \\ 0 & Otherwise \end{cases} \tag{1}$$

Each period is labeled with the result of applying the two membership functions, $(\psi_i(x, y, z), \varphi_i(\psi_i, \psi_{i-1}))$. When both results are 1 an availability problem is considered for the duration of both intervals. We define the unavailability period since the first second of the interval labeled as $(1, 1)$ until the next interval labeled as $(0, 0)$. An example of the algorithm operation is shown in Table I

In the second interval of Table I, there was one packet sent by a client but there was no data sent to him by servers so the first flag is 1 and the second one is 0 because it was the first suspected interval. After this first interval, the servers, which belong to Hotmail service, sent 8 packets being all of them TCP RST packets. As there were only reset packet we label this second interval as $(1, 1)$. During the next 5 seconds the servers seem to have recovered because data packets from servers are seen again.

The example is a real case disruption interval detected for Hotmail server at the scenario. During that interval only reset packets where captured from servers and the packet trace was examined to show that servers were closing connections that had been inactive for more than 30 seconds.

These resets were not a response to any observed packet so it seems reasonable that the server was experiencing problems and thus this is the kind of event the algorithm addresses. The main parameter of the algorithm is the time interval duration, that can be chosen by the network administrator depending on the desired reaction time. Smaller values will increase resolution and will detect microfailures but will also increase false positives.

From our experience, values between 5 and 15 seconds are recommended.

III. Network scenario

The algorithm has been developed and tested, detecting availability problems of public internet servers for clients at Public University of Navarre. Captured data comes from author's research group infrastructure who has access to a sniffer with its own software between university main access and academic internet provider (Rediris) as seen on Figure 1. The group has an ongoing packet trace collection campaign since 2004 providing 1Gbps traces from the access of an academic community.

In this work, results are presented from captured data of the week of November 7th to 11th, 2013, checking the availability of popular servers at this community such as Facebook, Yahoo,

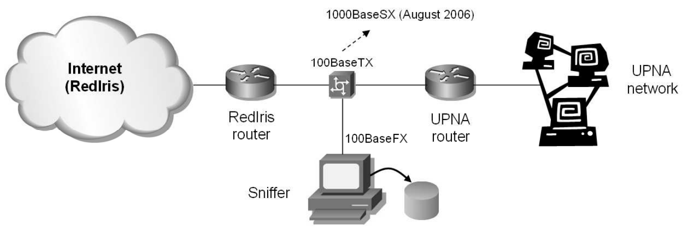


Figure 1. Traffic capturing from a University link

BBC and Hotmail. In order to compare the algorithm against an active monitor (like Nagios [1]), a very basic probing system is implemented. The active monitor tests the availability of selected servers by requesting the site `favicon.ico` file. This file provides an icon to be displayed at browser window and is widely used by web servers. The program requests the favicon file every 5 seconds for every service considered in the experiment and thus provides a ground truth value of availability for comparison purposes.

The active requests are performed from a desktop computer at the university network. The number of servers probed is not very large so the probing computer is not loaded and no request failures can be attributed to machine overloading. The proposed passive algorithm operates on traces obtained at network edge as seen above. It is evaluated offline for the results of this work, but may be easily programmed as an online system.

As servers used are very popular, there are other sources of availability information that were considered. Several web pages provide down times and real time user complaints of public servers but usually this information has not enough time granularity to test less than ten minute disruption events.

IV. Results

In this section results of unavailability detection with a week trace of traffic are presented (November 7th to 11th, 2013). Public servers addressed are: “Yahoo”, “Facebook”, “BBC”, “Hotmail” and also a local newspaper “Diario de Navarra” which are frequently visited by users at the University. Those servers, except the local newspaper, are also used by a large mass of users around the world and they are served by a pool of different IP addresses. They are probably distributed over large server farms or content distribution networks.

But even if those farms are probably designed to balance load and support peaks of demand, sometimes, the clients of the University are not able to reach these services.

Experiments with the basic active monitor that request `favicon.ico` file show the results in Table II for the servers under analysis. Figure 2 shows the events of unavailability with time. The service with more suspected intervals detected was Hotmail.

To test the proposed algorithm the packet trace of a full day is processed and the algorithm is applied on the traffic. The rest of the results are for day 08/11/2013 although other days are similar.

First, the network traffic is filtered to select packets from the probing agent and selected servers of interest. Although this is not the target of this work, addresses of these servers

TABLE I. EXAMPLE OF THE DOUBLE CHECK ALGORITHM DEVELOPED FOR A INTERVAL OF THE DAY 2013/11/8 AND HOTMAIL SERVERS

Start	End	Bytes Serv (x)	RST Serv (z)	Packet Cli (y)	ψ	φ
9:24:55	9:25:00	7016	0	14473	0	0
9:25:00	9:25:05	0	0	1	1	0
9:25:05	9:25:10	0	8	0	1	1
9:25:10	9:25:15	1699	1	3288	0	0

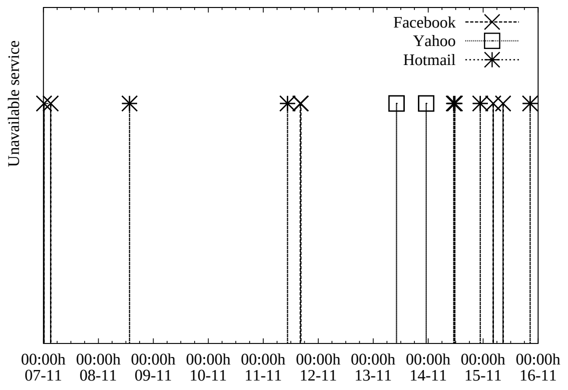


Figure 2. Events of time where the favicon was not be obtained

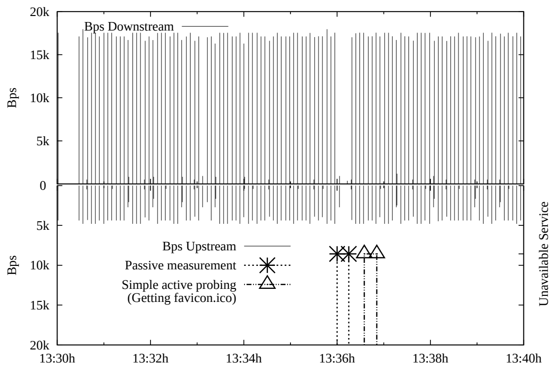


Figure 3. Intervals of time in which the monitoring client had problems for day Nov 8th

TABLE II. UNAVAILABLE SERVICE INTERVALS DETECTED BY REQUESTING THE FAVICON

Start	End	Day	Service
0:15:49	00:16:58	07/11/2013	Facebook
3:10:01	03:10:06	07/11/2013	Facebook
13:36:35	13:36:51	08/11/2013	Hotmail
16:08:12	16:25:40	11/11/2013	Facebook
10:34:59	10:35:21	11/11/2013	Hotmail
10:10:23	10:10:29	13/11/2013	Yahoo
23:08:21	23:08:27	13/11/2013	Yahoo
11:08:54	11:09:06	14/11/2013	Hotmail
11:39:18	11:39:36	14/11/2013	Hotmail
22:43:00	22:43:05	14/11/2013	Hotmail
8:40:31	08:40:44	15/11/2013	Facebook
20:30:03	20:30:13	15/11/2013	Hotmail
4:22:29	04:22:34	15/11/2013	Facebook

have first to be identified. To solve this, the payload of packets is examined to search for these server names in HTTP requests.

Both methods active and proposed algorithm show some unavailability issues for the Hotmail service, see Figure 3. The plot shows the volume of traffic from client machine to Hotmail as well as the time events identified by the passive algorithm and active favicon requester. Both algorithms identified the same event. Packet level examination of the event showed a single connection which suffered an unexpected reset from the server. The comparison also revealed that the time difference is due to the monitor client which was not NTP synchronized as the passive sniffer is. This shows a point to take into account in a distributed monitoring system when monitor clients are distributed time synchronization plays a critical role. The passive sniffer has a unique clock source so the problem of synchronization is simplified.

Packet level analysis of previous event showed the dialog of the packets below. The *x.x.x.x* represents the IP of the client and the *y.y.y.y* the IP of a Hotmail server. After the connection is established, the client sent the request through a push packet of 176 bytes. Usually, after this packet was sent by the client the server answered with the `favicon.ico`. However, in this case the server sent an ACK packet without data and after some seconds, around 11, closed the connection sending a reset packet. This kind of behaviour is unexpected and during these seconds the client would have noticed a malfunction using the service.

```
13:36:02 IP x.x.x.x.59133 > y.y.y.y.http: S
13:36:02 IP y.y.y.y.http > x.x.x.x.59133: S
13:36:02 IP x.x.x.x.59133 > y.y.y.y.http: . ack 1
13:36:02 IP x.x.x.x.59133 > y.y.y.y.http: P 176
13:36:02 IP y.y.y.y.http > x.x.x.x.59133: . ack 177
13:36:13 IP y.y.y.y.http > x.x.x.x.59133: R 1 ack 177
```

Others cases of non-typical reset packets were also observed in the intervals of unavailability studied. In many cases, before a server went down it did not answer to the clients, and after some time it started to send them reset packets to clients since they did not reconignize the previous established connections.

A. Comparison between active probing vs passive analysis unavailability detection method

The total traffic from all the clients using services that previously have been identified to have unavailability periods is analyzed. The objective is to distinguish the periods of time where all the users experience service access problems of the periods of time of isolated problems for individual clients.

To achieve this for each service, all the requested servers are joined together to study if in some period the clients were active but the servers were not working properly. The proposed algorithm is applied to the aggregated network traffic. The algorithm is configured using the IP addresses of all the servers as an unique service to be monitored and a time interval duration of 5 seconds. The unavailability events detected are shown in Figure 4. Interestingly there are more unavailable

TABLE III. UNAVAILABLE HOTMAIL SERVICE INTERVALS

Start	End
09:25:05	09:25:15
10:50:00	10:50:10
14:22:40	14:22:50
14:59:40	14:59:50
15:35:00	15:35:10
15:47:15	15:47:25
16:22:05	16:22:15
17:21:10	17:21:30
19:06:50	19:07:00
19:07:20	19:07:35
19:13:35	19:14:05
19:16:10	19:16:30

periods detected that way than the issues detected by using the favicon requester alone.

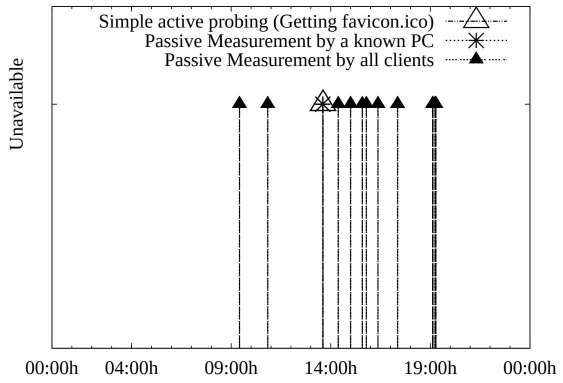


Figure 4. Comparison of the events of unavailable Hotmail service detected from request client for day Nov 8th

The previous event which was observed through the favicon.ico requests and observing the traffic for a single client who requests the favicon.ico, Figure 3, now is not labeled as problematic because at the same time other clients were able to use Hotmail. This interval was a problem of one server giving service to an individual client but it was not a problem of availability for the observed server since other clients were using the same service (other IP addresses of the same service). Thus this is revealed as a false positive warning that shows the risk of using only the monitoring client as a method to detect service failures.

But this experiment show other more important fact. By using the service as an aggregation of individual IP address of servers we are able to identify some unavailability intervals of a few seconds where the clients were suffering access problems but were not detected by active monitoring clients.These periods were not observed by the monitoring client because the favicon.ico was served by a proxy cache. Table III shows all the final disruption events detected.

These periods correspond to the sending of unexpected resets by the severs to the clients. The study of the traffic did not reveal any previously wrong behaviour of the clients which could provoke the send of resets packets by the server. During this seconds, suddenly one or more servers decide to abort the established connections with one or more different clients. As the duration of the intervals were short, these were not actually critical disruptions since the next connections were established. In case that this kind of periods had to be ignored it may be done by just increasing the time interval duration for example to 10 seconds.

TABLE IV. UNAVAILABLE HOTMAIL SERVICE INTERVALS

Start	End
15:34:50	15:35:10
19:13:40	19:14:10

Table IV shows events detected from the same traffic by using an interval duration of 10 seconds. Two cases detected correspond to two intervals of 20 and 30 seconds. During this time there were only reset packets sent from the servers to clients, which have previously completed a connection establishment. Other intervals of 10 seconds are not detected since as the service recovered faster the reset packets sent in order to abort client connections felt inside the same interval as the data packets sent by the servers once that they had recovered. Also, the intervals may not coincide exactly due to interval and event synchronization. The maximum error will be given by the minimum interval of time considered. For example, in the examples presented in this paper, the time of the disruption would be more or less 5 seconds since the interval is said, or 10 seconds when this is the used time interval.

We have checked also the rest of services whose some intervals were detected as unavailable by the monitoring client. The study of the traffic did not reveal any period with problems, there were not any interval of time where the server did not answer to the clients. The periods showed by the monitoring client were due to problems of the own client with the proxy cache or a particular server but not with the service.

B. *Traffic profiling of the requested services*

As a sanity check the full volume of traffic from the scenery network to the servers is observed to check that the amount of traffic was significant. Traffic for the 8th of November to Hotmail service is shown in Figure 5. Hotmail is shown since it is the service with more disruption events detected by the algorithms. The intervals of unavailability detected by the algorithm are drawn also. The first plot shows a full day of traffic and the second one zooms to 1 hour around the previous discussed event.

It can be seen that the amount of traffic suggest the service is working and the gap around 15:35:10 is clearly visible. After these period without traffic the service seem to reestablish normally, creating a traffic peak after the detected problem that reaches almost 5 MBps.

V. CONCLUSIONS

In this paper a simple algorithm to detect periods of unavailability services has been presented. It is based only on passive capture of traffic.

Although there are more service monitoring software available, to the authors best knowledge, they are based on active probing systems. Active monitoring presents some disadvantages which may discourage network administrators of its use in scenarios where the impact of the monitoring needs to be minimized. First, because it requires to check if a service

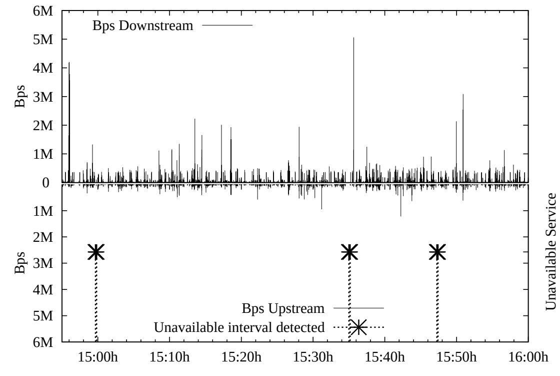


Figure 5. Bps for the use of Hotmail service by the university community

is available it would imply to make periodically requests to different servers. In some scenarios, like high loaded servers or monitoring third party services it is not be possible to make these requests as frequently as needed, in order to avoid overhead or security alarms. Apart from that, the probing requests should be chosen carefully in order to avoid problems with proxy caches which could give the impression that the service is working properly while other clients would not be able to use the service. Another problem is the difficulty to select a location for monitoring clients in multiple subnet scenarios. In these cases, at least a pair of clients should be placed in each subnet in order to detect possible problems inside. Moreover, every client should be clock synchronized in order to report coherent times with the rest of monitoring clients.

As the proposed model is passive and based only on the study of packet counts between servers and clients it will not interfere with the traffic on the network. Therefore, any problem of interference, monitoring client overload, network problems with measured server availability is avoided.

Another advantage of using the proposed model is that it is based in a single location. That means the measure is not dependent on the location of multiple monitoring agents. The network administrator have just to select an appropriate passive observing location, where it can see the traffic between the population of clients to monitor and the servers of interest. This is a much simpler decision that can be typically solved by placing the sniffer at organization's network's edge.

VI. FUTURE WORK

Currently, we are working to extend the algorithm to detect service failures without focusing on specific servers, just by analyzing sniffed traffic and applying the current algorithm to every connection seen. In this manner the algorithm can work as an service anomaly detection system that warns administrator of service issues. This is useful in large organizations that may not have a clear list of services accessed by users but nevertheless need to react to service unavailability problems.

An improvement that can be implemented in order to reduce the number of false positives, is to use the two membership functions described in the algorithm to apply some method of fuzzy logic.

ACKNOWLEDGMENT

This work was supported by the Spanish Ministry of Science and Innovation through the research project INSTINCT (TEC-2010-21178-C02-01). Also, the authors want to thank Public University of Navarra for funding through PIF grant

REFERENCES

[1] "NAGIOS, a commercial-grade network flow data analysis solution," 2009-2015. [Online]. Available: http://www.nagios.com/ [accessed: 2015-01-30]

[2] "ZABBIX, the ultimate enterprise-level software designed for monitoring availability and performance of it infrastructure components," 2001-2014. [Online]. Available: http://www.zabbix.com [accessed: 2015-02-02]

[3] "CACTI, a complete network graphing solution." 2004-2012. [Online]. Available: http://www.cacti.net/ [accessed: 2014-12-29]

[4] "MUNIN, networked resource monitoring tool," 2003-2013. [Online]. Available: http://munin-monitoring.org/ [accessed: 2015-01-15]

[5] X. Liu, J. Heo, L. Sha, and X. Zhu, "Adaptive control of multi-tiered web applications using queueing predictor," in Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP, 2006, pp. 106–114.

[6] D.-J. Lan, P. N. Liu, J. Hou, M. Ye, and L. Liu, "Service-enabled automatic framework for testing and tuning multi-tier system," in e-Business Engineering, 2008. ICEBE '08. IEEE International Conference on, 2008, pp. 79–86.

[7] D. Chua, E. Kolaczyk, and M. Crovella, "Efficient monitoring of end-to-end network properties," in INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE, vol. 3, 2005, pp. 1701–1711.

[8] Y. Park, "Systems monitoring using petri nets," in Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on, vol. 4, 1997, pp. 3245–3248.

[9] C. H. Choi, M. G. Choi, and S. D. Kim, "CSMonitor: a visual client/server monitor for corba-based distributed applications," in Software Engineering Conference, 1998. Proceedings. 1998 Asia Pacific, 1998, pp. 338–345.

[10] C. Steigner, J. Wilke, and I. Wulff, "Integrated performance monitoring of client/server software," in Universal Multiservice Networks, 2000. ECUMN 2000. 1st European Conference on, 2000, pp. 395–402.

[11] G. Song, "The study and design of network traffic monitoring based on socket," in Computational and Information Sciences (ICCIS), 2012 Fourth International Conference on, 2012, pp. 845–848.

[12] G. Fang, Z. Deng, and H. Ma, "Network traffic monitoring based on mining frequent patterns," in Fuzzy Systems and Knowledge Discovery, 2009. FSKD '09. Sixth International Conference on, vol. 7, 2009, pp. 571–575.

[13] A. Tachibana, S. Ano, and M. Tsuru, "Selecting measurement paths for efficient network monitoring and diagnosis under operational constraints," in Intelligent Networking and Collaborative Systems (INCoS), 2011 Third International Conference on, 2011, pp. 621–626.

[14] Y. Bejerano and R. Rastogi, "Robust monitoring of link delays and faults in IP networks," in INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies, vol. 1, 2003, pp. 134–144.

[15] K. Claffy, G. Miller, and K. Thompson, "The nature of the beast: Recent traffic measurements from an Internet backbone," in International Networking Conference (INET) '98. Geneva, Switzerland: The Internet Society, Jul 1998, pp. 1–1.

[16] P. Yang, W. Luo, L. Xu, J. Deogun, and Y. Lu, "TCP congestion avoidance algorithm identification," in Proceedings of the 2011 31st International Conference on Distributed Computing Systems, ser. ICDCS '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 310–321. [Online]. Available: http://dx.doi.org/10.1109/ICDCS.2011.27

[17] D. Schatzmann, S. Leinen, J. Kgel, and W. Mhlbauer, "FACT: Flow-based approach for connectivity tracking," in Passive and Active Measurement, ser. Lecture Notes in Computer Science, N. Spring and G. Riley, Eds. Springer Berlin Heidelberg, 2011, vol. 6579, pp. 214–223.

[18] R. Hofstede, P. Celeda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, "Flow monitoring explained: From packet capture to data analysis with netflow and ipfix," vol. PP, no. 99, 2014, pp. 1–1.

[19] G. Klir and B. Yuan, Fuzzy sets and fuzzy logic. Prentice Hall New Jersey, 1995, vol. 4.

NumEquaRes — Web Application for Numerical Analysis of Equations

Stepan Orlov and Nikolay Shabrov

Computer Technologies in Endineering dept.
St. Petersburg State Polytechnical University
St. Petersburg, Russia
Email: majorsteve@mail.ru, shabrov@rwwws.ru

***Abstract*—A new Web application for numerical simulations, NumEquaRes, is presented. Its design and architecture are motivated and discussed. Key features of NumEquaRes are the ability to describe data flows in simulations, ease of use, good data processing performance, and extensibility. Technical challenges specific to Web applications for simulations, related to performance and security, are discussed. In conclusion, current results are summarized and future work is outlined.**

***Keywords*–Simulation; Web application; Ordinary differential equations.**

I. Introduction

In this work we present a new Web application, NumEquaRes [1] (the name means "Numerical Equation Research"). It is a general tool for numerical simulations available online. Currently, we are targeting small systems of ordinary differential equations (ODE) or finite difference equations arising in the education process, but that might change in the near future — see Section IX.

The reasons for developing yet another simulation software have emerged as follows. Students were given tasks to deduce the equations of motions of mechanical systems — for example, a disk rolling on the horizontal plane without slip [2], or a classical double pendulum [3], — and to try further investigating these equations. While in some cases such an investigation can more or less easily be done with MATLAB, SciLab, or other existing software, in other cases the situation is like there is no (freely available) software that would allow one to formulate the task for numerical investigation in a straightforward and natural way.

For example, the double pendulum system exhibits quasi-periodic or chaotic behavior [3], depending on the initial state. To determine which kind of motion corresponds to certain initial state, one needs the Poincaré map [4] — the intersection of phase trajectory with a hyperplane. Of course, there are ODE solvers in MATLAB that produce phase trajectories. We can obtain these trajectories as piecewise-linear functions and then compute intersections with the hyperplane. But what if we want 10^4–10^5 points in the Poincaré map? How many points do we need in the phase trajectory? Maybe 10^7 or more? Obviously, the simplest approach described above would be waste of resources. A better approach would look at trajectory points one by one, test for intersections with hyperplane, and forget points that are no longer needed. But there is no straightforward way to have simulation process like this in MATLAB.

Of course, there is software (even free software) that can compute Poincaré maps. For example, the XPP (X-Window PhasePlane) tool [5] can do that. But what we have learned from our examples is that we need certain set of features that we could not find in any existing software. These features are as follows:

- ability to explicitly specify how data flows in a simulation should be organized;
- reasonable computational performance;
- ease of use by everyone, at least for certain use cases;
- extensibility by everyone who needs a new feature.

The first of these features is very important, but it is missing in all existing tools we tried (see Section VII). It seems that developers of these tools and authors of this paper have different understanding of what a computer simulation can be. Common understanding is that the goal of any simulation is to reproduce the behavior of system being investigated. Numerical simulations therefore most often perform time integration of equations given by a mathematical model of the system. In this paper, we give the term *simulation* a more general meaning: it is data processing. Given that meaning, we do not think the term is misused, because time integration of model equations often remains the central part of the entire process. Importantly, researcher might need to organize the execution of that part differently, e.g., run initial value problem many times for different initial states or parameters, do intermediate processing on consecutive system states produced by time integrator, and so on.

Given the above general concept of numerical simulation, our goal is to provide a framework that supports the creation of data processing algorithms in a simple and straightforward manner, avoiding any coding except to specify model equations.

Next sections describe design decisions and technologies chosen for the NumEquaRes system (Section II); simulation specification (Section III) and workflow semantics (Section IV); performance, extensibility, and ease of use (Section V); examples of simulations (Section VI); comparison with existing tools (Section VII); technical challenges conditioned by system design (Section VIII). Section IX summarizes current results and presents a roadmap for future work.

II. Design decisions and choice of technologies

Keeping in mind the primary goals formulated above, we started our work. Traditionally, simulation software have been designed as desktop applications or high performance computing (HPC) applications with desktop front-ends. Nowadays, there are strong reasons to consider Web applications instead of desktop ones, because on the one hand, main limitations for doing so in the past are now vanishing, and, on the other hand, there are many well-known advantages of Web apps. For example, our "ease of use" goal benefits if we have a Web app, because this means "no need for user to install any additional software".

Thus we have decided that our software has to be a Web application, available directly in user's Web browser.

Now, the "extensibility by everyone" goal means that our project must be free software, so the GNU Affero GPL v3 license has been chosen. That should enforce the usefulness of software for anyone who could potentially extend it.

The "Reasonable performance" goal has determined the choice of programming language for software core components. Preliminary measurements have shown that for a typical simulation, native code compiled from C++ runs approx. 100 times faster than similar code in MATLAB, SciLab, or JavaScript (as of JavaScript, we tested QtScript from Qt4; with other implementations, results might be different). Therefore, we decided that the simulation core has to be written in C++. The core is a console application that runs on the server and interacts with the outer world through its command line parameters and standard input and output streams. It can also generate files (e.g., text or images).

JavaScript has been chosen as the language for simulation description and controlling the core application. However, this does not mean that any part of running simulation is executing JavaScript code.

The decision to use the Qt library has been made, because it provides a rich set of platform-independent abstractions for working with operating system resources, and also because it supports JavaScript (QtScript) out of the box.

Other parts of the applications are the Web server, the database engine, and components running on the client side. For the server, we preferred Node.js over other technologies because we believe its design is really suitable for Web applications — first of all, due to the asynchronous request processing. For example, it is easy to use HTML5 Server Sent Events [6] with Node.js, which is not the case with LAMP/WAMP [7].

The MongoDB database engine has been picked among others, because, on the one hand, its concept of storing JSON-like documents in collections is suitable for us, and, on the other hand, we do not really need SQL, and, finally, it is a popular choice for Node.js applications.

As of the client code running in the browser, the components used so far are jQuery and jQueryUI (which is no surprise), the d3 library [8] for interactive visualization of simulation schemes, the marked [9] and MathJax [10] libraries to format markdown pages with $\TeX$ formulas. In the future, we are planning to add 3D visualization using WebGL.

III. Simulation specification

The very primary requirement for NumEquaRes is to provide user with the ability to explicitly specify how data flows are organized in a simulation. This determines how simulations are described. This is done similarly to, e.g., the description of a scheme in the Visualization Toolkit (VTK) [11], employing the "pipes and filters" design pattern. The basic idea is that simulation is a *data processing system* defined by a scheme consisting of *boxes* (filters) with *input ports* and *output ports* that can be connected by *links* (pipes). Output ports may have many connections; input ports are allowed to have at most one connection. Simulation data travels from output ports to input ports along the links, and from input ports to output ports inside boxes. Inside each box, the data undergoes certain transformation determined by the box type.

Typically boxes have input and output ports, so they are *data transformers*. Boxes without input ports are *data sources*, and boxes without output ports are *data storage*.

Simulation data is considered to be a sequence of *frames*. Each frame can consist of a scalar real value or one-dimensional or multi-dimensional array of scalar real values. The list of sizes of that array in all its dimensions is called *frame format*. For example, format $\{1\}$ describes frames of scalar values, and format $\{500,400\}$ describes frames of two-dimensional arrays, each having size 500×400. The format of each port is assumed to be fixed during simulation.

Links between box ports are logical data channels, they cannot modify data frames in any way. This means that data format has to be the same at ports connected by a link. Some ports define data format, while some do not; instead, such a port takes format of port connected with it by a link. Thus, data format *propagates along links*. Furthermore, data format can also *propagate through boxes*. This allows to provide quite flexible design to fit the demands of various simulations.

IV. Simulation workflow

This section explains how simulation runs, i.e., how the core application processes data frames generated by boxes.

Further, the main routine that controls the data processing is called *runner*.

A. Activation notifications

When a box generates a data frame and sends it to an output port, it actually does two things:

- makes the new data frame available in its output port;
- *activates* all links connected to the output port. This step can also be called *output port activation*.

Each link connects an output port to an input port, and its activation means sending notification to input port owner box. The notification just says that a new data frame is available at that input port.

When a box receives such a notification, it is free to do whatever it wants to. In some cases, these notifications are ignored; in other cases, they cause box to start processing data and generate output data frames, which leads to link activation again, and the data processing goes one level deeper. For example, the `Pendulum` box has two input ports, `parameters` and `state`. When a data frame comes to `parameters`, the activation notification is ignored (but next time the box will be able to read parameters from that port). When a data frame comes to `state`, the activation is not ignored. Instead, the box computes ODE right hand side and sends it to the output port `oderhs`.

B. Cancellation of data processing

Link activation notification is actually a function call, and the box being notified returns a value indicating success or failure. If link activation fails, the data processing is *canceled*. This can happen when some box cannot obtain all data it needs from input ports. For example, the `Pendulum` box can process the activation of link connected to port `state` only if there are some parameters available in port `parameters`. If it is so, the activation succeeds. Otherwise, the activation fails, and the processing is canceled.

If a box sends a data frame to its output port, and the activation of that output port fails, the box always cancels the data processing. Notice that this is always done by returning a value indicating activation failure, because the box can only do something within an activation notification.

C. Data source box activation

Each simulation must have at least one *data source* box — a box having output ports but no input ports. There can be more than one data source in a simulation.

Data sources can be *passive sources* or *generators*. A generator is a box that can be notified just as a link can be. A passive data source cannot be notified.

A passive data source produces one data frame (per output port) during the entire simulation. The data frame is available on its output port from the very beginning of the simulation.

D. Initialization of the queue of notifications

When the runner starts data processing, it first considers all data sources and builds the initial state of the *queue of notifications*. For each generator, its notification is enqueued. For each passive data source, the notification of each of its links is enqueued.

E. Processing of the queue of notifications

Then the queue is processed by sending the activation notifications (i. e., calling notification functions) one by one, from the beginning to the end. If a notification call succeeds, the notification is removed from the queue. Otherwise, if the notification call fails (i.e., the data processing gets canceled), the notification is moved to the end of the queue, and the process continues.

The runner processes its queue of notifications until it becomes empty, or maximum number of activation notification failures (currently 100) is exceeded. In the latter case, the entire simulation fails.

F. Post-processing

When the queue of notifications becomes empty, the runner can enqueue *post-processors* before it stops the data processing. The only example of a post-processor is the `Pause` box. Post-processors, like generators, are boxes that can receive activation notifications.

G. User input events

The above process normally takes place during the simulation. In addition, there could be events that break the processing of the queue of notifications. These events are caused by *interactive user input*. Once a user input event occurs, an exception is thrown, which leads to the unwinding of any nested link activation calls and the change of the queue of notifications. Besides, each box gets notified about simulation restart.

The queue of notifications is changed as follows when user input occurs. First, the queue is cleared. Then one of two things happens.

- If the box that threw the exception specifies which box should be activated after restart, the notifications for that box are enqueued (if the box is a generator, its activation notification is enqueued; otherwise, the activation notifications of all links connected to its output ports are enqueued). An input box can only specify itself as the next box to activate, or specify nothing.
- If the box that threw the exception specifies no box to be activated after restart, the standard initialization of the notification queue is done.

After that, the processing of notification queue continues.

There is an important issue that must be taken care of. Simulation can potentially be defined in such a way that its execution leads to an infinite loop of recursive invocation of activation notifications. This normally causes program to crash due to stack overflow. In our system, however, some boxes (not all, but only those activating outputs in response to more than one input notification) are required to implement counters for recursive call depth. When such a counter reaches 2, simulation is considered to be invalid and is terminated. This allows to do some kind of runtime validation against recursion at the cost of managing call depth counters.

V. Performance, extensibility, and ease of use

As stated in Section I, computational performance and functional extensibility are considered important design features of the NumEquaRes system. This section provides technical details on what has been done to achieve performance and support extensibility. Last subsection highlights design features that make system easier to use.

A. Performance

To achieve reasonable performance, it is not enough to just use C++. Some additional design decisions should be made. Most important of them are already described above. The ability to organize simulation workflow arbitrarily allows to achieve efficient memory usage, which is illustrated by an example in Section I. A number of specific decisions made in the design of NumEquaRes core are targeted to high throughput. They are driven by the following rules.

- Perform simulation in a single thread. While this is a serious performance limitation for a single simulation, we have made this decision because the simulation runs on the Web server, and parallelization inside a single simulation is likely to impact the performance of server, as it might run multiple simulations simultaneously. And, on the other hand, single thread means no synchronization overhead.
- No frequent operations involving interaction with operating system. Each box is responsible for that. For example, data storage boxes should not write output data to files or check for user input frequently. The

performance might drop even if the time is measured using `QTime::elapsed()` too frequently.

- No memory management for data frames within activation calls. In fact, almost 100% of simulation time is spent in just one activation call made by runner (during that call, in turn, other activation calls are made). Therefore, memory management outside activation calls (e.g., the allocation of an element of the queue of notifications) is not a problem. Still some memory allocation happens when a box writes its output data, but this is not a problem as well, since such operations are not frequent.
- No movement of data frames in memory. If a box produces an output frame and makes it available in its output port, all connected boxes read the data directly from memory it was originally written to. This item and the previous one both imply that there are nothing like queues of data frames, and each frame is processed immediately after it is produced.
- No virtual function calls within activation calls. Instead, calls by function pointer are preferred.

A simple architecture of classes has been developed to comply with the rules listed above and, in the same time, to encapsulate the concepts of box, port, link, and others. These classes are split into ones for use at the initialization stage, when simulation is loaded, and others for use at simulation run time. First set of classes may rely on Qt object management system to support their lifetime and the exposure of parameters as JavaScript object properties. Classes of the second set are more lightweight; their implementations are inlined whenever possible and appropriate, in order to reduce function call overhead.

Although NumEquaRes core performance has been optimized in many aspects, it seems impossible to combine speed and flexibility. Our experience with some examples indicates that hand-coded algorithms run several times faster than those prepared in our system.

B. *Extensibility*

The functionality of NumEquaRes mostly resides in boxes. To add a new feature, one thus can write code for a new box. Boxes are completely independent. Therefore, adding a new one to the core simply boils down to adding one header file and one source file and recompiling. The core will be aware of the presence of the new box through its box factory mechanism. Next steps are to support the new box on server by adding some meta-information related to it (including user documentation page) and some client code reproducing the semantics of port format propagation through the box. The checklist can be found in the online documentation.

Some extensions, however, cannot be done by adding boxes. For example, to add 3D visualization, one needs to change the client-side JavaScript code. We are planning to simplify extensions of this kind; however, this requires refactoring of current client code.

C. *Ease of use*

First of all, NumEquaRes is an online system, so user does not have to download and install any software, provided user already has a Web browser. All user interaction with the system is done through the browser.

To formulate a simulation as a data processing algorithm, user composes a scheme consisting of boxes and links, and there is no need to code.

Online help system contains a detailed documentation page for each box; it also explains simulation workflow, user interface, and other things; there is one step-by-step tutorial.

To prepare a simulation, user can find a similar one in the database, then clone it and modify. User can decide to make his/her simulation public or private; public simulations can be viewed, run, and cloned by everyone. To share a simulation with a colleague, one shares a hyperlink to it; besides, simulations can be downloaded and uploaded.

Currently, user might have to specify part of simulation, such as ODE right hand side evaluation, in the form of C++ code. We understand this might be difficult for people not familiar with C++. To mitigate this problem, there are two features. Firstly, each box that needs C++ code input provides a simple working example that can be copied and modified. Secondly, NumEquaRes supports the concept of *code snippets*. Each piece of C++ input can be given a documentation page and added to the list of code snippets. These snippets can easily be reused by everyone.

VI. Examples of simulations

This section lists several examples of simulations.

Figure 1 shows one of the simplest simulations — the one that plots a single phase trajectory for a simple pendulum. The ODE system is provided by the `ode` box. NumEquaRes has a number of options for user to supply equations. In particular, it is possible to provide C++ code that computes ODE right hand side. Such code compiles and runs on the server, if it passes a security check. The ODE right hand side depends on the state variables and the vector of parameters. They are supplied through input ports. Parameters are specified in the `odeParam` box. State variables come from the `solver` box. The solver performs numerical integration of the initial value problem, starting from the user-specified initial state (the `initState` box). The solver can be configured to perform a fixed number of time steps or to run until interrupted by a data frame at its `stop` port. Each time the solver obtains a new system state vector, it sends the vector to its `nextState` port. Once the solver finishes, it activates the `finish` port to let others know about it. In this simulation, consecutive system states are projected to the phase plane (the `proj` box) and then rasterized by the `canvas` box. Finally, the data comes to the `bitmap` box that generates the output image file. Notice that this simulation has three data sources, `odeParam`, `solverParam`, and `initState`.

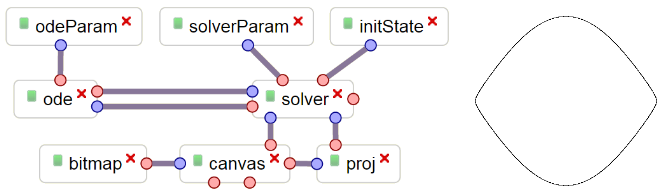


Figure 1. Single phase trajectory

From this simplest example one can see how to construct simulation scheme from boxes and links that computes what user needs. Other examples are more complex, but they basically contain boxes of the same types, plus probably some more. So far, there are 40 different box types in NumEquaRes, and it is beyond the scope of this article to describe them all. Further, we will just focus on some of them to show how simulations work.

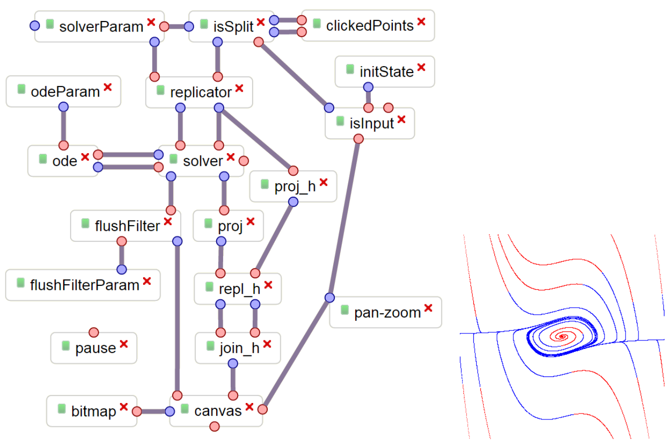


Figure 2. Interactive phase portrait

An important aspect of a simulation is its ability to *interact with the user*. There are a few boxes that transform various kinds of interactive user input (clicking, moving sliders, rotating mouse wheel, etc.) into numerical values. These boxes usually act as simple filters of data frames; they replace some components of data frames with values obtained from user. Figure 2 shows an example of interactive simulation: it generates phase trajectories going through points on plane — the ones user has clicked with the mouse. The box `isInput` is responsible for that kind of input. Each generated phase curve has two parts: blue in the time-positive direction (with resp. to the clicked point) and red in the time-negative direction.

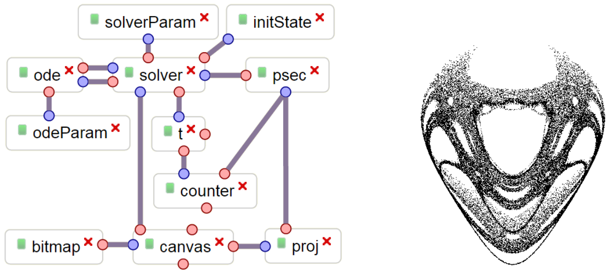


Figure 3. Double pendulum, Poincaré map (50000 points, 28.5 s)

Figure 3 shows the Poincaré map for the classical double pendulum system. Importantly, there is no need to store phase trajectory or individual points of intersection of the trajectory with the plane during simulation. The entire processing cycle (test for intersection; projection; rasterization) is done as soon as a new point of the trajectory is obtained. After that, we need to store just one last point from the trajectory. Simulations like this are what we could not do easily in MATLAB or SciLab, and they have inspired us to develop NumEquaRes.

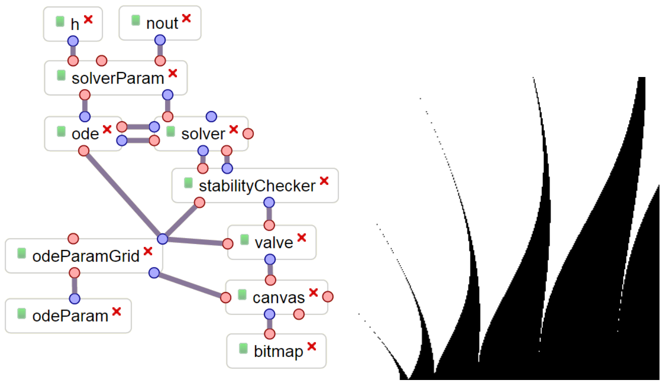


Figure 4. Ince-Strutt stability diagram (500 × 500 points, 6.3 s)

Figure 4 shows a simple simulation that allows one to obtain a stability diagram of a linear ODE system with periodic coefficients on the plane of parameters. Here the picture on the right is the Ince–Strutt diagram for the Mathieu equation [12]. People who have experience with it know how difficult it is to build such kind of diagrams analytically, even to find the boundaries of stability region near the horizontal axis. What we suggest here is the brute force approach — it is fast enough, general enough, and it is done easily. The idea is to split the rectangle of parameters into pixels and analyze the stability in the bottom-left corner of each pixel (by computing eigenvalues of the monodromy matrix [4]), then assign pixel color to black or white depending on the result. In this simulation, important new boxes are `odeParamGrid` and `stabilityChecker`. The former one provides a way to generate points on a multi-dimensional grid, and the latter one analyzes the stability of a linear ODE system with periodic coefficients.

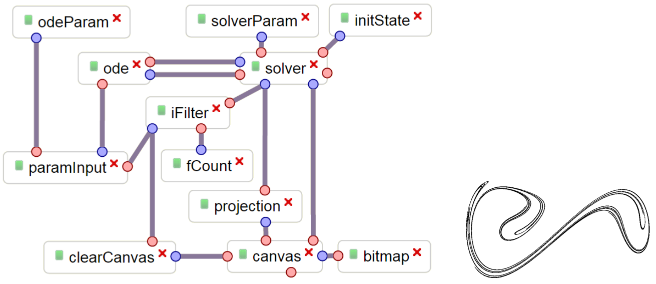


Figure 5. Strange attractor for forced Duffing equation (interactive simulation)

Figure 5 shows another application of the Poincaré map, now in the visualization of the strange attractor arising in the forced Duffing equation [13]. User can change parameters interactively and see how the picture changes. This simulation is simpler than the one shown in Figure 3, because to obtain a new point on canvas, one just needs to apply time integration over known time period of system excitation.

Figure 6 shows an interactive simulation of the Mandelbrot set [14]. User can pan and zoom the picture using the mouse. Importantly, we did not have to develop any new box types in order to describe the logic of convergence analysis for sequences of complex numbers generated by the system. This

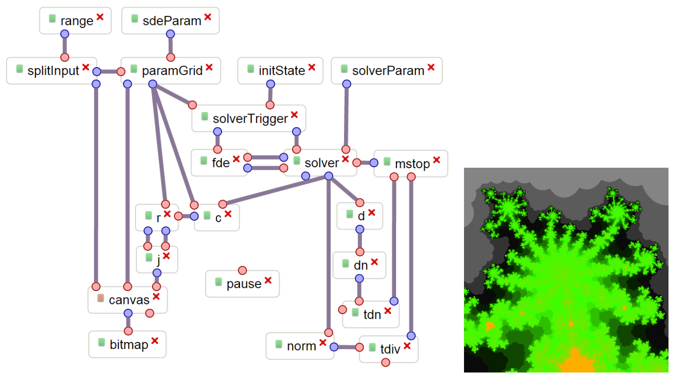


Figure 6. Colored Mandelbrot set (interactive simulation)

is done with general purpose boxes `d` (computes differences of subsequent data frames), `dn` (computes vector norm), and `tdn` (detects if a scalar value exceeds some threshold). Pixel colors depend on how many iterations passed (box `c` is a counter, its output value is joined with pixel coordinates at box `j` and sent to `canvas`).

VII. Comparison with other tools

Direct comparison between NumEquaRes and other existing tools is problematic because all of them (at least, those that we have found) do not provide an easy way for user to describe the data processing algorithm. In some systems, the algorithm can be available as a predefined analysis type; in others, user would have to code the algorithm; also, there are systems that need to be complemented with external analysis algorithms.

Let us consider example simulations shown in Figures 3, 4, 5, and try to solve them using different free tools; for commercial software, try to find out how to do it from the documentation. Further in this section, figure number refers to the example problem.

TABLE I. Comparison of NumEquaRes with other tools

Name	*Free*	*Web*	*Can solve*	*Fast*
Mathematica	no	yes	3, 4, 5; needs coding	n/a
Maple	no	no	3, 4, 5; needs coding	n/a
MATLAB	no	no	3, 4, 5; needs even more coding	no
SciLab	yes	no		no
OpenModelica	yes	no	none	could be
XPP	yes	no	3, 5	yes
InsightMaker	yes	yes	none	n/a

In Table I, commercial proprietary software is limited to most popular tools — Mathematica, Maple, and MATLAB. In many cases, purchasing a tool might be not what a user (e.g., a student) is likely to do.

All of the three example simulations are solvable with commercial tools Mathematica, Maple, and MATLAB.

In Mathematica, it is possible to solve problems like 3, 5 using standard time-stepping algorithms since version 9 (released 24 years later than version 1) due to the `WhenEvent` functionality. Problem 4 can also be solved. All algorithms have to be coded. Notice that Wolfram Alpha [15] (freely available Web interface to Mathematica) cannot be used for these problems.

Maple has the `DEtools[Poincare]` subpackage that makes it possible to solve problem 3 and others with Hamiltonian equations; problems 4, 5 can be solved by coding their algorithms.

With MATLAB or SciLab, one can code algorithms for problems 4, 5 using standard time-stepping algorithms. For problem 3, one needs either to implement time-stepping algorithm separately or to obtain Poincaré map points by finding intersections of long parts of phase trajectory with the hyperplane. Both approaches are more difficult than those in Mathematica and Maple. And, even if implemented, simulations are much slower than with NumEquaRes.

OpenModelica [16] is a tool that helps user formulate the equations for a system to be simulated; however, it is currently limited to only one type of analysis — the solution of initial value problem. Therefore, to solve problems like 3, 4, 5, one has to code their algorithms (e.g., in C or C++, because the code for evaluating equations can be exported as C code).

XPP [5] provides all functionality necessary to solve problems 3, 5. It contains many algorithms for solving equations (while NumEquaRes does not) and is a powerful research tool. Yet it does not allow user to define a simulation algorithm, and we have no idea how to use it for solving problem 4.

Among other simulation tools we would like to mention InsightMaker [17]. It is a free Web application for simulations. It has many common points with NumEquaRes, although its set of algorithms is fixed and limited. Therefore, problems 3, 4, 5 cannot be solved with InsightMaker.

VIII. Technical challenges

The design of NumEquaRes governs technical challenges specific to Web applications for simulations. They are related to performance and security, and are discussed in this section.

A. *Server CPU resources*

Currently, all simulations run on the server side. Some of them can be computationally intensive and consume considerable amount of CPU time. For example, there are simulations that consume 100% of single CPU core time for as long as user wishes. This is a problem if the number of users grows. Of course, we do not expect millions of users simultaneously running their simulations, but still there is a scalability problem.

The problem can be addressed in a number of ways. Firstly, the server can be an SMP computer, so it will be able to run as many simulations as the number of CPU cores, without any loss of performance. Secondly, it is technically possible to have a cluster of such computers and map its nodes to user sessions. Obviously, this approach requires the growth of server hardware to provide sufficient server performance.

A different approach is to move running simulations to the client side. In this case, the server loading problem will disappear. But how is it possible to offer user's browser to run something? Actually, today the only choice seems to be JavaScript. We will have to compile simulations into it, or to the asm.js subset of JavaScript. This approach is quite possible for some simulations, but is problematic for other ones that can make use of some large libraries like LAPACK.

B. User code security

NumEquaRes web server accepts C++ code as part of simulation description provided by user. This is the direct consequence of our wish to provide good computational performance of simulations. Such pieces of code typically describe how to compute the right hand side of an ODE system, or how to compute another transformation of input data frames into the output data frames. The server compiles that code into dynamic library to be loaded and executed by core application that performs the simulation. Potentially, we have serious risk of direct execution of malicious code.

Currently, this problem is solved as follows. Once user code is compiled into a library (shared object on UNIX or dynamically linked library on Windows), it is checked for the absence of any external or weak symbols that are not found in a predefined white list (the list contains symbol names for mathematical functions and a few more). Due to this, user code is not able to make any system calls. For example, it cannot open file `/etc/passwd` and send it to the user because it cannot open files at all. If the security check on the compiled library fails, no attempt to load it is done, and the user gets notified about the reason of check failure.

On the other hand, malicious code could potentially exploit such things as buffer overrun and inline assembly. It is an open problem now how to ensure nothing harmful will happen to the server due to that. However, the ban on any non-white-listed calls seems to be strong enough. Probably, one more level of protection could be achieved with a utility like chroot.

A better approach to provide security is to disallow any C++ code provided by user. But this would imply giving the user a good alternative to C++ allowing to describe his/her algorithms equally efficiently. For example, there could be a compiler of formulas into C++ code. Nothing like this is implemented at the moment, but can be done in the future. In this case, the user code security problem will vanish.

IX. Conclusion and future work

A new tool for numerical simulations, NumEquaRes, has been developed and implemented as a Web application. The core of the system is implemented in C++ in order to deliver good computational performance. It is free software and thus everyone can contribute into its development. The tool already provides functionality suitable for solving many numerical problems, including the visualization of Poincaré maps, stability diagrams, fractals, and more. Simulations run on server; besides, they may contain C++ code provided by user. This creates two challenges — potential problems of server performance and security. The security problem has been addressed in our work; the performance problem is not currently taken into account.

The algorithm of simulation runner implies that the order of activation calls it makes is not important, i.e., does not affect simulation results. While this is true for typical simulations, counter-examples can be invented. Further work is to make it possible to distinguish such simulations from regular ones and render them invalid.

NumEquaRes is a new project, and the current state of its source code corresponds more to the proof-of-concept stage than the production-ready stage, because human resources assigned to the project are very limited. To improve the source code, it is necessary to add developer documentation, add unit tests, and deeply refactor both client and server parts of the Web interface.

Further plans of NumEquaRes development include new features that would significantly extend its field of application. One of them is an engine helping user to formulate mathematical model equations — for example, for mechanical modeling of multibody systems. To simulate these models, more advanced time-stepping algorithms should be implemented.

Another set of planned features aims to enhance the level of presentation of simulation results (currently, it is quite modest). Among them is 3D visualization and animation.

Last but not least, an important usability improvement can be achieved with a feature that visualizes simulation data flows; its role is similar to debugger's.

References

[1] "Numequares — an online system for numerical analysis of equations," URL: http://equares.ctmech.ru/ [accessed: 2015-02-21].

[2] E. J. Routh, The Advanced Part of a Treatise on the Dynamics of a System of Rigid Bodies, 6th ed. Macmillan, London, 1905, reprinted by Dover Publications, New York, 1955.

[3] L. Meirovitch, Elements of vibration analysis. New York: McGraw-Hill, 1986.

[4] G. Teschl, Ordinary Differential Equations and Dynamical Systems, ser. Graduate studies in mathematics. American Mathematical Soc., URL: http://books.google.ru/books?id=FSObYfuWceMC [accessed: 2015-02-21].

[5] B. Ermentrout, Simulating, Analyzing, and Animating Dynamical Systems: A Guide to XPPAUT for Researchers and Students, ser. Software, Environments and Tools. Society for Industrial and Applied Mathematics, 2002, URL: http://books.google.ru/books?id=Qg8ubxrA060C [accessed: 2015-02-21].

[6] "Using server-sent events," URL: https://developer.mozilla.org/en-US/docs/Server-sent_events [accessed: 2015-02-18].

[7] "Lamp (software bundle)," URL: http://en.wikipedia.org/wiki/LAMP_(software_bundle) [accessed: 2015-02-21].

[8] "D3.js — data-driven documents," URL: http://d3js.org/ [accessed: 2015-02-21].

[9] "A full-featured markdown parser and compiler, written in javascript," URL: https://github.com/chjj/marked [accessed: 2015-02-21].

[10] "Mathjax — beautiful math in all browsers," URL: http://www.mathjax.org/ [accessed: 2015-02-21].

[11] VTK user's guide. Kitware, Inc., 2010, 11th ed.

[12] M. Abramowitz and I. Stegun, Mathieu Functions, 10th ed. Dover Publications, 1972, chapter 20, pp. 721–750, in Abramowitz, M. and Stegun, I., Handbook of Mathematical Functions, URL: http://www.nr.com/aands [accessed: 2015-02-22].

[13] C. M. Bender and S. A. Orszag, Advanced Mathematical Methods for Scientists and Engineers I: Asymptotic Methods and Perturbation Theory. Springer, 1999, pp. 545–551.

[14] J. W. Milnor, Dynamics in One Complex Variable, 3rd ed., ser. Annals of Mathematics Studies. Princeton University Press, 2006, vol. 160.

[15] "Wolframalpha — computational knowledge engine," URL: http://www.wolframalpha.com/ [accessed: 2015-02-26].

[16] P. Fritzson, Principles of Object-Oriented Modeling and Simulation with Modelica 2.1. Wiley-IEEE Computer Society Pr, 2003.

[17] S. Fortmann-Roe, "Insight maker: A general-purpose tool for web-based modeling & simulation," Simulation Modelling Practice and Theory, vol. 47, no. 0, 2014, pp. 28 – 45, URL: http://www.sciencedirect.com/science/article/pii/S1569190X14000513 [accessed: 2015-02-21].

A Catalogue of Thresholds for Object-Oriented Software Metrics

Tarcísio G. S. Filó and Mariza A. S. Bigonha

Department of Computer Science (DCC)
Federal University of Minas Gerais (UFMG)
Belo Horizonte, Minas Gerais, Brazil
e-mail: {tfilo,mariza}@dcc.ufmg.br

Kecia A. M. Ferreira

Department of Computing (DECOM)
Federal Center for Technological Education (CEFET-MG)
Belo Horizonte, Minas Gerais, Brazil
e-mail: kecia@decom.cefetmg.br

Abstract—**Thresholds for the majority of software metrics are still not known. This might be the reason why a measurement method that should be part of a software quality assessment process is not yet present in object-oriented software industry. In this work, we applied an empirical method to 111 system dataset, identifying thresholds for 17 object-oriented software metrics. Furthermore, we propose some improvements in this employed method. Differently from previous work, we have developed a catalogue of thresholds that gathers a greater amount of object-oriented software metrics, allowing the assessment of methods, classes and packages. Our approach suggests three ranges in the thresholds: Good/Common, Regular/Casual and Bad/Uncommon. Although they do not necessarily express the best practices in Software Engineering, they reflect a quality standard followed by most of the evaluated software. To evaluate our catalogue, we present a case study which shows its application in the evaluation of a proprietary software, in contrast with the developers consensus about its internal quality. Results show that our thresholds are capable of indicating the real panorama of the evaluated software.**

Keywords–Software Engineering; Object-oriented programming; Quality analysis and evaluation; Metrics/Measurement.

I. Introduction

Measurement is considered a fundamental part of any engineering discipline, and Software Engineering disciplines are not exceptions. In this context, software metrics refer to measurements that can be applied to check the indicators of processes, projects and software products. Evaluating software quality through measurements allows to define quantitatively the success or failure of a particular attribute, identifying the need of improvement. Managing software quality may allow to achieve a low number of defects and reliable standards of maintainability, reliability, and portability [1].

Despite the importance of metrics in object-oriented software quality management, they have not been effectively used in software industry [2][3]. One possible reason is the fact that for the majority of metrics, thresholds are not defined. Knowing these values is essential because they may allow the metrics to be used to the assessment of software quality. Moreover, without the knowledge of these thresholds, we cannot answer simple questions like “Which classes in the system have a large number of methods?” or “Which methods in the system have a large number of parameters?”.

In the current scenario of Software Engineering, the internal quality of software is usually evaluated by means of qualitative inspections, which takes time and generates high costs. The use of metrics in conjunction with a catalogue of thresholds empirically derived may provide an efficient approach to assess the software quality in an automated way.

Our major contribution is a catalogue of thresholds for 17 object-oriented software metrics, which covers a larger amount of metrics, providing the assessment of methods, classes and packages. Even though previous researches have proposed different techniques to derive thresholds for software metrics, most of them cover only few metrics [3]–[6]. In such a scenario, we do not aim to propose a new method to derive thresholds. Instead, we employed the empirical method proposed by Ferreira et al. [3], which is based on the analysis of the statistical distribution of the measures found in practice. Moreover, we introduce some improvements in this method. When we compare the contributions of this paper with the results presented by Ferreira et al. [3] and other previous studies, we can spot three majors differences. (1) We provide thresholds for a large number of software metrics. (2) The proposed thresholds aim to provide a benchmark for the quantitative evaluation of the internal quality of software systems, considering not only classes, but also methods and packages. (3) Differently from previous work, we evaluate our catalogue of thresholds in a proprietary software, of considerable size, supported by the qualitative definitions about the aspects of its internal quality reported by the developers themselves, extending the thresholds evaluation to outside of the open-source universe.

The remaining of this paper is organized as follows: Section II presents the data collection, a set of systems, as well as the preparation of this data to be used in the proposed thresholds. Section III describes the employed method to extract thresholds, followed by illustrative examples presented in Section IV. Section V presents the results of this research, showing a catalogue of the identified thresholds. Section 6 relates a case study conduct in a proprietary software to verify the effectiveness of applying our catalogue in software quality management. In Section VII, we discuss how our work is related with existing efforts in the literature. Section VIII discusses threats to validity. Section IX presents possible future directions of this research and makes final remarks.

II. Data Collection

This section describes Qualitas.*class* Corpus [7], which is the set of systems used in this research, as well as the data preparation and the generation of statistical data on metrics necessary to the development of this work. Qualitas.*class* Corpus [7] provides compiled Eclipse Java projects for the 111 systems included in Qualitas Corpus, provided by Tempero et al. [8]. Qualitas.*class* Corpus relied on Metrics 1.3.8 [9], which contains implementation details about the metrics, to compute their values, providing XML files with values of 17 metrics:

Basic Metrics: no. of classes (NOC), no. of methods (NOM), no. of fields (NOF), no. of overriden methods (NORM), no. of parameters (PAR), no. of static methods (NSM) and no. of static fields (NSF).

Complexity metrics: method lines of code (MLOC), specialization index (SIX), *McCabe* cyclomatic complexity (VG) and nested block depth (NBD).

CK metrics: weighted methods per class (WMC), depth of inheritance tree (DIT), no. of children (NSC) and lack of cohesion in methods (LCOM).

Coupling metrics: afferent/efferent coupling (CA/CE).

To read the available XML files at Qualitas.*class* Corpus, it was developed a tool that generates text files containing all the measurements for each metric. We relied on R [10], a tool for statistical computing, to generate the cumulative relative frequency graph, which gives the summary of the frequency below a given level, as well as on the histogram in logarithmic scale, which plots the histogram in double logarithm scale. We also used this tool to generate an statistical dataset on object-oriented software metrics [11], intending to help researchers in their work on software metrics.

III. Method to Identify Thresholds

An important problem in statistics is how to obtain information about the form of the population from which a sample is drawn. For this purpose, it is used EasyFit [12] to perform the selection of the appropriate distribution that has a best fit for a dataset. Besides that, EasyFit plots the *pdf* (probability density function) graph which describes the probability of a variable assuming a value x: $f(x) = p(X = x)$. The purpose of this step is to set the appropriate distribution for each software metric studied. Exploring the distributions of software metric values is crucial to improve the understanding of the internal structures of software [13]. From the distribution, it is possible to understand its characteristics, for example, if its average value is representative for the analysis or if the distribution is heavy-tailed or skewed-right [3][14].

Given the graphical views and the knowledge of the characteristics of the probability distributions which are best fitted to the measures, it is possible to derive thresholds for the metrics. In the approach of Ferreira et al. [3], when the metric has a distribution with a representative average value, like the *Poisson* distribution, this value is taken as typical for this metric, otherwise, the authors worked with three ranges for the metric values: *Good*, *Regular* and *Bad*. The *good* range corresponds to values with high frequency. The authors argue this is the most common values of the metric in practice, and nevertheless these values do not necessarily express the best practices in Software Engineering, they expose the pattern of most software systems. The *bad* range corresponds to values with quite low frequency, and the *regular* range corresponds to values that are not too frequent neither have very low frequency. The visual analysis of the graphical views allows establishing the thresholds. It is important to notice that Ferreira et al. applied this method to all set of software systems, and also group them by application domain, size and type, but they did not find relevant differences in the suggest thresholds among these approaches. So, in accordance with these results, we applied the method to the entire set of systems, expecting that the suggested thresholds are useful for all systems, regardless of the of application domain, size and type.

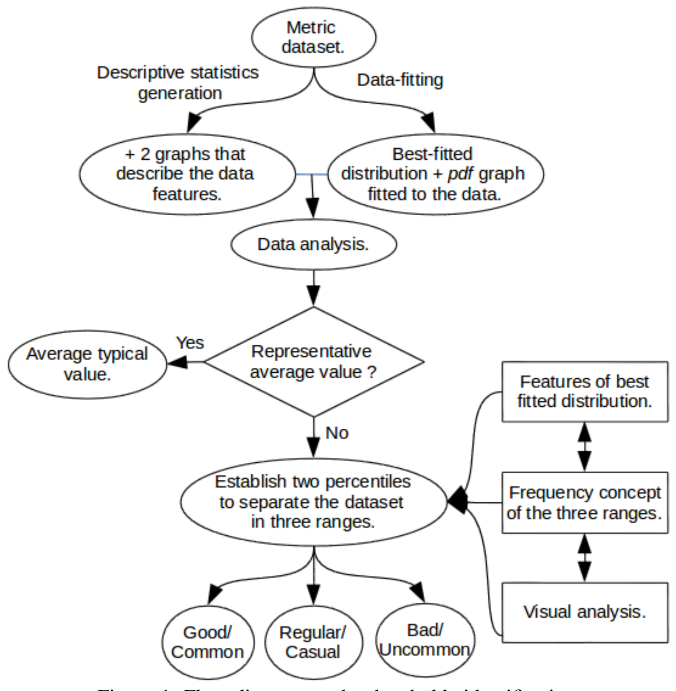

Metric dataset.
Descriptive statistics generation
Data-fitting
+ 2 graphs that describe the data features.
Best-fitted distribution + pdf graph fitted to the data.
Data analysis.
Average typical value.
Yes
Representative average value ?
No
Features of best fitted distribution.
Frequency concept of the three ranges.
Visual analysis.
Establish two percentiles to separate the dataset in three ranges.
Good/ Common
Regular/ Casual
Bad/ Uncommon

Figure 1. Flow diagram to the thresholds identification.

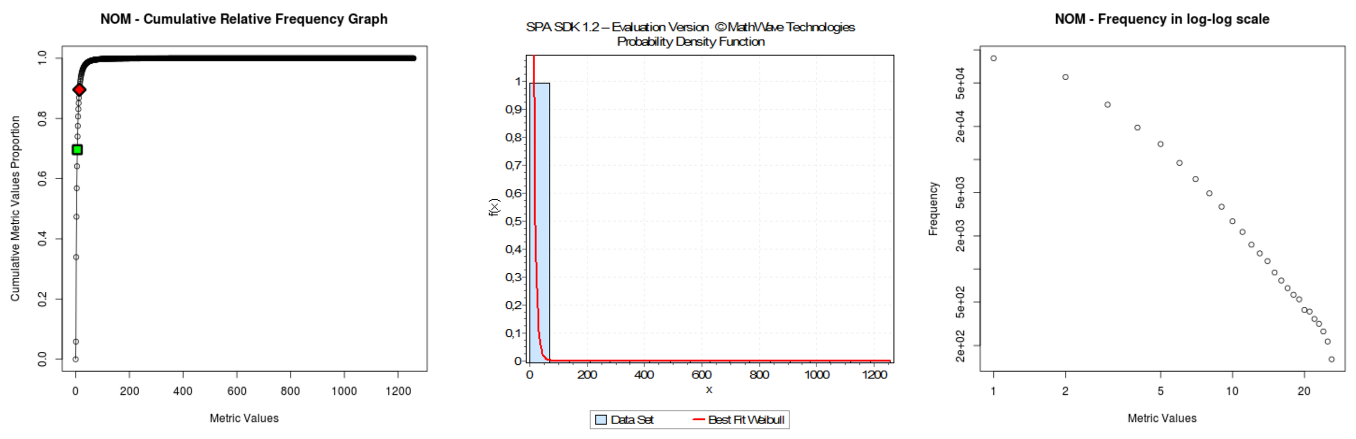
This paper also presents the proposed improvements to the original method of Ferreira et al. [3]. First, we modify the ranges names to: *Good/Common*, *Regular/Casual* and *Bad/Uncommon*, which we believe will express better the importance of frequency concept in the suggested thresholds. Secondly, we established, rather than the values directly, two percentiles, based on a visual analysis of the graphical views and on the frequency concept in the thresholds. These percentiles are capable to separate the dataset in the three ranges of values mentioned. Although the visual analysis is not dispensed, the use of predefined percentiles brings a relevant improvement to the method, it allows to obtain the values directly from the dataset, making the application of the method more reproducible. Figure 1 summarizes the method to identify thresholds in a flow diagram.

IV. Illustrative Examples

This section describes the data analysis of *Number of Methods (NOM)* and *Depth of Inheritance Tree (DIT)*.

A. *Number of Methods*

The Cumulative Relative Frequency Graph showed in Figure 2a suggests a heavy-tail distribution, because the approximation of 100% of cumulative relative frequency along the x axis (metric values) occurs in a drastically faster way, i.e., there is a nearly instantaneous approximation of 100% of the measures. This means that the systems under analysis possess several classes with few methods and a small number of classes with many methods. Figure 2b shows that the dataset of NOM is best fitted to *Weibull* distribution, with parameters



(a) (b) (c)

Figure 2. NOM: (a) Cumulative Relative Frequency Graph (b) *pdf* fitted to the Weibull (c) Histogram *log-log* scale.

$\alpha = 0,852$ and $\beta = 5,879$. As the shape parameter α is less than one, Weibull is a heavy-tailed distribution. If this is the case, the sample mean and variance cannot be used as estimators of the population because the central limit theorem does not apply, which would mean that basing any conclusions on sample means without fully understanding the distribution would be questionable at best [13]. So, the mean value is not representative. Figure 2c exhibits the dataset in *log-log* scale. In this graph, it is noticed a straight leaning to the left, a power law feature [3][13]. This pattern enhances the features already mentioned, the majority of classes has few methods and the mean is not representative. As this metric does not have a value which may be taken as typical, we identified the values representing the 70° and 90° percentiles of the dataset, which correspond to the values 6 and 14. The 70° and 90° percentiles were chosen by a visual analysis of the Cumulative Relative Frequency Graph showed in Figure 2a. The points marked with green color/square shape and red color/diamond shape represent the regions identified in this analysis. Furthermore, the choice of these percentiles is also based on the concepts of *Good/Common*, *Regular/Casual* and *Bad/Uncommon* ranges. Thus: (1) based on visual analysis, (2) the established concept for the ranges, and (3) inspired by the work of Alves et al. [5] — who used percentiles to statistically part quality metric profiles in thresholds identification — we tried to apply the 70° and 90° percentiles in most of the metrics in order to identify the measures able to separate the three suggested ranges. We found up there are variations that go in accordance with the distribution curve features, when they are taller or flattened, or depending on a higher or lower metric value (x axis) to reach a greater cumulative frequency, in which the 70° and 90° percentiles do not have significance. In such cases, we relied mainly on the visual analysis and distribution features to identify the regions that are able to separate the three suggested ranges. The values 6 and 14 allow us to separate number of methods metric in three ranges: *Good/Common* ($NOM \leq 6$), *Regular/Casual* ($6 < NOM \leq 14$) and *Bad/Uncommon* ($NOM > 14$).

B. Depth of Inheritance Tree

The data of DIT do not suggest a heavy-tailed distribution, but a right-skewed one, as showed by the Cumulative Relative Frequency Graph in Figure 3a. According to Figure 3b, the dataset is best fitted to the *Gumbel Max* distribution, with parameters $\alpha = 2.170$ and $\beta = 1,469$. By the adjustment line of the data to the distribution, it is possible to identify the right-skewed feature. In this kind of distribution, the mean value is not representative. Figure 3c shows the dataset in a *log-log* scale. In this graph, it is not noticed a straight leaning to the left, which does not suggest a power law. We have identified the values that represent the 70° and 90° percentiles of the dataset, which correspond to the values 2 and 4. The 70° and 90° percentiles were identified by a visual analysis of the Cumulative Relative Frequency Graph showed in Figure 3a. So, the identified threshold for this metric is: *Good/Common* ($DIT \leq 2$), *Regular/Casual* ($2 < DIT \leq 4$) and *Bad/Uncommon* ($DIT > 4$).

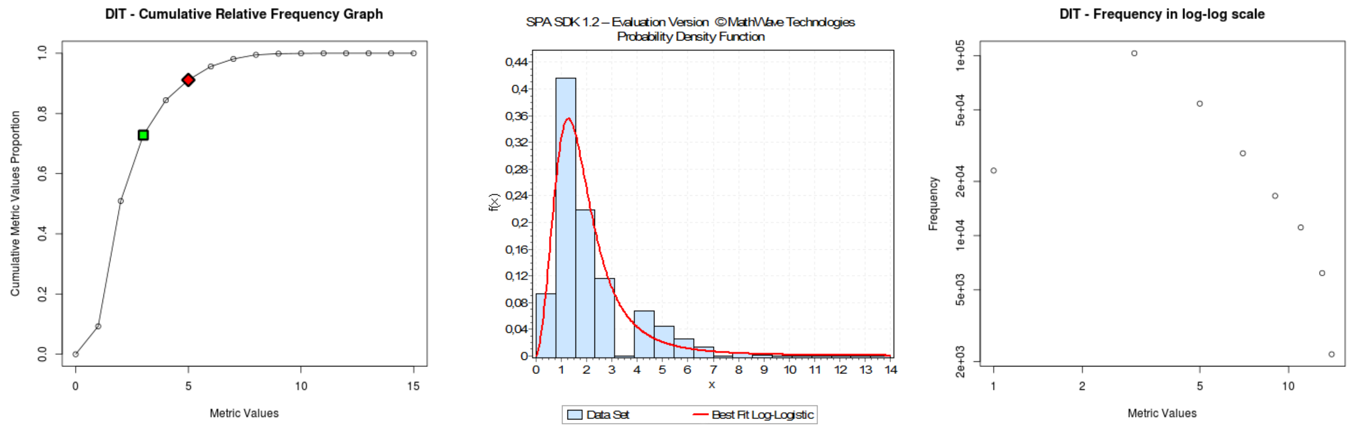
V. Results

Table I presents the identified thresholds for each metric analysed. Table II shows the best fitted distributions for each metric, with their parameters. Our catalogue of thresholds reflects a pattern followed by most of the software systems in Qualitas.*class* Corpus, which may be useful in some scenarios of Software Engineering.

TABLE I. IDENTIFIED THRESHOLDS.

Metric	Good/Common	Regular/Casual	Bad/Uncommon
CA	$m \leq 7$	$7 < m \leq 39$	$m > 39$
CE	$m \leq 6$	$6 < m \leq 16$	$m > 16$
DIT	$m \leq 2$	$2 < m \leq 4$	$m > 4$
LCOM	$m \leq 0,167$	$0,167 < m \leq 0,725$	$m > 0,725$
MLOC	$m \leq 10$	$10 < m \leq 30$	$m > 30$
NBD	$m \leq 1$	$1 < m \leq 3$	$m > 3$
NOC	$m \leq 11$	$11 < m \leq 28$	$m > 28$
NOF	$m \leq 3$	$3 < m \leq 8$	$m > 8$
NOM	$m \leq 6$	$6 < m \leq 14$	$m > 14$
NORM	$m \leq 2$	$2 < m \leq 4$	$m > 4$
NSC	$m \leq 1$	$1 < m \leq 3$	$m > 3$
NSF	$m \leq 1$	$1 < m \leq 5$	$m > 5$
NSM	$m \leq 1$	$1 < m \leq 3$	$m > 3$
PAR	$m \leq 2$	$2 < m \leq 4$	$m > 4$
SIX	$m \leq 0,019$	$0,019 < m \leq 1,333$	$m > 1,333$
VG	$m \leq 2$	$2 < m \leq 4$	$m > 4$
WMC	$m \leq 11$	$11 < m \leq 34$	$m > 34$

The identification of anomalous measurements is seen as a part of the measurement process that may be part of a software



(a) (b) (c)

Figure 3. DIT: (a) Cumulative Relative Frequency Graph (b) *pdf* fitted to the Gumbel Max (c) Histogram *log-log* scale.

TABLE II. BEST FITTED DISTRIBUTIONS FOR THE METRICS.

Metric	Distribution	Parameters
CA	*Gen. Extreme Value*	$k = 0.797$, $\sigma = 4.303$, $\mu = 1.775$
CE	*Gen. Extreme Value*	$k = 0.527$, $\sigma = 2.834$, $\mu = 2.397$
DIT	*Log-Logistic*	$\alpha = 2.170$, $\beta = 1,469$
LCOM	*Beta*	$\alpha_1 = 0.043$, $\alpha_2 = 6.777$, $b = 8.290$
MLOC	*Pareto 2*	$\alpha = 1.226$, $\beta = 3.051$
NBD	*Gumbel Max*	$\sigma = 0.858$, $\mu = 0.931$
NOC	*Log-Logistic*	$\alpha = 1.452$, $\beta = 5.520$
NOF	*Beta*	$\alpha_1 = 0.059$, $\alpha_2 = 64.580$, $b = 2026.446$
NOM	*Weibull*	$\alpha = 0.852$, $\beta = 5.879$
NORM	*Power Function*	$\alpha = 0.006$, $a = 0$, $b = 235.200$
NSC	*Beta*	$\alpha_1 = 0.001$, $\alpha_2 = 9.467$, $b = 25465.529$
NSF	*Beta*	$\alpha_1 = 0.018$, $\alpha_2 = 18.284$, $b = 4130.615$
NSM	*Beta*	$\alpha_1 = 0.017$, $\alpha_2 = 27.961$, $b = 1646.287$
PAR	*Gumbel Max*	$\sigma = 0.973$, $\mu = 0.399$
SIX	*Power Function*	$\alpha = 0.031$, $a = 0$, $b = 24.578$
VG	*Chi-Squared*	$\nu = 1.000$, $\gamma = 1.000$
WMC	*Log-Logistic*	$\alpha = 1.142$, $\beta = 4.687$, $\gamma = 0.000$

quality assessment. After the measurements have been made, developers should compare them with previous ones, looking for unusually high values [1]. The identified thresholds may be useful in this identification, as they provide a way to do this comparison with the quality that is common in software development. But, an anomalous measure does not necessarily mean a problem, it suggests that there might be problems. Once the artifacts are quantitatively identified, one must inspect them to decide if the anomalous metric measures mean that the software quality is compromised [1]. Other scenario is the application of the thresholds in a filtering mechanism to reduce the dataset in a *bad smell* detection strategy [15].

VI. Evaluation

For the evaluation of our catalogue, we conducted a case study which evaluated a proprietary software from a public organization with a deteriorated internal quality, in order to verify the ability of the proposed thresholds in indicating this panorama. For privacy reasons, we call it XYZ. This study was divided into 3 parts, aiming to evaluate, respectively, the metrics of methods and classes, and the correlation of bad smells occurrence with our thresholds evaluation. At the end of this section, we present a qualitative analysis of the utility of the identified thresholds for the metrics which were not applied in the evaluation of XYZ.

XYZ is considered a successful software by its users and stakeholders. However, there is consensus on the team that led its development and now leads the maintenance and evolution that, actually, it has a bad internal quality. XYZ has 54,297 TLOC, 2,532 methods, 603 classes and 139 packages.

From its deployment in 2009, there was no execution of preventive maintenance, neither refactorings to improve its internal quality. During the last years, XYZ has been suffering constant maintenance, such as: fixing runtime errors or system requirements, adding new features or modifying the existing ones, improving the processing speed of its functionalities and adjusting the code to changes in the environment.

As XYZ is constantly changing, it is natural that their internal structures become more complex [16]. Aiming to compare the information about the aspects involving the maintenance and evolution of XYZ with the view of the programmers, we collected opinions of four members of the development team about its internal quality and the reasons why they think the software is in this situation. There were no forms or specific directions, we chose to let opinions to flow naturally in order to characterize the software quality from their qualitative view. Analysing the reports, we noticed a consensus that XYZ has a deteriorated internal quality. The factors cited as the root of this problem are: the lack of adoption of methodologies to systematize maintenance, lack of a software architect and ineffective requirements.

A. Part 1 - Evaluation of Methods

The purpose of Part 1 is to check if XYZ has relatively more methods classified as *Bad/Uncommon* and less methods classified as *Good/Common* than most existing software of the sample, Qualitas.*class* Corpus, what would be in conformance to the qualitative consensus of its low quality.

To do that, we measured the percentage of methods classified as *Good/Common*, *Regular/Casual* and *Bad/Uncommon* by the metrics of methods of 111 system of Qualitas.*class* Corpus, i.e., one of the software in the dataset, hsqldb, has 73.55% of its methods classified as *Good/Common*, 17.69% as *Regular/Casual* and 8.76% as *Bad/Uncommon* by MLOC metric. Subsequently, the systems were ordered as follows: in ascending order by the percentage of methods classified as

Bad/Uncommon and in descending order by the percentage of methods classified as *Good/Common*. Then, we got the positions of XYZ in the obtained rankings. A low position at the first ranking means that, relatively, XYZ shows a higher proportion of methods classified as *Bad/Uncommon* than most of the analysed software. A low position at the second ranking suggests that few methods of XYZ are well evaluated by the suggested thresholds than other software in the sample. As the software has poor quality, this would suggest a correct evaluation, i.e., the software is notoriously bad and its methods were evaluated as *Bad/Uncommon*. The evaluated metrics were:

Method Lines of Code (MLOC): in the first proposed ranking, XYZ was at 100° position. So, it was no worse than 12 of the 111 systems dataset. By the second one, XYZ was at 104° position, not being worse than 8 other systems.

Nested Block Depth (NBD): in the first ranking, XYZ was at 92° position. In the second one at 101° position.

Number of Parameters (PAR): in the first one, XYZ was at 107° position. In the second one, at 105° position.

***McCabe* cyclomatic complexity (VG):** in both rankings, XYZ was at 103°, not worse than just 9 other systems.

1) Conclusion: The data show that XYZ has, relatively, fewer methods evaluated as *Good/Common* than the vast majority of the systems present in the Qualitas.*class* Corpus. This is in agreement with the established qualitative scenario about the low quality of this system, i.e., the proposed thresholds for method metrics were able to reflect, quantitatively, the scenario of low quality of this system. In contrast, XYZ has, relatively, a high number of methods evaluated as *Bad/Uncommon* than other software in the dataset. This result suggests that the proposed thresholds to metrics that evaluate methods will not show quality where there are problems.

B. Part 2 - Evaluation of Classes

In this part of the case study, we evaluated a sample of low quality classes defined by the development team of XYZ with respect to the obtained classifications with the identified thresholds for metrics of classes. Table III presents the sample of classes, with fictitious names, and the obtained classifications, where $+1$ means *Good/Common*, -1 means *Bad/Uncommon* and 0 means *Regular/Casual*. Next we analyze the results of each metric.

TABLE III. SAMPLE OF LOW QUALITY CLASSES OF XYZ.

	NOF	DIT	WMC	NSC	NORM	LCOM	NOM	SIX
LSI	+1	+1	-1	+1	+1	-1	-1	+1
NSI	+1	+1	-1	+1	+1	-1	-1	+1
RSI	+1	+1	-1	+1	+1	-1	-1	+1
NB	-1	0	-1	+1	-1	-1	-1	-1
NTB	-1	+1	-1	+1	+1	-1	-1	+1
RB	-1	+1	-1	+1	+1	-1	-1	+1
ND	+1	0	-1	+1	+1	+1	-1	+1
LD	+1	0	-1	+1	+1	+1	-1	+1

Number of Fields (NOF): the NOF column of Table III shows that 5 classes were well evaluated and 3 classes were poorly evaluated by the threshold of NOF. As these classes are defined qualitatively as problematic, they were inspected in order to identify their features across to the received quantitative evaluation. LSI, NSI and RSI are *service* classes (fictitious class names). According to Fowler [17], an anemic domain model occurs when the business logic is not put in the domain objects. Instead, there are a number of *service* objects that capture this logic. When pulling behaviors into *services*, they become *Transaction Scripts*, which organize the business logic by procedures that treat requests from the view layer. These *services* are at the top of the domain model and use the model as a data repository. Then, the domain objects become "bags of getters and setters", not encapsulating the logic for the data. The *services* are a grouping of procedural functions related to domain data. Thus, anemic objects do not have behaviors and *services* are grouping of procedural functions. Therefore, these classes were classified as *Good/Common* by the identified threshold for NOF, not because they have a reasonable amount of fields, but for not having fields at all. Therefore, by using only NOF, these classes would be well evaluated, despite being problematic within the project, providing potential errors in the quantitative evaluation. NDAO and LDAO are objects of type DAO (*Data Access Object*), which encapsulate all data access logic within an application. These are classes that, by their goal within the system architecture, have few attributes, without characterizing an architectural violation or a bad programming practice, as occurs with the anemic domain model. These classes are problematic because of their size and complexity, and not by their number of fields. Thus, the classes were correctly well evaluated in relation to the NOF threshold. NB, NTB and RB are objects of type *managed bean*, which were poorly evaluated by the identified threshold for NOF. This type of class is typical in *JavaServer Faces* applications, where each *managed bean* should be associated to one or more components of a web-page [18]. These classes are large and complex because they are not being componentized in *managed beans* more cohesive, which meet a specific purpose within the web-page. So, the qualitative assessment of these classes match the result of applying the identified threshold for NOF, both suggesting that the classes show poor quality.

Depth of Inheritance in Tree (DIT): the DIT column of Table III shows that three classes were evaluated as *Regular/Casual* and the rest of them were evaluated as *Good/Common*. Inheritance is a resource rarely used in this system, mainly because of the anemic domain model. Thus, the vast majority of classes do not use this resource and, consequently, do not have a deep inheritance tree. Therefore, they were evaluated as *Good/Common* by the DIT threshold. In fact, there are no problems in these classes related to the depth of inheritance tree. So, these cases are not considered incorrect evaluations, on the contrary, the identified threshold of DIT correctly evaluated these classes as *Good/Common*. It is necessary to understand correctly what is being evaluated with the metric to perform a correct interpretation of the obtained results. We must remember that a positive rating by the DIT threshold in most classes of the system does not mean that the design is making an appropriate use of inheritance, because inheritance can not even being used. The classes that were evaluated as *Regular/Casual*, ND and LD have an inheritance hierarchy imposed by the used persistence framework. In a qualitative evaluation, it was concluded that the DIT threshold evaluated these classes correctly, as they are not impossible to be understood neither are of easy understanding.

Weighted Methods per Class (WMC): by WMC column of Table III, we observe that all classes were evaluated as

Bad/Uncommon by the identified threshold. WMC is a measure of the class complexity and taking into account the characteristics of XYZ and its evolutionary process, our threshold is capable of showing the natural increase of complexity related by Lehman [16] in a quantitative way. These classes are really complex and they assumed many responsibilities throughout the software evolution, becoming hard to understand and maintain.

Number of Children (NSC): the NSC column of Table III shows that all classes were evaluated as *Good/Common* by the NSC threshold. Indeed, these are classes that do not have children, not showing problems related to the evaluated aspects of this metric.

Number of Overriden Methods (NORM): the NORM column of Table III shows that all classes were well evaluated by the identified threshold, except for NB class. Indeed, by the low utilization of inheritance resource shown by the results of DIT and the qualitative analysis of the software, this result was expected, after all, the classes have no problems related to excessive overwriting of methods and, therefore, were correctly evaluated well. For the NB class, which is poorly evaluated, we can see that this class was one of the three classes that were classified as *Regular/Casual* by the DIT threshold, indicating a depth of inheritance tree outside the ideal and the next to be considered inappropriate. NB overwrites 7 methods, an amount considered high by the threshold. The major problem of this quantity of overwritten methods is that the class becomes difficult to understand. This problem is aggravated when combined with an improper inheritance hierarchy, because the class may override many methods due the fact that the parent class is not appropriate. So, the negative evaluation is consistent with the qualitative evaluation, as well as the positive evaluations were considered correct.

Lack of Cohesion in Methods (LCOM): the LCOM column of Table III shows that 6 of the 8 classes have low cohesion. Classes of *service* type do not have cohesion between their methods since they have no fields. Therefore, the inspected *services* classes have a consistent poorly evaluation by the LCOM threshold. As the *managed beans* are poorly componentized, they have low cohesion. If they were better componentized, cohesion would increase naturally, after all, the methods would have higher similarity. On the other hand, DAO objects have cohesion by the fact that their methods use fields inherited from the parent classes related to the *framework*, performing correlated operations in the database. Therefore, the positive evaluations were correct.

Number of Methods (NOM): the NOM column shows that all classes were poorly evaluated by NOM threshold, which is in accordance with the assessments made by WMC threshold. According to Lehman [16], the functionalities offered by a system must be continuously incremented in order to maintain user satisfaction. If this growth is done in an uncontrolled way, classes will grow more and more by adding methods and fields that meet the growing expectations of the user. Failure to do this growing in a designed way will deteriorate the software quality, since the classes become large and complex, as in the case of the classes analysed.

Specialization Index (SIX): the SIX column of Table III shows that 7 classes were well evaluated and the remaining one was poorly evaluated by the SIX threshold. This metric aims to assess how much a particular class overrides the behavior of its superclasses. As expected, due the relation of SIX with NORM and DIT, NB was poorly evaluated by SIX, as was poorly evaluated by NORM and not well evaluated by DIT. The other classes do not exceed normal levels of specialization index suggested by SIX threshold and, in fact, they do no overwrite the behavior of their superclasses in an excessive way.

1) Conclusion: For this part of the study case, it was established with the programmers of XYZ a set of 8 poor-quality classes, with the aim of study the evaluations obtained by applying the proposed thresholds. All classes had at least 3 classifications out of range *Good/Common*. This suggests that if our catalogue of thresholds were applied in the management of internal quality of software systems, all the classes of the sample would be defined as objects of inspection in a measurement process, for presenting unusually high values. However, we also concluded that a single metric of our catalogue should not be used to define the quality of a class. For example, if only NOF was used to evaluate the 8 classes, only two would be considered as poor-quality. This conclusion is consistent with the work of Rosenberg et al. [4], which suggests that a single metric should not be used to evaluate a class. So, the results suggest that our catalogue is an efficient way to evaluate the classes effectively.

C. Part 3 - Bad Smells Correlation

We applied JDeodorant [19], a plugin for Eclipse that identifies bad smells, in order to identify *long methods* and *god classes* in XYZ. After that, we evaluated all the methods and classes of this system with some metrics of our catalogue related to these bad smells concepts. With these data available, we crossed the information of two qualitative variables: the presence or absence of the bad smell and the classification or no classification as *Good/Common* by the threshold. So, we obtained the number of methods or classes that have or not the bad smell against the amount of classified and unclassified methods or classes as *Good/Common* by the thresholds. This type of information is called contingency table of two qualitative variables. Furthermore, we raised the following null hypothesis about the bad smell occurrence and the evaluation obtained by the method or class with m threshold, H_m^{null}: the bad smell occurrence is independent of the method or class not be classified by the metric m in *Good/Common* range.

To evaluate this hypothesis, we used a statistical test that evaluates the dependency between the qualitative variables, called Chi-Square Test for Independence, which determines whether there is a significant association between the two established variables. The input of this test is the 2×2 contingency table, and the output is the *p*-value, which is the probability of obtaining an statistic test equal or more extreme than the one observed in the sample about. If the *p*-value is greater than 5%, we do not reject the null hypothesis. Otherwise, we can reject it, which would suggests that there is a relationship between the presence of the bad smells and the method or class not be classified as *Good/Common* by the threshold. The test was applied using the R tool [20] for the WMC, NOM, NOF and LCOM thresholds with the identified *god classes* and the evaluation obtained by the MLOC, NBD and VG with the identified *long methods*. For all of these tests, we obtained p-values less than 1%, rejecting the null hypothesis at 99% of confidence level. Therefore, it was possible to check that there is a statistical dependency between: (1) the class be evaluated as *Good/Common* by the thresholds of WMC, NOM, LCOM and NOF and do not have the bad smell *god class* and (2) the method be evaluated as *Good/Common* by the thresholds of MLOC, NBD and VG and do not have the bad smell *long method*. These situations testify in favour of the correctness of the evaluations performed by our thresholds.

D. Other Metrics

Number of Classes (NOC): a package is a collection of related types providing access protection and name space management [21]. As much as classes are added to a given package, group of classes tend to be less interrelated, suggesting a possible re-division into smaller packages to create new groups which reflect a better defined domain. The NOC threshold may be used to identify packages with a large number of classes, in order to evaluate possible adjustments.

Afferent Coupling (CA): CA measures the number of external classes to a specific package that depend on their inner classes [22]. The higher the CA value, the greater the responsibility of that package and the higher its relevance within the software. A package with many external dependencies becomes a risk artifact, knowing that a change on it may impact directly and indirectly in many classes. In this sense, the identified thresholds are useful in order to say what is a high CA value, based on the standards of software quality that has been developed. Therefore, knowing what is a high value of CA may aid to identify the need for re-distribution of responsibilities of a too influential package in a set of packages more cohesive.

Efferent Coupling (CE): CE is the number of internal classes in a package that depend on external classes of this package [22]. A high value of CE means that the package strongly depends on other classes of other packages, making it a more unstable artifact, given the high degree of dependent classes. Keeping a low degree of CE means getting a package with greater independence. The identified thresholds show that most packages in OO software have been developed with up to 6 dependent classes, occasionally they have 7 to 16 dependent classes and rarely more than 16.

Number of Static Methods (NSM): a static method belongs to the class, rather than to an instance of the class [21]. Despite this mechanism breaks the object-oriented programming concept, it has practical utility in the context of object-oriented software development in the Java Platform [23]. However, they make the software less flexible because they cannot be overridden. Therefore, the NSM threshold allows to identify high values of static methods.

Number of Static Fields (NSF): a static field creates an attribute that belongs to the class rather than being associated with an instance of that class [21]. All class instances share the static field, which is in a fixed location in memory. Any change in an static field value will reflect in all instances of the class. Static fields are extremely useful in the object-oriented software developed in Java platform [23]. An example is the implementation of the design pattern *Singleton*, which guarantees the existence of only one instance of a particular class, providing global access to that object. However, classes that excessively use this feature and also static methods have acquired a bad reputation because it prevents developers to think in terms of objects [23]. The identified thresholds indicate what is a high value for NSF, allowing the system developer to identify classes in the design that are using this feature excessively.

VII. Related Work

In this section, we discuss related work that has been done in order to identify thresholds for object-oriented software related metrics. Rosenberg et al. [4] identified thresholds of metrics with the goal of applying these metrics to assess the reliability of software systems at NASA. The research was conducted in more than 20,000 classes distributed over 15 projects. The goal was to identify thresholds capable of discriminating weak code from solid code by statistical studies. This study presents thresholds for six software metrics at class level. As the analysis was done in the context of software development at NASA, the results not necessarily can be applied to other application domains. Moreover, as the dataset is not open and the method is not described in details, the results of the research are not reproducible.

Shatnawi et al. [24] presented a study of the relationship between object-oriented metrics and error-severity categories, identifying thresholds values that separate no-error classes from classes that had high-impact errors. This study presents thresholds for five metrics at class level. Moreover, the method was applied in a limited size and domain sample, being a threat to use these thresholds for software in general.

Alves et al. [5] designed a method that determines metric thresholds empirically from a statistical analysis of a benchmark of software systems, which are derived by choosing the 70%, 80% and 90% percentiles from these data. The authors focus on the method description, presenting thresholds for 3 metrics at method level and 2 metrics at class level. Besides that, fixed percentiles not necessarily work for all metrics. Due to this limitation, in the present work we carried out the data analysis by understanding the distribution curve of the values to establish these percentiles.

Oliveira et al. [6] proposed the concept of relative thresholds for evaluating metrics data that follow a heavy-tailed distribution. The thresholds are called relative because they assume that metric thresholds should be followed by most sources code entities, but that is also natural to have a number of entities in the “long-tail” that do not follow the defined limits. So, absolute thresholds should be complemented by the percentage of entities that the upper limit should be applied to. This work has focused on the method description, deriving thresholds for seven metrics at class level.

Our work presents 17 object-oriented software metrics derived by the same method, that is a large amount of thresholds compared with previous studies. Besides that, our catalogue does not cover only metrics at class level, but also metrics at method and package level. The used approach is easily reproducible and does not bring much statistical complexity. Moreover, the proposed thresholds are based on analysis of a large amount of software, of various sizes and domains, making the results more reliable in terms of representativeness of software generally.

VIII. Threats to Validity

Software metric tools can measure different values for the same metric. The identified thresholds may classify artifacts as worse than they in fact are. So, like Qualitas.*class* Corpus [7] relied on Metrics Plugin for Eclipse to collect the measures, we cannot assure that the identified thresholds will be applicable when using tools that collect metrics with an implementation different from Metrics. The sample of applications used in this research may also be a threat to the validity of this study, because of its size and representativeness for software in general. However, Qualitas.*class* Corpus has at least equal software samples than other analysed studies and is composed of various types and domains of software systems. Another threat to validity is that this approach assumes that the metrics are unidirectional in the sense of having a clear good and bad orientation. So, if all classes show, for example, $DIT = 0$, they would be classified as *Good/Common* by the suggested thresholds, although the design is suboptimal, it does not use inheritance. However, none of our suggested thresholds contemplate the evaluation of the global quality of the software systems and $DIT = 0$ is the most frequent value found in practice. This triggers an alarm about the misuse of our catalogue and misinterpretation of its results, being fundamental to evaluate the scenario in which it will be applied.

IX. Conclusion and Future Work

The knowledge of the thresholds is of fundamental importance in the promotion of the effective use of software metrics regarding the management of internal quality of software systems. In this research, we employed the method proposed by Ferreira et al. [3] to propose a catalogue of 17 thresholds for object-oriented software metrics, covering the quantitative evaluation of methods, classes and packages. The method is based on the analysis of the statistical distributions of measures found in practice. It was observed that the metrics fit a heavy-tailed or a skewed-right distribution. So, three ranges of values were taken as the metric thresholds. The range names were modified to *Good/Common*, *Regular/Casual* and *Bad/Uncommon*, which express better the importance of frequency concept in the thresholds. Although they do not necessarily express the best design principles established for Software Engineering, they reflect a quality standard followed by most of the software evaluated.

The evaluation of the proposed thresholds in a proprietary software showed the effectiveness of our catalogue of the thresholds in indicating the real panorama of the internal quality of software systems, that is, the evaluation do not show quality where there is not. For metrics that do not participate in the evaluation conducted in a proprietary software, we presented a qualitative analysis which describes their appliance in the identification of possible problems in object-oriented software systems. So, we presented at least one analysis for each one of our suggested thresholds. With our catalogue of thresholds, we presented a contribution in the promotion of software metrics as an effective instrument to manage the internal quality of software systems.

As future work, we intend to continue evaluating the identified thresholds through more case studies, aiming to continue the investigation if the range in which the metric falls reflects the real situation of the assessed artifact. Furthermore, we intend to conduct the development of a tool that performs a strategy composition of metrics to identify software entities to be refactored, by applying metric thresholds.

References

[1] I. Sommerville, Software Engineering, 9th ed. Harlow, England: Addison-Wesley, 2010.

[2] M. Riaz, E. Mendes, and E. D. Tempero, "A systematic review of software maintainability prediction and metrics," in ESEM, 2009, pp. 367–377.

[3] K. A. Ferreira, M. A. Bigonha, R. S. Bigonha, L. F. Mendes, and H. C. Almeida, "Identifying thresholds for object-oriented software metrics," Journal of Systems and Software, vol. 85, no. 2, Feb. 2012, pp. 244–257.

[4] L. Rosenberg, S. Ruth, and A. Gallo, "Risk-based Object Oriented Testing," in Proceedings of the 24 th annual S.E. Workshop, NASA, S.E.Lab, 1999, pp. 1–6.

[5] T. L. Alves, C. Ypma, and J. Visser, "Deriving metric thresholds from benchmark data." in ICSM. IEEE Computer Society, 2010, pp. 1–10.

[6] P. Oliveira, M. T. Valente, and F. P. Lima, "Extracting relative thresholds for source code metrics," in CSMR-WCRE, Software Evolution Week - IEEE Conference on, Feb 2014, pp. 254–263.

[7] R. Terra, L. F. Miranda, M. T. Valente, and R. S. Bigonha, "Qualitas.class Corpus: A compiled version of the Qualitas Corpus," Sof. Eng. Notes, vol. 38, no. 5, 2013, pp. 1–4.

[8] E. Tempero, C. Anslow, J. Dietrich, T. Han, J. Li, M. Lumpe, H. Melton, and J. Noble, "Qualitas corpus: A curated collection of java code for empirical studies," in APSEC2010, Dec. 2010, pp. 336–345.

[9] "Eclipse metrics plugin 1.3.8," 2014, URL: http://metrics2.sourceforge.net [accessed: 2014-12-30].

[10] R, "R project for statistical computing," 2014, URL: http://www.r-project.org/ [accessed: 2014-12-30].

[11] T. G. Filó, M. A. Bigonha, and K. A. Ferreira, "Statistical dataset on software metrics in object- oriented systems," Sof. Eng. Notes, vol. 39, no. 5, 2014, pp. 1–6.

[12] "Easyfit," 2014, URL: http://www.mathwave.com/products/easyfit.html [accessed: 2014-12-30].

[13] G. Baxter, M. Frean, J. Noble, M. Rickerby, H. Smith, M. Visser, H. Melton, and E. Tempero, "Understanding the shape of java software," in OOPSLA, New York, NY, USA, 2006, pp. 397–412.

[14] L. Doane, David & Seward, "Measuring Skewness: A Forgotten Statistic?" J. of Statistics Education, 2011, pp. 1–18.

[15] M. Lanza, S. Ducasse, and R. Marinescu, Object-Oriented Metrics in Practice: Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems. Springer, 2007.

[16] M. M. Lehman, "Programs, cities, students, limits to growth?" Programming Methodology, 1978, pp. 42–62, inaugural Lecture.

[17] M. Fowler, "Anemic domain model," 2014, URL: http://www.martinfowler.com/bliki/AnemicDomainModel.html [accessed: 2014-12-30].

[18] Oracle, "The java ee 6 tutorial," http://docs.oracle.com/javaee/6/tutorial/doc/bnaqm.html, 2014.

[19] "Jdeodorant," 2014, URL: http://www.jdeodorant.com/ [accessed: 2014-12-30].

[20] "Chi-squared Test of Independence," 2014, URL: http://www.r-tutor.com/elementary-statistics/goodness-fit/chi-squared-test-independence [accessed: 2014-12-30].

[21] Oracle, "Java se technical documentation," 2014, URL: http://docs.oracle.com/javase/ [accessed: 2014-12-30].

[22] R. Martin, "OO design quality metrics - an analysis of dependencies," in Workshop Pragmatic and Theoretical Directions in O.O. Software Metrics. OOPSLA, 1994, pp. 1–6.

[23] J. Bloch, Effective Java, 2nd Edition, The Java Series. NJ, USA: Prentice Hall PTR, 2008.

[24] R. Shatnawi, W. Li, J. Swain, and T. Newman, "Finding software metrics threshold values using ROC curves," Journal of Software Maintenance and Evolution: Research and Practice, vol. 22, no. 1, 2010, pp. 1–16.

Hybrid Mockup-driven Development: An Agile Model-driven Approach for Web Applications

Gürkan Alpaslan and Oya Kalıpsız
Department of Computer Engineering
Yıldız Technical University
İstanbul, Turkey
e-mail: gurkan89@hotmail.de, oya@ce.yildiz.edu.tr

***Abstract*— Model-driven development is an effective way thanks to the automated code generation, documentation and creating a prototype of the project. Agile principles target to improve software developing pace and customer satisfaction with iterative approach and agile principles. By considering these two approaches, their inefficiencies are avoided and their advantages are combined. In this paper, we combine these two approaches for web applications; for this purpose, we choose the *hybrid MDD* method as skeleton, which is an agile model-driven approach. We target with this approach to create a more efficient software method for web applications.**

Keywords-Agile model-driven development; Hybrid model; Mockup-driven development model

I. INTRODUCTION

Model-driven software development (MDD) is based on the approach that "everything is model" [1]. Models are the abstract representation of the system elements [2]. MDD's basic goal is to develop software on a higher abstraction level than programming languages. Models are produced before writing source codes and the software is developed using these models. This way also provides software documentation.

An agile development method [3] is an approach which has values, principles and practices. Agile methods propose the iterative software development. With every iteration, a piece of software is produced, and, by the end of the last iteration, total software emerges. In agile methods, customers are the part of the software life cycle and all stakeholders are working together.

The main goal of combining agile methods and MDD is to benefit from both approaches. The method of that is to implement MDD principles to agile methods by using agile architecture as the skeleton. For this purpose, there are different approaches in the literature [4]. The first method is the Agile Model-Driven Development (AMDD) high level life cycle [5][6][7]. This structure divides the software life cycle into two parts, namely, the initial part and development part. In initial part, the system scope is determined and initial models are produced. The development part is the section where the iterations are implemented. The more improved structure is the Hybrid model [8]. Hybrid model has also two parts, but in this approach, different teams are defined. This approach is defined for general projects, not for the customized platforms, with limitations lightweight project constraint.

This paper's goal is to customize the Hybrid model for web applications by adding new suggestions for improving productivity and software quality. For this purpose, Mockup-driven development approach [9] and Hybrid model are combined. Mockup-driven development is a model-driven development for web applications. Mockups are the structures that produce models for web applications that are used as the prototypes of web applications [10]. Mockup-driven development propose to develop web applications from mockups which are more visual than diagrams. By automated code generation, produced mockups can convert to Hyper Text Markup Language (HTML), Cascading Style Sheets (CSS) and JavaScript codes.

In the following section, we describe the agile model-driven basics. Then, we describe the mockup-driven development approach in Section 3. After that, we describe our approach, hybrid mockup-driven development, in Section 4. An ongoing case study is described in Section 5. Finally, we conclude our experience and future work are described.

II. AGILE MODEL-DRIVEN DEVELOPMENT

Agile model-driven development proposes an iterative and incremental approach that supports all model-driven benefits like documentation and automated code generation [11][12]. In this approach, all models are produced by the agile models. Agile models [13] are the structures which are developed by implementing agile principles. Some of the agile core principles [14] are:

- Assume Simplicity
- Embrace Change
- Incremental Change
- Maximize Stakeholder Investment
- Rapid Feedback
- Working Software is Your Primary Goal
- Travel Light

To develop models properly with these principles is an important part of the approach. In AMDD, the total requirement list is divided into portions and the portions are implemented consecutively.

Generally, AMDD proposes to develop the software by test-driven development method. Test-driven development

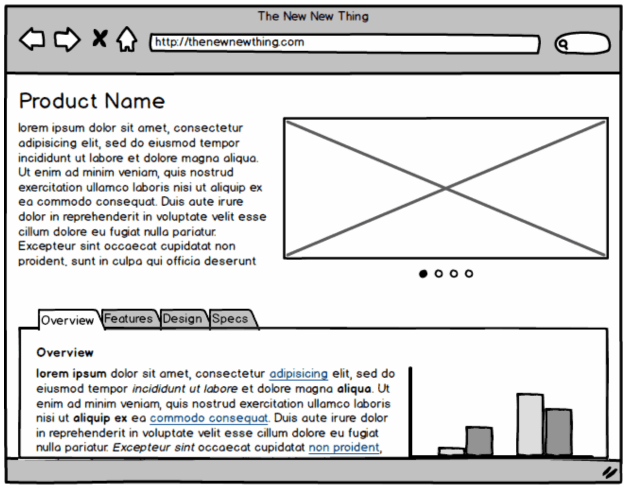
[15][16] proposes to produce the software test units before writing the source codes. After the source code is produced, the code is tested and the results are evaluated. Three results are possible: passed, failed and refactor. Passed means the code has passed the test succesfully, failed means that code cannot work or fulfill the requirements and refactor means code works but should be improved.

III. Mockup-driven Development

In web application development, HTML, CSS and JavaScript codes are used for designing both visual interface and system logic run. Mockups provide these elements by the automated code generation with supporting mockup tools. For this purpose, they are proper and simple to use in web applications. It provides to decrease error possibility and development time.

Mockups can define as the models of web applications [9]. Mockup tools provide an design interface which has items using in web systems. Developer can design the attributes of web items and make to connection with other items.

In our projects, we plan to use Axure [17] mockup tool. An example of mockup is shown in Figure 1.



lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt

lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor *incididunt ut labore* et dolore magna **aliqua**. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut **aliquip ex** ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. *Excepteur sint* occaecat cupidatat non proident,

Figure 1. An example mockup design [18]

In Figure 1, an example website is designed by mockup items. Information writings and graphics are used in example. The graphic on the right up of the page in Figure 1 can change with the clicking little four circles. Bottom design of the page in example can also change with the clicking buttons that named as overview, feature, design and spaces. This level is the model part of the process. After the model developing is completed, this model transforms to real websites.

IV. Hybrid Mockup-driven Development Approach

In our approach, we combine the hybrid method and mockup-driven development approach by adding some new parts and customize the hybrid method for web applications by using mockups. The reason of customizing for web application is that the mockup-driven development method is suitable for web applications. Mockups are more visual than diagrams and easier to understand for all stakeholders who have no knowledge about software engineering. This advantage makes it easier to analyze the requirements and working with mockups.

The approach can be used for small or medium size projects. The reason of that is the challenges of remodeling when lots of iteration occur.

A. *The Steps of the Approach*

The hybrid mockup-driven development is an iterative approach where three interconnected teams work in parallel. Figure 2 shows the processes of the approach. In the figure, the numbers near the processes represent the teams that work on the process. These teams are the business analyst team, the model-driven development team and the agile development team. The business analyst team is responsible for communication with customers and prepare the requirements. The MDD team is responsible for preparing the mockup environment and the generated codes, which are produced by mockup tools with automated code generation. The agile development team is responsible for hand-crafted codes and integrating them with generated codes. The team is also responsible for preparing the test units and testing the iteration software.

In the agile method, all the teams work together with high interaction. In the approach, these teams work together with support each other in every step. Teams use a common sharing area and all team member has information what the other teams have done. Also, before the iteration begin, all teams meet together and discuss that what and how they will do next iteration.

The steps in the approach are listed below.

- The business analyst team gets the requirements from customers.
- The business analyst team and MDD team produce the mockups. In this part, using pair development and a customer representative are working together for mockups.
- The agile team prepare the test environment and test code units for mockups.
- The MDD team prepare the mockup tools and mdd environment.
- Produced mockups transform to HTML, CSS, JavaScript codes with automated code generation by MDD team.
- The automated produced codes are shared with the agile team and by the agile team, this codes are completed with hand-crafted codes. Final code parts are implemented to test units.
- Finally, developers have a software part and they present it to customers.
- They get the feedbacks, document the reviews and pass to the next iteration.
- With every iteration, this period repeats until the total software has been produced.

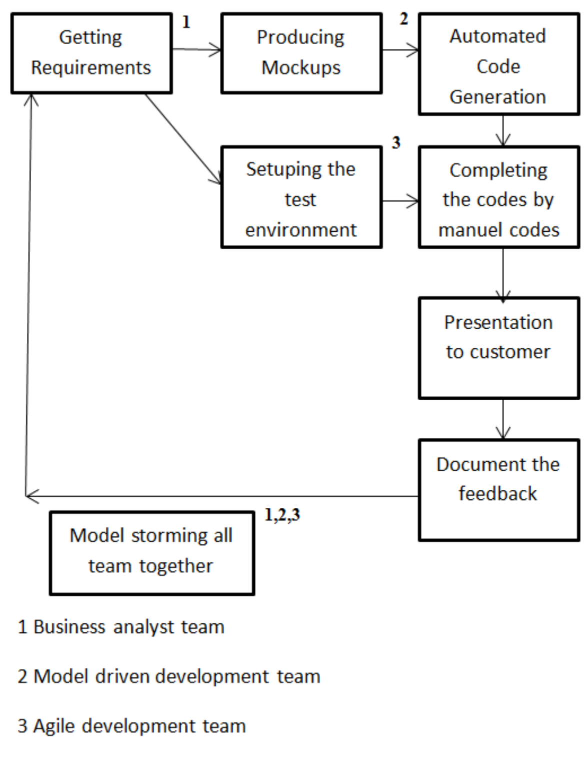

Getting Requirements
1
Producing Mockups
2
Automated Code Generation
2
Setuping the test environment
3
Completing the codes by manuel codes
3
Presentation to customer
1
Document the feedback
1
Model storming all team together
1,2,3
1 Business analyst team
2 Model driven development team
3 Agile development team

Figure 2. Hybrid mockup-driven development

The model storming part is a transition part for all teams to be prepared for next iteration. In this part, all teams meet together and firstly evaluate the completed iterations reviews. After that, they prepare the next iteration activity plan.

System test is done with the method of test-driven development approach, which described in Section 2. Test plans designed by agile development team. Black box test can be used for this type of small size web projects and the systems which has supported with automated code generation. A benefit of MDD is decreasing the error possibility. Test cycle is a sequential activity. When every iteration is done, the iteration's test is implemented. Test results are documentated like the method described in Section 2 and proper updates are implemented.

B. *Evaluation of the Approach*

Hybrid mockup-driven development propose an iterative approach with paralel working different teams. So, it divides the workload into small portions. With paralel working teams, waiting time of the other teams is decreased. Also, it provides to intermediate changing on any problem or fixing part while the development process.

The basis of the approach is the agile principles which are described in Section 2. All parts of process run with supporting agile princibles, so it takes the advantage of agile architecture. Mockups are used, so more visual development is provided.

Generally, the basic benefits of the approach are software development pace, flexibility, error detection easiness, and simplicity.

C. *The comparison of the processes*

The approach is compared to hybrid model and AMDD high level life cycle. The comparison is done on these criteria: main objective, main contribution, and usage areas. In main objective part, the basis attributes of processes are described. In main contribution part, the basis contribution of software industry is described. In usage area part, customized platform of processes are described. Table 1 shows the basic differences.

TABLE I. THE COMPARISON OF THE PROCESSES

Process name	Main objective	Main contribution	Usage areas
AMDD High level life cycle	An iterative and test-driven approach with MDD princibles usage	First method that combining agile and MDD princibles	General projects
Hybrid MDD method	Development on models with difference teams iterative and incremental	The difference teams and work sharing describing on AMDD life cycle	General projects
Hybrid mockup-driven method	Development on mockups with difference teams iterative and incremental	Mockup usage with Hybrid approach	Web app.

AMDD high level life cycle is the first approach that shows that is possible to combine agile methods with model-driven methods. A hybrid process improves that model with adding different teams approach and new life cycle diagram. In our approach, we target to improve the hybrid method by using mockup and to demonstrate that mockup-driven development approach can use with hybrid method.

V. CASE STUDY

We gave this approach to a few student project teams in 'Software Engineering' class of our university and requested them to develop their projects with this approach. Also, we requested another teams to develop their project with the traditional method. So, we target to compare our approach and the traditional method.

This is an ongoing project; currently, the student project teams are in the development phase. We selected two teams developing the different projects for better analyzing the approach. One of the teams includes five students developing an online "Cinema Ticket System". Another team includes four students developing an online "Language Course Registration System". We described the approach to the teams and decided to use Axure mockup tool [17]. They firstly splitted the three specialized team as MDD team, agile development team and business analyst team. After that, we requested them to split their project into iteration modules. Currently, they are working on first iteration; we plan to complete the entire project in three months.

VI. Conclusion and Future Work

Developing software with mockups by producing prototype is an efficient way thanks to their visual interface. By this visual interface, customer requirements are realized more properly and the time for analyst part can be reduced. Paralel working teams can provide faster development and more communication. Thanks to the automated code generation, reducing the development time and amount of software errors are targeted.

However, integrating automated generated codes and hand-crafted codes can cause some problems in large scale projects. In the forward iterations, rebuilding of previous iteration model is an challenge for the projects have lots of iterations. For this purpose, limited the approach for small or medium size projects can be more proper.

In future work, we get the analyst results of the case study work described in Section 5. We report the development challenges of the approach by the feedback of teams. We identified the evaluation criteria. They are flexibility, rapidity, learning, cost effectiveness, metadata management, tool support, problem domain analyst, simplicity and maintainence. Also, we compare the approach with the traditional method and other agile model-driven methods by the analyst results.

References

[1] S. Vale and S. Hammoudi, "Context-aware model driven development by parameterized transformation", Proceedings of MDISIS, pp. 167-180, 2008.

[2] T. Stahl and M. Volter, Model-Driven Software Development, John Wiley & Sons, 2006.

[3] T. Dyba and T. Dingsory, "What Do We Know about Agile Software Development?", IEEE Software, pp. 6-9, 2009.

[4] R. Matinnejad, "Agile Model Driven Development: An Intelligent Compromise", Software Engineering Research, Management and Applications (SERA), 2011 9th International Conference on, pp. 197-202.

[5] S.W. Ambler, "Agile Model Driven Development" , XOOTIC Magazine, 2007.

[6] Agile Model Driven Development, http://agilemodeling.com/essays/amdd.htm, retrieved: 03.2015.

[7] S.W. Ambler, The Object Primer 3rd Edition: Agile Model Driven Development with UML 2.0 New York: Cambridge University Press, 2004.

[8] G. Guta, W. Schreiner, and D. Draheim "A Lightweight MDSD Process Applied in Small Projects", In Proceedings of the 35th Euromicro Conference on Software Engineering and Advanced Applications, pp. 255-258, 2009.

[9] E. Benson, "Mockup Driven Web Development", 22nd Intenational World Wide Web Conference, pp. 337-342, 2013.

[10] F. Basso, M. Pillat, F.R. Frantz, and R.Z. Frantz, "Study on Combining Model-Driven Engineering and Scrum to Produce Web Information Systems", 16th International Conference Enterprise Information Systems, pp. 137-144, 2014.

[11] G. Alpaslan and O. Kalıpsız, "Model driven development with agile process", Elektrik - Elektronik, Bilgisayar ve Biyomedikal Mühendisliği Sempozyumu , 2014.

[12] G. Alpaslan and O. Kalıpsız, "Researching of the agile model driven processes", Ulusal Yazılım Mimarisi Sempozyumu, 2014.

[13] A. Cockburn, Agile Software Development, Addison-Wesley, December 2001.

[14] V.C. Nguyen and X. Qafmolla, "Agile Development of Platform Independent Model in Model driven Architecture", 3th International Conference on Information and Computing, pp. 344-347, 2010.

[15] D. Astels, "Test Driven Development: A Practical Guide", Prentice Hall, 2003.

[16] S. Tilley, "Test-Driven Development and Software Maintenance", Proceedings of the 20th IEEE International Conference on Software Maintenance , pp. 488-491, 2004.

[17] Interactive Wireframe Software & Mockup Tool, http:// http://www.axure.com/, retrieved: 03.2015.

[18] Creating Your First Mockup | Balsamiq, http:// http://support.balsamiq.com/customer/portal/articles/871902-creating-your-first-mockup, retrieved 03.2015.

Critical Issues in SPI Programs: A Holistic View

Cristiane Soares Ramos, Ana Regina Rocha
COPPE/UFRJ
Federal University of Brazil
Rio de Janeiro, RJ, Brasil
email: cristianesramos@unb.br, darocha@cos.ufrj.br

Káthia Marçal de Oliveira
LAMIH, UMR CNRS 8201
Université de Valenciennes
Valenciennes, France
email: kathia.oliveira@univ-valenciennes.fr

Abstract - **With the competitive market in the domain of Information Technology (IT), companies are forced to expand their field to offer not only the service of software development, as well as direct customer services (e.g., help desk) or specialist technology services (e.g., testing and quality certification of systems). In this context, to apply software process improvement (SPI) requires addressing different needs and, therefore, the use of different models and standards. In this paper, we propose to use a holistic view to look at the whole company in planning a SPI program. This is done by defining that which we call critical issues, as elicited from stakeholders on the strategic, tactical and operational levels of the company. Critical issues are those which the stakeholders consider very important to have in the company in terms of software process but that they perceive the company does not have or is deficient in. Critical issues support SPI planning as well as a return on investment (ROI) evaluation, in a complete strategy for SPI institutionalization. Preliminary results obtained show that the critical issues are useful to steer the planning effort in the SPI program so to deal with that which is critical for the company.**

Keywords - Software Process; Improvement; multimodel; benefits; ROI.

I. INTRODUCTION

It is a fact well known that software process improvement (SPI) programs should be in line with the business goals of the organization [1][2]. This is particularly difficult when a company has a wide field of expertise, in services ranging from software development to direct customer service. Moreover, to define SPI programs, companies have several standards (e.g., ISO 9000 [3], ISO 29110 [4]) at hand along with quality models (e.g., CMMI [5[-[7], and MPS.Br [8][9]). The solutions proposed for SPI programs rarely include different standards and models in the same program nor consider the legacy of models already implemented in the organization.

This situation contributes to uncoordinated efforts and consequently concurrent improvement actions on different hierarchical levels or different areas of the company that choose a suitable model for their own needs, without taking into account other company initiatives [10]. To address this problem, we argue that it is essential to have a holistic view for the planning and evaluation of the benefits from SPI programs.

As defined in the Cambridge dictionary, holistic view means, 'dealing with or treating the whole of something or someone and not just a part'. We proposed, therefore, to take into account the expectation of benefits of the stakeholders from different levels of the IT Company (strategic, tactical, and operational) to plan the SPI program. To that end, we argue that we should identify critical issues to plan the SPI program in a way that it could later be used to evaluate the ROI with the institutionalization of the SPI program.

Critical issues are prioritized based on the company's business goals. They guide the goals of the SPI program and support the selection of one or more models/standards that best fit company needs. That means we consider the use of a multimodel approach in the definition of a SPI program.

This paper describes the work done to support the identification of such critical issues and also how to use them in the planning of a SPI program, using a real industry case.

The next section briefly introduces the idea of a multimodel approach for SPI. Then, Section 3 shows our approach to plan a multimodel program for SPI, using critical issues. Section 4 shows how to apply and use this approach in practice. Section 5 points at the advantages, limitations, and ongoing work avenues of this research

II. MULTIMODEL APPROACH FOR SPI

With the wide diversity of SPI standards, one of the biggest difficulties faced by organizations is the identification of the best-suited model to support them in achieving their business goal. In Brazil, a study on the evolution of software quality [11] showed a large adoption of the MPS-SW [8] and CMMI-DEV [5] models. Their use has brought benefits, such as higher customer satisfaction, increased productivity, and cost reduction. These and other benefits are often found in the literature as return on investment from these models (see, for example, [12-13]).

However, companies that have business features that go beyond the scope of improvement of these models require the implementation of other models. We can mention: (i) companies that develop software products and have a help desk or desk-service to meet the demands of their customers and (ii) companies that outsource the development of some lifecycle activities of their products.

In the first case, CMMI-SVC [7] and MPS-SV [9] can be used; in the second one, CMMI-ACQ [6]. Multinational companies also need to implement specific models, as required for contracts in different countries (e.g., the Brazilian model is being required in several contracts with Brazilian Government).

One can also find several other models/standards required and used in a smaller scale, such as ISO 9000 [3], MoProSoft [14], ISO 29110 [4], MPT.Br [15] (a Brazilian model for software testing).

A joint implementation of these models simultaneously allows treating different points of improvement in the organization in a more appropriate way. However, it is necessary to determine when the joint implementation is needed, which models are relevant, on which levels of maturity/capability they should be deployed, and how the improvement program should be structured towards a feasible implementation in the company, without generating unnecessary costs, wasted resources, rework and looking for the best ROI.

With this scenario in mind, the concept of multimodel environment emerged [16] as a result of the effort of companies to integrate models and international standards to achieve process improvement. Its use, however, requires an understanding and an interpretation of how different models co-relate, which makes an implementation of multimodel improvement a complex task. Thus, the use of a strategy to harmonize and match these models is a critical success factor [17].

The harmonization implies defining solutions to support the company [10]: (1) determine and understand which models will help it accomplish its corporate goal; (2) understand both the differentiating and the overlapping features of these models; (3) create an organizational process focused on the company's mission and incorporating the features and contents of all models of choice.

In this context, Mirna et. al. [16] propose a method that focuses on the business goal and selects the standards and models that best fit the company, indicating what should be done to achieve such goals. PRIME (Process Improvement in Multimodel Environments) [18] proposes the alignment between business and improvement goals. Models are selected and categorized according to their type of contribution to company goals, and the points of intersection between models are determined. Both studies measure the results, evaluating the achievement of the organizational goals. Other examples of multimodel approaches can be found in [19].

Several benefits justify an investment in multimodel approaches, such as [18]: focus on the business rather than focusing on the model; cultural change in relation to the establishment of the processes in the company; measurement system; robustness and effectiveness of the organizational approach in the long run. However, a multimodel approach also presents some challenges [20][21]: getting the commitment from senior management; determining the organization's strategy, integrating and coordinating training; integrating measures so that they do not target the adopted models; knowing the differences and similarities regarding the various models adopted in the design of improvements.

We argue that, to support a multimodel approach, one needs a holistic view of the company in a way that the potential benefits achieved by the SPI program are visible to all the stakeholders involved.

III. A HOLISTIC VIEW FOR SPI VIA THE IDENTIFICATION OF CRITICAL ISSUES

Our proposal for process improvement is based on the idea that a company that decides to carry out a SPI program wishes to have some benefits that justify such investment. Thus, the characterization and understanding of the needs of the company should be made judiciously and should guide both the definition of the improvement goal and the selection of models to be implemented, to enhance the possibility of getting a better return on the investment. By applying a holistic view we assume to look at the company as a whole and, therefore, to look for benefits for all the stakeholders involved in the SPI program.

To support this idea we propose to identify critical issues in the company. Critical issues are what all stakeholders or their representatives, on the different organizational levels (strategic, tactical, and operational) consider very important to have in the company in terms of process but sense the company does not have or has in a deficient manner. By identifying these critical issues, the company can better define a SPI Program plan and later evaluate the benefits attained, to start a new cycle of improvement, institutionalizing a continuous process for improvement.

As shown in Figure 1, the core of a holistic view is the identification of critical issues that can address different stakeholder needs. Based on these critical issues the continuous streamlining of the SPI is done as follows: (1) *Characterizing the company*: understanding the characteristics of the company as related to its field of expertise (software development, software maintenance, software testing, service desk, product marketing, acquisitions, etc.), the improvement program initiatives already undertaken, and its types of customers (government, national, and international).

At this point the **Critical Issues** (CI) are identified with the different stakeholders from the strategic, tactical and operational levels; (2) *Defining a SPI Program Plan* – defining the SPI plan, including improvement goals, models/standards to be followed, resources, risk analysis and mitigation, and an execution schedule; (3) *Executing the SPI Program Plan*: Carrying out the plan, and measuring results based on the critical issues identified; and (4) *Evaluating the* Benefits *of a SPI Program* (4) – evaluating the reach of the benefits with the stakeholders, considering the critical issues they identified. These benefits represent the ROI for the SPI program and if not attained, they can be re-considered in the next improvement cycle.

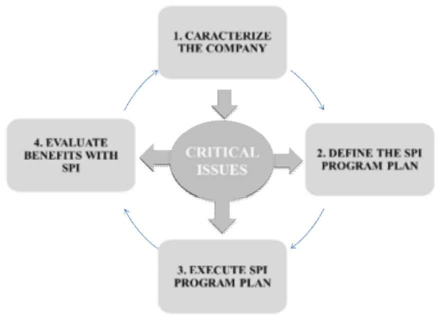

1. CARACTERIZE THE COMPANY
2. DEFINE THE SPI PROGRAM PLAN
3. EXECUTE SPI PROGRAM PLAN
4. EVALUATE BENEFITS WITH SPI
CRITICAL ISSUES

Figure 1. Continuous SPI with Holistic View

To support the identification of critical issues we defined questionnaires based on a detailed research on the benefits of SPI as shown in next subsection.

A. *Supporting the Identification of Critical Issues for SPI*

To define the questionnaires for the identification of critical issues we used a proposal that establishes the implementation of theoretical procedures (theoretical studies supporting the definition of the issues and their semantic validation) and experimental procedures (application of the questionnaire).

The theoretical procedures were supported by two main sources: (1) a systematic mapping of literature [22], where the benefits of Process Improvement Programs were mapped (in brief, from 112 papers found in literature, 28 were considered pertinent, and of those 34 different benefits for SPI were identified); and (2) the purpose of the processes for quality models in the context of Software (MPS-SW and CMMI-DEV models), Services (MPS-SV and CMMI-SVC models) and Testing (MPT.Br model). These models were picked due to their potential adoption by Brazilian companies.

As shown in Figure 2 the statements (i.e., issues) of the questionnaire were set in different categories: process issues and issues about benefits in SPI. Process issues were organized into common issues and specific issues, for each of the process types (software, services, or testing). Altogether there were 74 issues. This organization avoids the repetition of issues, making the overlapping of the process and help explicit in the definition of the SPI.

The definition of the questionnaire is based on [23] that adapted SERVQUAL questionnaires [24]. This way, all the issues are organized into two questionnaires: (i) one to collect the Importance of the issue for the respondent, and (ii) the second one to collect one's Perception of each issue in the company.

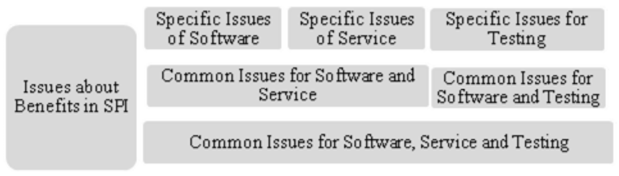

Issues about Benefits in SPI
Specific Issues of Software
Specific Issues of Service
Specific Issues for Testing
Common Issues for Software and Service
Common Issues for Software and Testing
Common Issues for Software, Service and Testing

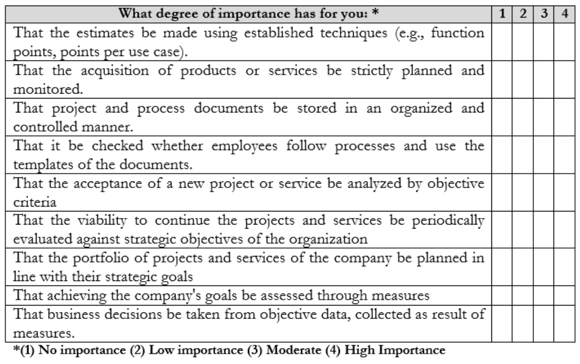
Figure 2. Categories of Issues in the Questionnaire

This way, each company stakeholder answers for each issue to the extent one considers it important for the organization according to one's point-of-view (strategic, tactical, or operational) and what one's perception is of the issue in the company. The issues are answered in a four-point Likert scale.

Critical issues are the ones identified as very important for the company but with a low level of perception by the stakeholders. Figure 3 and 4 show respectively an example of the issues presented in both questionnaires (to collect the degree of Importance and degree of Perception).

Each one of the statements in the questionnaire (i.e., the issues) is associated to some quality model or quality processes from a model, in a way that can further support the identification of which model/process should be addressed in the company. For instance, in Figures 3 and 4, the first statement from each questionnaire is related to Project Planning/Work Planning; the second statement is related to the Supplier Agreement Management from these models; the third to Configuration Management and the fourth to Process and Product Quality Assurance.

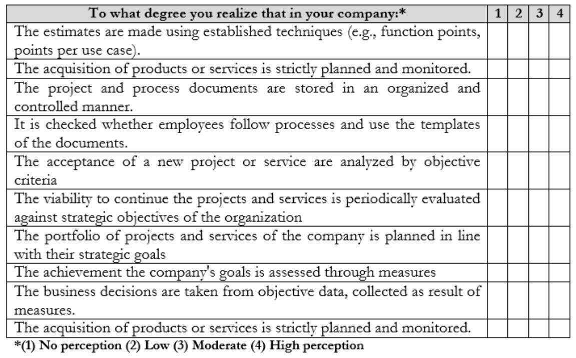
Two experts in Software Engineering, specifically on software process improvement, reviewed all the statements defined to ensure compliance with the following criteria: clearness (intelligible even for less experienced respondents) and simplicity (expressing a single idea). Once the questionnaires were reviewed, two semantic validations with the participation of industry professionals were done.



What degree of importance has for you: *	1	2	3	4
That the estimates be made using established techniques (e.g., function points, points per use case).				
That the acquisition of products or services be strictly planned and monitored.				
That project and process documents be stored in an organized and controlled manner.				
That it be checked whether employees follow processes and use the templates of the documents.				
That the acceptance of a new project or service be analyzed by objective criteria				
That the viability to continue the projects and services be periodically evaluated against strategic objectives of the organization				
That the portfolio of projects and services of the company be planned in line with their strategic goals				
That achieving the company's goals be assessed through measures				
That business decisions be taken from objective data, collected as result of measures.				

***(1) No importance (2) Low importance (3) Moderate (4) High Importance**

Figure 3. Some statements of the questionnaire of Importance



To what degree you realize that in your company:*	1	2	3	4
The estimates are made using established techniques (e.g., function points, points per use case).				
The acquisition of products or services is strictly planned and monitored.				
The project and process documents are stored in an organized and controlled manner.				
It is checked whether employees follow processes and use the templates of the documents.				
The acceptance of a new project or service are analyzed by objective criteria				
The viability to continue the projects and services is periodically evaluated against strategic objectives of the organization				
The portfolio of projects and services of the company is planned in line with their strategic goals				
The achievement the company's goals is assessed through measures				
The business decisions are taken from objective data, collected as result of measures.				
The acquisition of products or services is strictly planned and monitored.				

***(1) No perception (2) Low (3) Moderate (4) High perception**

Figure 4. Some statements of the questionnaire of Perception

TABLE I – CRITERIA FOR QUESTIONNAIRE ADAPTATION

Nature of work in companies	**Common issues**			**Specific issues**		
	SW, SV, T	SW, SV	SW, T	SW	SV	T
Soft. Develop. or maintenance	X	X	X	X	-	-
Service-desk	X	X	-	-	X	-
Software testing	X	-	X	-	-	X

SW=Software; SV=Service; T=testing

The application of the full questionnaire depends on the nature of the work done by the companies as shown in Table I. Issues not pertinent are eliminated from the questionnaire before it is applied.

B. Supporting the SPI Program Plan

Critical issues are used to support the planning of the SPI Program. In order to reduce the degree of subjectivity in this planning a need was seen to adopt a method that would aid the company to grasp what processes deal with the more critical issues and, when needed, prioritise and set the order of the processes to be implemented in the different improvement cycles.

QFD (Quality Function Deployment) aims at relating product requirements to those of the clients, seeking to identify how the product requirements are used to build a product that meets client requirements [25]. The strategy put forward in this work does not aim at the building of a product but rather the planning of a SPI Program that addresses the critical issues identified by the company.

Thus, the QFD was adjusted to relate the critical issues (client requirements) with the software processes (product requirements) so to prioritise the software processes that will be used in the definition of a plan for the improvement program. The main output product is the relative importance (RI) index of the processes which will guide the prioritisation of the processes in the planning of cycles for the improvement of the software process.

IV. OUR APPROACH IN PRACTICE: AN INDUSTRY CASE

In this section we show how the questionnaires described in the section above were applied and used to define the SPI program plan in a very small Brazilian company.

A. Characterizing the Company and Identifying Critical Issues

The company has customers in Brazil and overseas. It does not hire services and/or software development from other companies and it does not have the Brazilian Government as a client. It works with software development and maintenance by demand, develops and sells final software products (components, COTS). It also has a help-desk service for its customers. Currently, one of its projects is to work as a software testing factory. As regards its history of process improvements, the company was rated as first level in the 2009 Brazilian model (MPS-SW), although the processes are only partially followed nowadays.

TABLE II – CRITICAL ISSUES IDENTIFIED AS PER CATEGORY

Common Issues			**Specific Issues**		
SW, SV, T*	**SW, SV**	**SW, T**	**SW**	**SV**	**T**
2	10	2	0	7	8
7%	34%	7%	0%	24%	28%
Accumulated (Common + Specific)			48%	66%	41%

**SW=Software, SV=Service, T=testing*

The questionnaires were answered by all company employees (stakeholders). As a result, 29 critical issues in the processes and 13 critical issues in the benefits of SPI issues were identified. Table II shows the number of critical issues as per category (see Figure 2). The majority of critical issues are related to services (66%), followed by software (48%), and Testing (41%). We also found that 48% are common issues in service, software and testing quality models.

We should like to point that, in defining improvement actions that address critical issues, the processes to be implemented should consider the overlap between models and also the characteristics that differentiate them. The critical issues are discussed below from the perspective of the respondents' profiles: (a) Technical team - operational level; (b) Project manager - tactical level; and (c) Senior manager - strategic level.

1. Critical Issues in Processes

For the technical team the critical issues are more concentrated in the realm of testing: control of test environment incidents; standardization of test results in projects; quantitative assessment of quality objectives; identification and elimination of root causes of defects; and implementation of peer review. As regards services, the critical issues are about the control of the implementation of changes and the ability to monitor performance requirements. There are expectations about improvements in risk management and knowledge sharing amongst the employees. It is interesting to note that there are no common critical issues between the technical team and the project manager.

From the perspective of the project manager, the critical issues are related to: the use of estimation techniques (C01); management of the projects and services portfolio: planning of the services portfolio based on strategic planning (C15); feasibility assessment for project continuation against the strategic goals of the company (C06). It is expected that the achievement of the business goals of the company are evaluated by measurements (C07). The critical issues specific to the services area refer to the definition of mechanisms for the development of new services; compliance with Service Level Agreements; control of budget and accounting services; information security and communication of information on services.

As regards testing, the critical issues for the project manager are: managing the completion of testing activities; automated source code assessment; re-use of work products previously developed in other projects and control of the use of new tools to support the testing process. Finally, the critical issues for senior managers concern supplier management agreements; use of estimation techniques (C01); management of the projects and services' portfolio (C05, C06, C15); knowledge sharing amongst employees; implementation of peer reviews; automated assessment of the source code; use of criteria for making important decisions; use of measures to evaluate the achievement of company goals (C07); and control of budget and accounting services.

As expected, no common critical issue was found in the three profiles. The expected benefits vary according to the responsibilities and activities performed by the respondents. This reinforces the importance of involving different stakeholders on different hierarchical levels in the definition of the critical issues.

2. *Critical Issues in SPI Benefits*

Critical issues in SPI benefits are expectations of benefits that are results of the entire SPI program, but some of them may be associated with specific processes (e.g., B01, B04, B08, B22, and B24).

For the technical team, the critical issues concern the monitoring of projects (B24), reduction of re-work, and the quality of the work life. Compliance projects schedules (B22) are the only common critical issue of the technical team with the project managers. The critical issues raised by the project manager refer to expectations of improvements related to the accuracy of estimates (B04); prompt answer to market demands; and a greater market share. Critical issues for senior management relate to company growth in terms of number of projects and clients in Brazil and overseas; ensuring that the cost of projects and services are kept as planned (if there is no change in requirements) (B08); better understanding of the tasks and responsibilities by the project teams (B01); and greater market share.

B. *Defining a SPI Program Plan*

The analysis of the critical issues identified in the company indicates the need of a multimodel approach with improvement actions for software, services and testing. As the company aims at increasing the number of projects and customers abroad, the use of quality models with international recognition is advisable; in this case the CMMI-DEV and CMMI-SVC. However, the MPT.Br and MPS-SW/MPS-SV models should be used for some specific issues related to testing and knowledge management, as there are no CMMI processes that address these issues.

To define the SPI Program Plan we should analyze all critical issues, bearing in mind that they should be addressed in several improvement cycles. Deciding which questions to address in the first place is no trivial task and should be done with caution. A bad start places the entire improvement program at risk. Thus, it is recommended that the critical issues identified be evaluated by senior management (strategic level), based on different criteria. The first one is the alignment with the business goals of the company. Once this aspect is ensured, it should also take into consideration company size, its legacy as regards SPI, target budget for SPI and the natural dependence between processes (i.e., processes presented on a high maturity level depend on the implementation of other processes, and therefore are hardly possible at first).

As shown in Table II, the company identified many critical issues to tackle and most of them relate to services (66%). However, the main company business is software development and maintenance and as a result of that senior management chose to first face the critical issues in the domain of software (48%). In spite of that, many of such issues are common to the services and testing areas.

The QFD approach was applied, considering, as client requirements, the critical issues mapped in the Software context. Having established the correlations between the critical issues and the processes, in the end the Relative Importance (RI) Index was found for the processes. Table III shows the four highest RI processes and the CI associated to such processes.

The processes with highest RI index are prioritised for implementation. Based on this, 4 processes were chosen to be implemented in the first SPI cycle: PP, PMC, M&A and PFM, as they had already been implemented in the company (PP and PMC) and are the basis for others. We chose also REQM (requirements management) because it has many correlation with other processes. The process M&A is so important for the analyses of benefits of the SPI program. For the second SPI cycle were selected the engineering process Verification and Validation, because of the high interest in the improvement to the quality of the product and the business goal of answer the market demand software testing projects.

The idea is to implement these processes in a capability level 2 (of CMMI) in the first cycle and to improve continuously in each new cycle of improvement. Considering the process to be implemented in the first and second cycles (six months each one), they cover software, service and testing quality models. Therefore, the definition of the process specific for the company should ensure the incorporation of all the required features for each model. Each cycle will be

At the end of each improvement cycle the perception questionnaire will be reapplied to measure if the level of perception of the critical issues are improving, which may influence in the prioritization of critical issues for the next improvement cycles. Moreover, to execute the SPI Program Plan a set of measures is proposed for each process to allow the evaluation of improvement actions and a return of the investment.

TABLE III – TOP FOUR RI INDEX FOR THE PROCESSES

Process	RI	CI
PP - Project Planning	13,67%	C01, B01, B04, B22, B24
PMC - Project Monitoring and Control	10,73%	
M&A - Measurement and Analysis	7,17%	C07
PFM - Portfolio Management	7,12%	C05, C06, C15

(*) B – Benefit in SPI, C-common to between quality models

V. CONCLUSION

This paper presents an approach to support the implementation of SPI program based on a multimodel environment and on the eliciting of CI considering the whole company. CI are prioritised according to business goals of the company and guide the objectives, scope and planning of the SPI Program. We point as an advantage of this approach the participation of stakeholders from the different levels and profiles in the identification of critical issues; and, the idea of promoting the visibility of benefits achieved by the improvement actions considering the scope of the initial expectations of the stakeholders.

The holistic view occurs in two different dimensions: the horizontal and the vertical. The horizontal dimension goes through many lines of business. The vertical dimension goes through only one line of business considering the views of various hierarchical levels. The ability to cover these two dimensions depends on the structure of the company and their own desire to address them.

It is necessary to do a better systematisation of the final meeting, to support the alignment of the business goals with the prioritisation of critical issues and the definition of the number of SPI cycles. The general recommendations we made were essential for the decision making process, although the experience and certainty of top management were key factors for the definition of the SPI plan.

We are currently working on this weakness, investigating the pick chart technique as proposed by Lean [26] and QFD, to construct a House of Quality for the critical issues and the processes. Other ongoing works include the definition of a catalogue of measures for all CI and their related process to allow an evaluation of the ROI through the analysis of the benefits reaped; to support the company, presented in this paper in the execution of its SPI Program plan; and, to apply this approach to other companies.

REFERENCES

.[1] J. Guzmán, H. Mitre, A. Amescua, and M. Velasco, "Integration of strategic management, process improvement and quantitative measurement for managing the competitiveness of software engineering organizations," Software Quality Journal, vol. 18, pp. 341-359, 2010.

[2] T. Birkholzer, C. Dickmann, and J. Vaupel, "A Framework for Systematic Evaluation of Process Improvement Priorities," in Software Engineering and Advanced Applications (SEAA), 2011 37th EUROMICRO Conference on, 2011, pp. 294-301.

[3] ISO 9000:2005, "ISO/IEC 9000:2005 Quality systems - Fundamentals and vocabulary". International Organization for Standardization, 2005.

[4] ISO 29110, "ISO/IEC 29110-4-1: Software engineering -- Lifecycle profiles for Very Small Entities (VSEs)-- Part 4-1: Profile specifications: Generic profile group" International Organization for Standardization, 2011.

[5] SEI, "CMMI for Development, Version 1.3," Carnegie Mellon University.2010. Available: http://cmmiinstitute.com/resources/cmmi-development-version-13 [retrieved: March, 2015]

[6] SEI, "CMMI® for Acquisition, Version 1.3," Carnegie Mellon University. CMU/SEI-2010-TR-032, 2010. Available:http://cmmiinstitute.com/resources/cmmi-acquisition-version-13 [retrieved: March, 2015]

[7] SEI, "CMMI® for Services, Version 1.3," Carnegie Mellon University,2010. Available:http://cmmiinstitute.com/resources/cmmi-services-version-13[retrieved: March, 2015]

[8] SOFTEX, MPS.BR - MPS General Software Guide: Associação para Promoção da Excelência do Software Brasileiro, 2012. Available:http://www.softex.br/wp-content/uploads/2013/07/MPS.BR_Guia_Geral_Software_2012-c-ISBN-1.pdf [retrieved: March, 2015]

[9] SOFTEX, MPS.BR - MPS General Service Guide: Associação para Promoção da Excelência do Software Brasileiro, 2012. Available:http://www.softex.br/wp-content/uploads/2013/07/MPS.BR_Guia_Geral_Servicos_2012..pdf [retrieved: March, 2015]

[10] J. Siviy, P. Kirwan, L. Marino, and J. Morley. (2008, March). The value of harmonizing multiple improvement technologies: a process improvement professional's view. Available: http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=29159 [retrieved: March, 2015]

[11] P. Neto, G. Abib, M. Gomel, J. Pécora, A. Junglos, F. Ishi, and G. Braga, "Software quality evolution in Brazil from 1994-2010 based on research and projects of PBQP Software", Ministério da Ciência, Tecnologia e Inovação, Brasília, 2011.

[12] G. Travassos and M. Kalinowski, iMPS 2011Performance results of the companies that adopted the MPS model 2008-2011: Softex, 2012. Available: http://www.softex.br/wp-content/uploads/2013/08/iMPS-2011-Resultados-de-Desempenho-das-Empresas-que-Adotaram-o-Modelo-MPS-de-2008-a-2011.pdf [retrieved: March, 2015]

[13] M. Unterkalmsteiner, T. Gorschek, A. Islam, C. Cheng, R. Permadi, and R. Feldt, "Evaluation and measurement of software process improvement-A systematic literature review," IEEE Transactions on Software Engineering, vol. 38, pp. 398-424, 2012.

[14] H. Okbata, Modelo de Processo para la industria del Software MoProSoft, Versión 1.3, 2005.

[15] SOFTEX-RECIFE, MPT.Br Reference Guide Model: SOFTEX Recife, 2011. Available: http://mpt.org.br/mpt/wp-content/uploads/2013/05/MPT_Guia_de_referencia.pdf [retrieved: March, 2015]

[16] M. Mirna, M. Jezreel, C. Jose, S. Tomas, and A. Giner, "Advantages of using a multi-model environment in software process improvement," in Electronics, Robotics and Automotive Mechanics Conference (CERMA), 2011 IEEE, 2011, pp. 397-402.

[17] C. Pardo, F. Pino, F. Garcia, M. Piattini, and M. Baldassarre, "An ontology for the harmonization of multiple standards and models," Computer Standards & Interfaces, vol. 34, p. 12, 2012.

[18] J. Siviy, P. Kirwan, J. Morley, and L. Marino. (2008, March). Maximizing your process improvement ROI through harmonization. Available: http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=28907 [retrieved: March, 2015]

[19] C. Pardo, F. Pino, F. García, M. Piattini, and M. Baldassarre, "Trends in harmonization of multiple reference models," in evaluation of novel approaches to software engineering. vol. 230, L. Maciaszek and P. Loucopoulos, Eds., ed: Springer Berlin Heidelberg, 2011, pp. 61-73.

[20] L. Ibrahim, "A process improvement commentary," Crosstalk: The Journal of Defense Software Engineering, vol. 21, p. 4, 2008.

[21] U. Andelfinger, A. Heijstek, and P. Kirwan. (2006). A unified process improvement approach for multi-model improvement environments. Available: http://www.sei.cmu.edu/library/abstracts/news-at-sei/feature1200604.cfm [retrivied: March, 2015]

[22] C. Ramos, K.. Oliveira, and A. Rocha, "Towards a strategy for analysing benefits of software process improvement programs," in The 25th International Conference on Software Engineering & Knowledge Engineering, Boston, 2013, p. 6.

[23] J. Xexéo, "Sistema de informação como instrumento de programas de qualidade," Doutorado, COPPE, Programa de Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2001.

[24] A. Parasuraman, V. Zeithaml, and L. Berry, "SERVQUAL - A multiple-item for scale for measuring consumer perceptions of service quality," Journal of Retailing, vol. 66, pp. 12-40, 1988.

[25] F. Franceschini, Advanced quality function deployment: CRC Press LLC, 2002.

[26] G. Mike, D. Rowlands, and B. Castle, What is Lean Six Sigma?: McGraw-Hill, 2004.

Data Verification of Telecommunication Projects for Risk Assessment Models

Ayşe Buharali Olcaysoy
Yildiz Technical University & Turkcell Technology
Istanbul, Turkey
ayse.buharali@turkcell.com.tr

Oya Kalipsiz
Yildiz Technical University
Istanbul, Turkey
kalipsiz@yildiz.edu.tr

***Abstract*—Every project is important and has some risks. However, the definition of risk needs to be clarified, since risk is usually mixed with other project concepts such as constraint and problem. Risk represents future uncertain events with a probability of occurrence and a potential for loss. It is possible to reduce or even eliminate this negative impact with risk management methods. In risk management, risk models are created to discover possible project risks in early stages of the project. Risk models can also be used to predict the success of a project in the very beginning. In order to create a predictive risk model, a large dataset is required. In this work, we created a risk dataset containing 357 projects of a telecommunication company using their records spanning two years between 2010 and 2012.**

Keywords-Software Project Management; Software Risk Management.

I. INTRODUCTION

Each year Standish Group publishes CHAOS manifesto, which shows percentage of successfully completed software projects in global companies. According to CHAOS manifesto, only 39% of the projects were completed successfully (delivered on time, on budget, with required features and functions) in 2012. This ratio is still very low, even though it is increasing compared to previous years [1]. Risk management plays an important role in raising this ratio. Increasing the number of academic studies on different aspects of risk management also shows the awareness of the importance of risk management. Verna et al. [2] conducted a survey to investigate risk and risk mitigation strategies in global software development in 2013. In this survey, authors investigated 37 papers reporting 24 unique global software development projects. They also reported that the number of studies on risk management on global software development is increasing every year.

Risk assessment is an important part of risk management. Risk assessment enables a project manager to evaluate the possible risks in the early phases of project life cycle. In risk assessment, a model is constructed to discover possible risks or to evaluate the effects of risks on the progress of the project [3]. Our aim is to create a predictive risk model to discover and analyze risks of a software project in the early phases. In order to create such, a model a large dataset is required. However, to our best knowledge, there is no such dataset.

In this study, we collected a risk dataset related to internal projects of a company, which is operating in the Turkish telecommunication market. In order to create a clean dataset, several preprocessing and feature selection methods were performed. Consistency of the created risk dataset is verified through clustering and statistical distribution.

The remainder of this paper is organized as follows; Section II summarizes the existing work on risk analysis and assessment. In Sections III and IV, we give information on our proposed risk assessment model and detailed information about data preprocessing, feature selection and verification steps. We conclude this paper with Section V.

II. RELATED WORK

Many risk management studies refer to the studies of Pretty and Briand [4]. They developed a tool, namely METRIX, for software risk analysis and management. The tool employs a modeling technique that is based on the Optimal Set Reduction algorithm [4].

Foo and Muruganantham have developed SRAM (Software Risk Assessment Model). SRAM is determined based on the results of the survey on the outcomes of past projects. The quality of the project in SRAM, time and cost of the criteria identified nine critical elements of risk relationship [5]. This value is determined only according to the risks associated with the internal dynamics.

In 2006, Jiamthubthugsin and Sutivong proposed a risk assessment model [6], which is based on an assumption that evolutionary cycles can be modeled by Weibull's family distribution. The factors used in the model are requirement volatility, staff productivity, software complexity and development time.

The study published in 2008 by Gupta and Sadik, provided a software risk assessment and forecasting model, SRAEM (Software Risk Assessment and Estimation Model) [7]. Using this model, a near-success of software project with accuracy can be estimated. This model not only performs a risk assessment, but also estimates the risks of software project.

Risk management also plays an important role in software architecture decisions. Vliet and Poort demonstrated how risk and cost affect making a decision about the software architect by their model RCDA (The Risk and Cost Driven Architecture) [8].

III. RISK ASSESSMENT MODEL FOR SOFTWARE PROJECTS

Several methods, including Monte Carlo simulation [9][10][11], COCOMO (Constructive Cost Model) [12][13] and data mining techniques [14][15], have been proposed to create a risk assessment model. In our future work, we are planning to create a new risk assessment model that is able to predict possible risks of a given project in the early stages. Using this model, we also plan to predict whether a given project will succeed or fail.

In order to create this model, we plan to use a classification algorithm that is able to dig out the most similar projects from a given project pool. Naïve Bayes classifier [16] and K-Means classification algorithms [17] are possible candidates for our model. We prefer The Naïve Bayes classifier because it is among the most effective at learning algorithms known and its accuracy is higher than the other learning algorithms [16]. The vectors in the K-Means classification algorithm can be replaced during the procedure and it always sets an algorithm that converges to a local optimum. The K-Means algorithm is faster and effective for most applications as the K-Means procedure is easily programmed [17].

A large dataset that contains a company's past projects is needed in order to create such a risk assessment model. However, such a dataset might contain irrelevant features and missing information. In this study, we collected a real life dataset from a telecommunication company and applied several preprocessing steps. Then, we validated this dataset using statistical features and clustering algorithms.

IV. EXPERIMENT AND RESULT

In this study, we used software projects, which were developed between 2010 and 2012 by a company operating in the Turkish telecommunication sector. Unutulmaz, Cingiz, and Kalipsiz worked on the same company's project data to examine the risk factors of the projects before the initiation phase [18][19]. This study discussed the whole risks during the project life-cycle are discussed. Risk data included the technical feasibility studies and the software projects that were developed according to the Waterfall methodology [20]. Data features of the risks involved in the project management database are shown in Table I.

TABLE I. RISK DATA

Feature Name	Description	Type
Risk No	Number generated by the system	Integer
Risk Status	Last status of risk	Multiple Choice
Project	Project name to which risk was belonged	Text
Assigned To	Responsible name of the risk	Multiple Choice
Risk Level	Risk level	Multiple Choice
Created By	Who created the risk record in the system	Multiple Choice
Created On	Risk created date	Date
Date Identified	Risk identified date	Date
Description	Description of the risk	Text
Probability	Risk probability	Multiple Choice
Risk Category	Risk category	Multiple Choice
Last Updated	The date of the risk register was last updated	Date
Detailed Description	Detailed description of the risk	Text
Action Plan	Plan for preventing the risk	Text
Closure Criteria	Risk criteria for closing	Text
Inform To	Person shall be informed in case of realization of the risk	Multiple Choice
Negative Impact	The magnitude of the impact of the risk	Multiple Choice
Phase Identified	Phase of the project when the risk is identified	Multiple Choice
Response	Action taken to risk	Text
Risk Factors	Risk factors affecting	Multiple Choice

A. *Feature Selection*

Dataset acquired from the company includes 19 features. Features which had test type were removed from the dataset as free text format information could not be formalized in an assessment model.

Features related to date ("Created On", "Date Identified", "Last Updated") and person name ("Assigned To", "Created By","Inform To") were also removed from the dataset since this information is not useful for assessment risks.

However, "Risk Factors" and "Risk Category" will be used in our future risk assessment model. These features are not used in this study because they can not be used for verification of the dataset (which is the target of this study). As a result, the following four data variables, shown in Table II, were chosen to be used in this study.

TABLE II. VALUES OF RISK FEATURES

Feature Name	Values
Risk Level	Critical, High, Normal, Low
Probability	Very High, High, Medium, Low, Very Low
Negative Impact	High, Very High, Tolerable ,Very Low, Low
Phase Identified	Test, Deployment, Planning, Analysis, Development, Closing

B. Preprocessing and Statistical Distribution

1658 risk records were extracted from the company's software projects between 2010 and 2012. Records that had any value were cleaned from the dataset; so, 434 records are remaining.

The statistical distribution of risk data according to the phases of software development life cycle is given in Figure 1. The number of risk records reduction from the analysis phase is to be expected. If this reduction will begin from the planning phase, our risk assessment model will reach the goal in the future because the risks will be estimated from the first phase of the project.

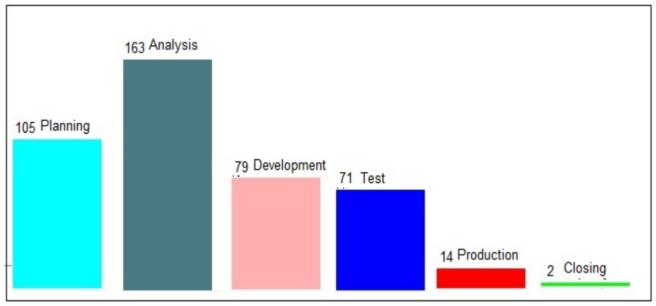


Figure 1. Risk Distribution according to the Project's Phases

The statistical distribution of the risk data according to the risk level is shown in Figure 2. This distribution will be used to compare with the results of the K-Means Clustering of the dataset in Part C.

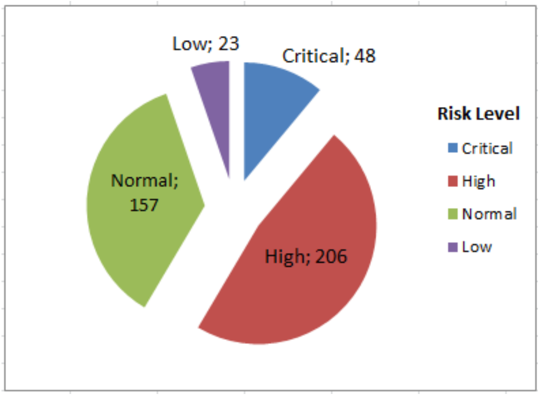


Figure 2. Statistical Distribution of Risks According to the Risk Levels

We examined the distribution of risk probability according to the risk level. This distribution didn't show any non-normal result, as shown in Figure 2. For example, the probability of high level risks is critical or high. If there will be a low risk level in this very high probability class, this result must be investigated.

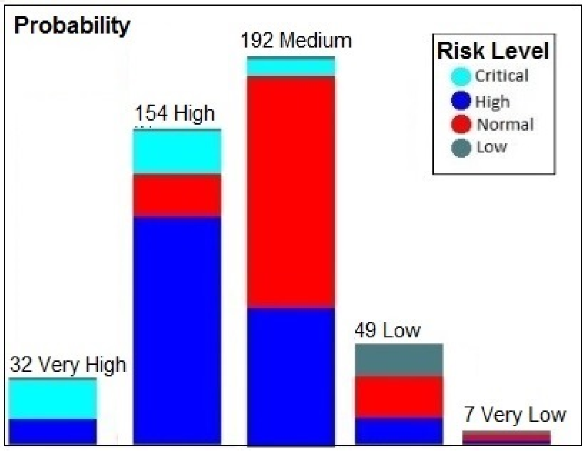


Figure 3. Statistical Distribution of Probability According to the Risk Levels

The relation between the negative impact of risk and level of risk was also examined in our study. The distribution of 58 records that had very high negative impact seems normal because the risk level of these records is critical, high or normal.

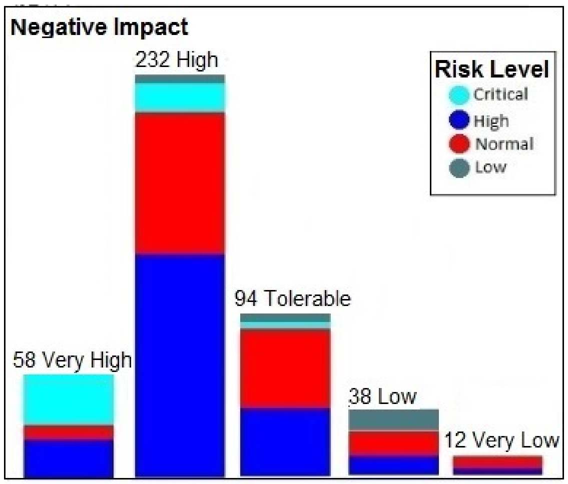


Figure 4. Statistical Distribution of Negative Impact According to the Risk Levels

C. *Preliminary Validation Analysis on the Data Set*

The dataset was formed by the project managers. There was not any automatic calculation or any information received directly from the project management process. Therefore, the data were open to human error. For this purpose, the risk probability and the negative impact, according to the four levels of risk, were tried to be considered by using the K-Means Clustering method. Table III showing the result of 434 records was obtained by applying K-Means Clustering to "probability" and "negative impact" features.

TABLE III. THE POSSIBILITY OF NEGATIVE EFFECTS, ACCORDING TO THE K-MEAN DISTRIBUTION

	Total Data	**Cluster 0**	**Cluster 1**	**Cluster 2**	**Cluster 3**
Number of Record	434	99	232	53	50
Features					
Probability	Medium	Medium	Medium	High	Medium
Negative Impact	High	Very High	High	Tolerable	Tolerable

Table IV shows the results of the risk levels clustering. The critical risks were assigned to the cluster 0. This result seems to make sense. However, the low level risks were assigned to the cluster 2 instead of the cluster 3. This result is needed to be investigated.

TABLE IV. THE LEVEL OF RISK ACCORDING TO THE DISTRIBUTION OF THE CLUSTERS

Original Risk Level	**Cluster 0 (CRITICAL)**	**Clsuter 1 (HIGH)**	**Cluster 2 (LOW)**	**Cluster 3 (NORMAL)**
Critical	**27**	16	4	1
High	33	**129**	29	15
Normal	26	82	16	**33**
Low	13	5	**4**	1

Risks, according to the only negative impact obtained clustering results, are in Table V; the results obtained by the level of risk we ran are shown in Table VI.

TABLE V. CLUSTERING RESULTS BY NEGATIVE EFFECTS

	Total Data	**Cluster 0**	**Cluster 1**	**Cluster 2**	**Cluster 3**
Number of Record	434	70	232	94	38
Negative Impact	High	Very High	High	Tolerable	Low

TABLE VI. THE LEVEL OF RISK ACCORDING TO THE DISTRIBUTION OF THE CLASS – THE NEGATIVE IMPACT

Original Risk Level	**Cluster 0 (CRITICAL)**	**Cluster 1 (HIGH)**	**Cluster 2 (LOW)**	**Cluster 3 (NORMAL)**
High	27	**129**	39	11
Normal	15	82	**46**	14
Critical	**27**	16	4	1
Low	1	5	5	**12**

Table VII shows the clustering results using the method only distributed by probability. Table VIII shows the distribution of the risk level is much more accurate.

TABLE VII. CLUSTERING RESULTS BY PROBABILITY

	Total Data	**Cluster 0**	**Cluster 1**	**Cluster 2**	**Cluster 3**
Number of Record	434	39	49	154	192
Probability	Medium	Very High	Low	High	Medium

TABLE VIII. THE LEVEL OF RISK ACCORDING TO THE DISTRIBUTION OF THE CLASS –PROBABILITY

Original Risk Level	**Cluster 0 (CRITICAL)**	**Clsuter 1 (LOW)**	**cluster 2 (HIGH)**	**cluster 3 (NORMAL)**
High	14	13	**111**	68
Normal	3	20	21	**113**
Critical	**19**	0	12	8
Low	3	**16**	1	12

When we obtained all these results in Table IX, we recognize that it will be more useful to use the results by the K-Means Clustering according to two features (negative impact and probability) in our future risk assessment model.

TABLE IX. THE RISK LEVEL DISTRIBUTION

	CRITICAL	HIGH	NORMAL	LOW
Original Data	48	206	157	23
Results by K-Means Clustering with 2 Feature	99	232	50	53
Results by K-Means Clustering with Negative Impact Feature	70	232	94	38
Results by K-Means Clustering with Probability Feature	39	154	201	49

V. CONCLUSION AND FUTURE WORK

The statistical distribution of the risk dataset and K-Means Clustering's results proved that our risk dataset is appropriate for our risk assessment model.

In the next stage of our study, we decide to create a predictive risk model to discover and analyze risks of a software project by using fuzzy logic methods [21] and other intelligent methods [16][22]. Inspired by the study on risk assessment model in 2010 [23], we decided to use fuzzy logic. In this study, they created the software project risk assessment model was based on fuzzy theory, then the domain experts used fuzzy language to evaluate and calculate the probability and impact of risks [23]. We will use these methods to improve the relationships between risk and other project features in our model. For this purpose, we will collect other project features data such as "project category", "project size" or even "the experience of project managers". We also see that the project manager's experience is needed to be taken into account in our model.

ACKNOWLEDGEMENT

Thanks to Turkcell Technology for allowing us to use the data in our study in accordance with their information security policy.

REFERENCES

[1] The Standish Group. Chaos Manifesto 2013 Report. The Standish Group International, Inc., 2013.

[2] J.M. Verner, O.P. Brereton, B.A., Kitchenham, M. Turner, and M. Niazi, "Risks and Risk Mitigation in Global Software Development: A Tertiary Study", Information and Software Technology, vol. 56, 2014, pp. 54–78.

[3] C. Ravindranath Pandian, "Applied Software Risk Management – A Guide for Software Project Managers", Auerbach Publications, 2007, p. 128.

[4] B.E. Gayet and L.C. Briand, "METRIX: A Tool for Software-Risk Analysis and Management", Annual Reliability and Maintainability Symposium, 1994, pp. 310–314.

[5] S.W. Foo and A. Muruganantham, "Software Risk Assessment Model", IEEE International Conference on Management of Innovation and Technology (ICMIT), IEEE Press, vol. 2, 2000, pp. 536-544, doi: 10.1109/ICMIT.2000.916747

[6] W. Jiamthubthugsin and D. Sutivong, "Resource Decisions in Software Development Using Risk Assessment Model", Proceedings of the 39th Hawaii International Conference on System Sciences, 2006.

[7] D. Gupta and M. Sadiq, "Software Risk Assessment and Estimation Model", International Conference on Computer Science and Information Technology, 2008.

[8] E.R. Poort and H. Vliet, "RCDA: 'Architecting As A Risk- And Cost Management Discipline", The Journal of Systems and Software, vol. 85, 2012, pp. 1995-2013.

[9] G.S. Fishman, "Monte Carlo Concepts, Algorithms, and Applications", 3rd ed, Springer, 1999, p. 1.

[10] Electronic Publication: http://www.oracle.com/us/products/applications/crystalball/risk-analysis-overview-404902.pdf, [retrieved: February, 2015].

[11] Electronic Publication: http://www.palisade.com/risk/monte_carlo_simulation.asp, [retrieved: February, 2015].

[12] B.W. Boehm, Software Engineering Economics, Prentice-Hall Inc., 1981, pp. 329-342.

[13] R. Fairley, "Risk Management for Software Projects", IEEE Software, vol. 11, May 1994, pp. 536-544.

[14] J. Han and M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufmann. 2006, p. 5.

[15] H. Jiang, C.K. Chang, J. Xia, and S. Cheng, "A History-Based Automatic Scheduling Model for Personnel Risk Management", Computer Software and Applications Conference, COMPSAC 2007, 31st Annual International, IEEE Press, vol. 2, July 2007, pp. 361-366, doi:10.1109/COMPSAC.2007.25.

[16] T.M. Mitchell, Machine Learning, McGraww-Hill Science, 1997, pp. 177-178.

[17] J.B. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations", Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability 1. University of California Press, 1967, pp. 281–297.

[18] A. Unudulmaz, O. Kalıpsız, and M.Ö. Cingiz, "Risk Faktörleri ve Risk Değerlendirme Modellerinin Farklı Veri Setleri Üzerinde Gerçeklenmesi". UYMS, 2013.

[19] M.Ö. Cingiz, A. Unudulmaz A., and O. Kalıpsız, "Yazılım Projelerindeki Problem Etkilerinin Yazılım Mimarisi ile İlişkilendirilmesi", UYMK, 2012.

[20] W.W. Royce, "Managing The Development of Large Software Systems", Proceedings of IEEE WESCON 26 (August), 1970, pp. 1–9.

[21] F.M. McNeill and E. Thro, Fuzzy Logic a Practical Approach, AP Professional, 1994, pp. 13–14.

[22] W. Elmenreich, "Intelligent Methods for Embedded Systems", Proceedings of 1st Workshop on Intelligent Solutions in Embedded Systems(WISES03), June 2003, pp. 3–5.

[23] A. Tang and R. Wang, "Software Project Risk Assessment Model Based on Fuzzy Theory", International Conference on Computer and Communication Technologies in Agriculture Engineering, 2010, pp. 328–330

A Lightweight Approach to the Early Detection and Resolution of Feature Interactions

Carlo Montangero

Dipartimento di Informatica
Università di Pisa, Pisa, Italy
Email: monta@di.unipi.it

Laura Semini

Dipartimento di Informatica
Università di Pisa, Pisa, Italy
Email: semini@di.unipi.it

Abstract—**The feature interaction problem has been recognized as a general problem of software engineering, whenever one wants to reap the advantages of incremental development. In this context, a feature is a unit of change to be integrated in a new version of the system under development, and the problem is that new features may interact with the others in unexpected ways. We introduce a common abstract model, to be built during early requirement analysis in a feature oriented development. The model is common, since all the features share it, and is an abstraction of the behavioural model retaining only what is needed to characterize each feature with respect to their possible interactions. The basic constituents are the abstract resources that the features access in their operations, the access mode (read or write), and the reason of each access. Given the model, the interactions between the features are automatically detected, and the goal oriented characterization of the features provides the developers with valuable suggestions on how to qualify them as synergies or conflicts (good and bad interactions), and on how to resolve conflicts. We provide evidence of the feasibility of the approach with an extended example from the Smart Home domain. The main contribution is a lightweight state-based technique to support the developers in the early detection and resolution of the conflicts between features.**

Keywords–Feature interactions; State-based interaction detection; Conflict resolution.

I. Introduction

The feature interaction problem has been recognized as a general problem of software engineering [1] [2] [3] [4], whenever an incremental development approach is taken. In this broader context, the term *feature*, originally used to identify a call processing capability in telecommunications systems, identifies a unit of change to be integrated in a new version of the system under development. The advantages of such an approach lay in the possibility of frequent deliveries and parallel development, in the *agile* spirit. The feature based development is now becoming more and more popular in new important software domains, like automotive and domotics. So, it is worthwhile to take a new look at the main problem with feature based development: a newly added feature may interact with the others in unexpected, most often undesirable, ways. Indeed, the combination of features may result in new behaviours, in general: the behaviours of the combined features may differ from those of the two features in isolation. This is not a negative fact, per se, since a new behaviour may be good, from an opportunistic point of view; however, most often the interaction is disruptive, as some requirements are no longer fulfilled. For instance, consider the following requirements, from the Smart Home domain:

Intruder alarm (IA)	Send an alarm when the main door is unlocked.
Main door opening (MDO)	Allow the occupants to unlock the main door by an interior switch.
Danger prevention (DP)	Unlock the main door when gas/smoke is sensed.

Assuming a feature per requirement, it is easily seen that combining *Intruder alarm* and *Danger prevention* leads to an interaction, since the latter changes the state so that the former raises an alarm. However, an alarm in case of gas leak or a fire is likely to be seen as a desirable side effect, so that we can live with such an interaction. Also, the combination of the first two features leads to an interaction: an alarm is raised, whenever the occupants decide to open the main door from inside. However, this is likely to be seen as an undesirable behaviour, since the occupants want to leave home quietly.

In general, the process of resolving conflicts in feature driven development has the same cyclic nature: look for interactions in the current specification, identify the conflicts, resolve them updating the specification, cycle until satisfaction.

Many techniques have been proposed to automate (parts of) this process. The search for interactions by manual inspection, as we did above, is obviously unfeasible in practice, due to the number of requirements in current practice. It is also the step with the greatest opportunity for automation. The other steps need human intervention since, at the current state of the art, they cannot be automatized. However, as discussed in Section IV, what is still lacking, in our opinion, is the ability to detect the interactions, identify the conflicts and resolve them by working on a simple model, as it may be available at the beginning of requirements analysis, before any major effort in the development of requirements.

We introduce a technique to support the detection and resolution of feature interactions in the early phases of requirements analysis. The approach is based on a common abstract model of the state of the system, which i) is simple enough to induce a definition of interaction which can be checked by a simple algorithm, and ii) can be modified, together with the feature specification, taking care only of few, essential facets of the system.

The model is *abstract*, since it is an abstraction of the behavioural model retaining only what is needed to characterize each feature with respect to the possible interactions: the constituents of the model are *resources*, that is, pieces of the state of the system that the features access during their operations. To keep the model, and the analysis, simple, the

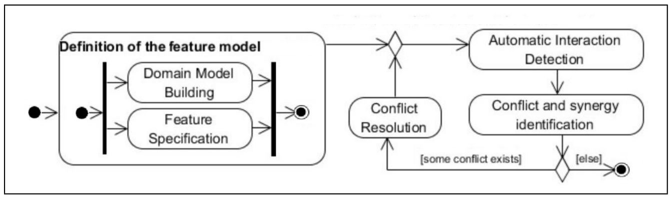

Definition of the feature model
Domain Model Building
Feature Specification
Automatic Interaction Detection
Conflict and synergy identification
Conflict Resolution
[some conflict exists]
[else]

Figure 1. Activities of the lightweight approach.

operations on the resources are abstracted to consider only their access mode, namely *read* or *write*. This way, however, we do not loose in generality since the essential cause of an interaction is a pair of conflicting accesses to a shared resource. In this respect we were inspired by notion of conflict between build tasks introduced by the CBN *software build* model [5].

The work required to build the abstract feature model can be amortized in two ways. The shared state models can be defined in a reusable and generic manner so that, for a given domain, they can be exploited in many different development efforts, as it happens in Software Product Lines; moreover, the model can be fleshed-out as requirements analysis proceeds.

In the following, we use the Smart Home domain described in [6] as a running example. The features are intended to automate the control of a house, managing the home entertainments, providing surveillance and access control, regulating heating, air conditioning, lighting, etc.

The next section describes the approach. Section III assess the correctness and completeness of the approach, and Section IV discusses related work. Finally, we draw some conclusions and discuss future work.

II. Approach

The lightweight approach to the detection and resolution of feature interaction consists in the activities presented in Figure 1 and elaborated in the next subsections.

Note that, from the point of view of the development process, there is no constraint on how the abstract feature model is built: in other words, domain model building and feature specification can be performed in sequence, as well as arm in arm as suggested in Figure 1. All the other activities are each dependent on the outcomes of the previous one in the list.

A. Domain model building

The description of the domain is an integral part of the abstract feature model. Its purpose is to provide a definition of the accessible resources, i.e., of the shared state that the features access and modify, detailed enough to allow describing the features precisely. There are no special requirements on the notation to express the model. In this paper, we use UML2.0 class diagrams for their wide acceptance.

Figure 2 shows the class diagram of part of the Smart Home design domain. The shared state is made up of the states of the all the resources, which may structured, like MainDoor, which owns a Lock.

The structure shown is not final, as new resources can be added by the analyst if he needs them, not only to introduce new features, but also to resolve conflicts, as it happens with refinement (Section II-E5).

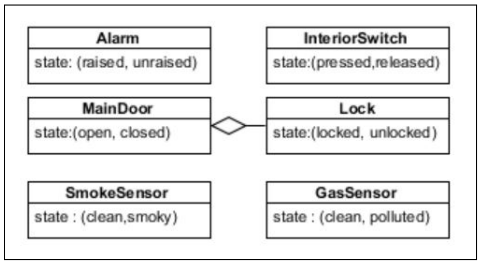


Figure 2. Smart Home Domain.

TABLE I. Feature specification template.

⟨name⟩ ⟨acronym⟩	read	write
⟨feature goal⟩	⟨label⟩ ⟨resource⟩ ⟨access reason⟩	⟨label⟩ ⟨resource⟩ ⟨access reason⟩

B. Feature specification

We model a feature defining: its goal; the resources in the domain model it accesses (r/w); the reason for each access. To make references short, we provide an acronym to each feature, and an integer label to each resource access. We introduce a template (Table I), which lists the feature name, its goal, and the involved resources, grouped in two sets (read or written) together with the reason for reading or writing each resource. The three features introduced in the previous section are represented in Table II.

Note that the accesses are numbered only for reference: no sequencing is implied, as the order of the accesses is abstracted away, as part of the simplicity of the model.

C. Interaction detection

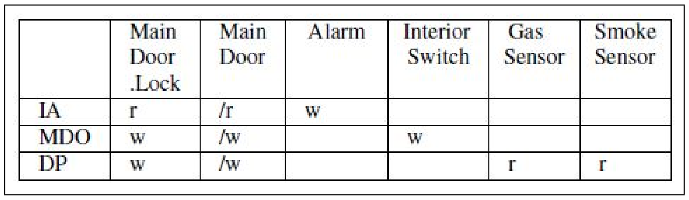
Our definition of feature interaction is based on the access mode (read or write) to the resources that make up the shared state of the system. The features access the resources in read mode to assess the state of the system, and in write mode to update it. By definition,

> there is an interaction whenever two features are composed in the same system, and at least one of them accesses in write mode at least a resource accessed also by the other, in any mode.

Let us reconsider the features defined above and the discussion in the previous section that led to detect some

TABLE II. Feature specification: IA, MDO, DP.

Intruder Alarm (IA)	**read**	**write**
To raise an alarm when the main door is unlocked.	(1) main door lock To know when to raise an alarm	(2) alarm To raise the alarm
Main door opening (MDO)	**read**	**write**
To manually unlock the door.	(1) InteriorSwitch To receive the command	(2) MainDoor.Lock To unlock
Danger prevention (DP)	**read**	**write**
To automatically unlock the door in case of danger	(1) GasSensor (2) SmokeSensor To know when there is an alert	(3) MainDoor.Lock To unlock



	Main Door .Lock	Main Door	Alarm	Interior Switch	Gas Sensor	Smoke Sensor
IA	r	/r	w			
MDO	w	/w		w		
DP	w	/w			r	r

Figure 3. Interaction detection matrix.

TABLE III. INTERACTING ACCESS TO MAINDOOR.LOCK.

Feature	Feature Goal	Mode	Access Reason
IA	To raise an alarm when the main door is unlocked.	r	To know if it has been unlocked
MDO	To manually unlock the door.	r	To unlock

interactions. We can rephrase it in term of resource accesses. For instance, consider the main door lock: accessing it in read mode allows knowing its current state, that is, if the door is locked or unlocked; accessing it in write mode allows locking or unlocking the door. Both IA and MDO access the door lock, in read and write mode, respectively. By definition, we have an interaction. Similarly, also IA and DP interact, since they access the same resource in the same way.

We are now ready to see how interaction detection can be automated: we build a matrix with a row per feature and a column per resource, and put *r* (*w*) in cell (F_i, R_i) when F_i accesses R_i in *read* (*write*) mode. The matrix is completed to take into account the composited resources of the domain. Indeed, potentially, the access to the field of a resource is an access to the resource itself and vice versa. We put */r* or */w* in a cell when the design domain entails that potentially there is a *derived* resource access. As an example, Figure 3 shows the matrix for IA, MDO, and DP.

In the interaction detection matrix, it is possible to identify all the pairs of interacting features: any pair of non empty entries in the same column with at least a *w* (or */w*) denote an interaction of the features in the selected rows. In the example, from the first column, we have (IA, MDO), (IA, DP), and (MDO, DP).

As an example where derived accesses are essential, let us assume a different version of DP: open the main door when gas/smoke is sensed. In order to find the interaction between the write on the main door (to open it) and the read on the door lock of feature IA, we need the derived read of IA on the main door.

The superclass relation (an example is given in the extended case study in Figure 4) is dealt with in a similar way: the access to a superclass is also an access to its subclasses.

D. Conflict and synergy identification

For each detected interaction a summarizing table is built, with the information on the goals of the interacting features and on the reasons for the interacting accesses.

As an example, table III captures the interaction (IA, MDO) on the main door lock.

Such a table will help the expert in the classification of the interaction and its resolution. At this point the expert

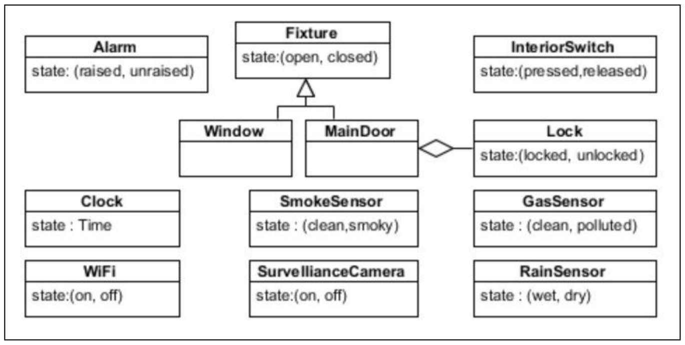


Figure 4. Extended Smart Home Domain.

can state if the interaction is a conflict, as clearly in this case, since we do not want the alarm to be sent when the opening is authorized, or a synergy. Instead, sending the alarm is useful when some danger sensor is triggered. Hence, there is a synergy between *Intruder Alarm* and *Danger Prevention*. Finally, also the interaction between *Main Door Opening* and *Danger Prevention* is a synergy. Indeed, the two features pursue the same goal, that is to open the door.

E. Conflict resolution

Once an interaction is recognized as a conflict in the analysis phase, we can take some actions to resolve it. In order to discuss them, we need to extend the working example. In addition to IA, MDO, DP we consider also a few more features, namely:

Air change (AC)	At 10:00 a.m. open the windows, at 10:30 a.m. close the windows.
Close window with rain (CW)	Close the windows when the rain sensor is triggered.
Video surveillance (VS)	Surveillance cameras are watched remotely via wifi.
Wifi switch-off (WSO)	Switch off the wifi at night.

The extended domain model is in Figure 4, and the specification of the new features is in Table IV.

Various routes to resolution have been proposed in the literature (see [7] [8] [9] for interesting surveys):

1) Restriction: Avoid tout-court that the conflicting features are ever applied in the same system. This is the resolution strategy to be taken when the two features have incompatible goals. In other cases, it is an option the expert can choose. In the running example, we could prevent *Video surveillance (VS)* and *Wifi switch-off (WSO)* from being applied in the same house.

2) Priority between the features: A weaker form of restriction is to guarantee that conflicting features are never applied at the same time. This behaviour can be obtained by defining priorities. Then, in the case two features are both enabled, only the one with higher priority is executed. In our example, priority can be likely used between *Air Change (AC)* and *Close window with rain (CW)*. Both features *write* on the resource *window*. In the case of rain at 10:00 a.m., we do not want the windows to be open.

3) Sequencing: This technique applies when the conflict is caused by a bad order of application of features that are triggered at the same time. In this case, to solve the conflict it is sufficient to force the features to be applied in the correct order.

4) Integration: According to this resolution strategy, the two interacting features are combined in a new one whose goal encompasses the goals of the two original ones.

VS and WSO can be integrated in a unique feature to switch off the wifi at night, and switch it on if an intruder is sensed, so that surveillance cameras can be watched from a remote machine.

5) Refinement: In any approach based on a shared state, we can apply another resolution strategy, considering if it is possible to add a new resource and make the two conflicting accesses insist on two distinct resources. Since two features conflict only because they access the same resource, this refinement solves the problem, by definition.

Think again of the conflict between *Intruder Alarm* and *Main door opening*. We might specify a new IA feature excluding the case where the door was unlocked using the interior switch. In some sense, we distinguish between the electrical and mechanical commands to the lock.

It is obvious that, after each resolution step, features are to be checked again to detect if the changes have solved the conflicts without introducing new ones.

The first three strategies do not change the features, but extend the model adding relations between them. A new structure is introduced that records mutual exclusions, priorities, and sequencing between features, as done, e.g., in [10]: This structure is used in the detection phase to disregard the pairs that might interact but will not, since incompatibilities have already been solved by the introduced relations.

III. Discussion

A discussion is needed on the soundness and completeness of our detection method with respect to existing ones. We restrict to design-time techniques, since we are interested in early detection. The most common way to define a feature interaction is based on behaviours [2]:

> A feature interaction occurs when the behavior of one feature is affected by the presence of another feature.

Soundness depends on the expert competence: the rough detection based on the shared resources access model can indeed render false positives, e.g., synergies. These will have to be discarded during the subsequent analysis. However, also the approaches analyzing the concrete behaviour cannot automatically distinguish between conflicts and synergies and some human intervention is still needed to complete the analysis.

On the other side, the completeness problem can be stated as: is it possible that the behaviour of two features interfere even if they do not access any shared resource? Consider the following example dealing with air conditioning (AC):

Natural AC (NAC)	If the room temperature is above 27 degrees and the temperature outside is below 25, open the windows.
AC switch-on (ACS)	If the room temperature is above 27 degrees switch-on the air conditioner.

These two features read the same resource, and act differently under identical conditions, but they do not interfere according to our definition. Do they interfere according to the behaviour based definition? The answer is no, the behaviour of each feature is not affected by the other one. Indeed, the conflict between the actions of opening the windows and switching on air conditioning can be stated only by an expert. Similarly, in our case, a relation between open windows and air conditioning can be recognized during domain description, permitting the conflict to be detected.

Sometimes features interactions are defined in an even more abstract way:

> Features interactions are conflicts between the interests of the involved people.

We express the personal interests in the feature goals, and base the analysis on it. Hence, we are compliant with respect to this notion. Understanding if the persons involved have conflicting interests is a different problem.

IV. Related Work

A. Programming features

Bruns proposed to address the problem at the programming language level, by introducing features as first class objects [1]. Our view is that such an approach is worth pursuing, but needs be complemented by introducing features for features in the early stages of the development process, namely in requirements analysis.

B. Requirements interaction

Taxonomies of feature interaction causes have been presented in the literature [3] [11]. Among the possible causes, there are interactions between feature requirements. We address here a special case of the general problem of requirements interaction. A taxonomy of the field is offered in [12]. It is structured in four levels, and identifies 24 types of interaction collected in 17 categories. It assumes that the requirements specification is structured in system invariants,

TABLE IV. More Smart Home Features

Air Change (AC)	**read**	**write**
To ventilate the house	(1) Clock To know when to open/close	(2) Window To open/close
Close window with rain (CW)	**read**	**write**
To close windows in case of rain	(1) RainSensor To know when to close	(2) Window To close
Video surveillance (VS)	**read**	**write**
To remotely control the house	(1) VideoCamera To read the recorded data (2) Wifi To access the camera	
Wifi switch-off (WSO)	**read**	**write**
To switch off the wifi when not used	(1) Clock To know when to switch-off	(2) Wifi To switch-off

behavioural requirements, and external resources description. Their analysis is much finer grained than ours. Should the two analysis be performed in sequence, our own should prevent the appearance of some interaction types in the second one, like those of the non-determinism type.

Nakamura et al. proposed a lightweight algorithm to screen out some irrelevant feature combinations before the actual interaction detection, on the ground that the latter may be very expensive [13]. They first build a configuration matrix that represents concisely all possible feature combinations, and is therefore similar in scope to our interaction matrix. However, it is very different in contents, since it is derived from feature requirements specifications in terms of Use Case Maps, which give a very detailed behavioural description of the features. The automatic analysis of the matrix lends to three possible outcomes per pair of features: conflict, no interaction, or interaction prone. In our approach, the automatic analysis gives only two outcomes: no interaction or interaction prone, as one might expect, given the simpler model.

Another similar approach is Identifying Requirements Interactions using Semi-formal methods (IRIS) [6]. Both methods are of general application, and require the construction of a model of the software-to-be. In IRIS the model is given in terms of policies, but the formality is limited to prescribing a tabular/graphical structure to the model. Both methods leave large responsibility to the engineers in the analysis. However, larger effort is required, and larger discretion is left to them in IRIS: in our approach, interaction detection is automatized, and the engineer can focus on conflict identification and resolution. Finally, the IRIS model is much more detailed than ours, so that resolving the identified conflicts may entail much rework, while resolution in our case provides new hints to requirements specification. The last consideration applies as well to the two previous approaches.

C. Design and run-time techniques

As another example of the ubiquity of the feature interaction problem, Weiss et al. show how it appears also in web-services [14]. The approach to design-time conflict detection entails the construction of a goal model where interactions are first identified by inspection, and the subsequent analysis is then conducted on a process algebraic refined formal model. Also in this case, our model is more abstract, and the two techniques may be used synergically.

In a visionary paper, Huang foresees a runtime monitoring module that collects information on running compositions of web-services, and feeds it to an intelligent program that, in turn, detects and resolves conflicts [15].

Several run-time techniques to monitor the actual behaviour of the system and detect conflicts and possibly apply corrective actions, are reported in the literature, as surveyed in [9]: for instance, [16] tackle the problem with SIP based distributed VoIP services; in [17] policies are expressed as safety conditions in Interval Temporal Logic, and they can be checked at run-time by the simulation tool Tempura. These techniques should be seen as complementary to the design-time ones, like ours: the combined use of both approaches can provide the developers with very high confidence in the quality of their product, as suggested also by [8], which discusses the need for both static and dynamic conflict detection and resolution.

D. Aspect oriented techniques

A related topic is that of interactions between aspect-oriented scenarios. A scenario is an actual or expected execution trace of a system under development. The work described in [18] is similar to ours, in so far as they place it in the phase of requirements analysis, propose a lightweight semantic interpretation of model elements. The technique relies on a set of annotations for each aspect domain, together with a model of how annotations from different domains influence each other. The latter allows the automatic analysis of inter-domain interactions. It is likely that, if feature and aspect orientation are combined in the same development, the two techniques could be integrated.

E. Formal methods

A recent trend of design-time conflict detection exploits the current advances in formal static analysis by theorem proving and model checking. The need for experimentation along this line has been recognized by Layouni et al. in [19], where they exploit the model checker Alloy [20] for automated conflict detection. In [21], we show how to express APPEL [22] policies in UML state machines, and exploit the UMC [23] model checker to detect conflicts. In [24], we automate the translation from APPEL to the UMC input language, and address the discovery and handling of conflicts arising from deployment-within the same parallel application-of independently developed management policies.

A feature interaction detection method close to model checking is presented in [25]: a model of the features is built using finite state automata, and the properties to be satisfied are expressed in the temporal logic Lustre. The environment of the feature is described in terms of the (logical) properties it guarantees, and a simulation of its behaviour is randomly generated by the Lutess tool; the advantage is that such an approach helps avoiding state explosion.

F. Abstract Interpretation

We remark a difference with the usual way of performing abstract interpretation [26], where the starting point is a detailed model, which is simplified, by abstracting away the information that is not needed for the intended analysis. What is proposed here is to start with an abstract view in terms of feature goals and resource accesses, and to perform conflict analysis and resolution up-front.

G. Interactions affecting performance

Recently, work has been done on detecting and resolving interactions that, thought not disrupting the behaviour, impact on the overall performance of the system. The approach described in [27] is based on a simple black box model: interactions are detected using direct performance measurements designed according to few heuristics. It would be interesting to assess whether our technique may supplement advantageously the heuristics to the point of balancing the cost of the required domain model.

V. CONCLUSIONS

We present a state based approach to the early detection, analysis and resolution of interactions in feature oriented software development. Starting with a light model of the state that the features abstractly share, the main steps of our approach are the generation of an interaction matrix, the assessment of each interaction (conflict or synergy), and the update of the model to resolve conflicts. The abstraction is such that only the mode (read or write) of an access to the shared state is considered; each access is characterized by its contribution to the overall goal of the feature it pertains to.

We provide a proof of concept of how interactions can be detected automatically, as well as of how the developers can get support in their assessment of the interactions and resolution of the conflicts, looking at the well known Smart Home domain.

An interesting development will be to evaluate whether to formalize the goal model, and how, in view of a (partial) automatic support to the developers' analysis tasks. Another line of development of the approach would be to supplement each resource in the shared space with a standard access protocol, to prevent conflicting interactions. Inspiration in this direction may come from well established practices, like access control schemes and concurrency control.

ACKNOWLEDGMENTS

The work was partly supported by the Italian MIUR PRIN project "Security Horizons".

REFERENCES

[1] G. Bruns, "Foundations for Features," in Feature Interactions in Telecommunications and Software Systems VIII, S. Reiff-Marganiec and M. Ryan, Eds. IOS Press (Amsterdam), June 2005, pp. 3–11.

[2] S. Apel, J. M. Atlee, L. Baresi, and P. Zave, "Feature interactions: The next generation (dagstuhl seminar 14281)," vol. 4, no. 7. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2014, pp. 1–24, URL: http://drops.dagstuhl.de/opus/volltexte/2014/4783/ [retrieved: Feb, 2015].

[3] A. Nhlabatsi, R. Laney, and B. Nuseibeh, "Feature interaction: the security threat from within software systems," Progress in Informatics, no. 5, 2008, pp. 75–89.

[4] V. Editors, "Feature Interactions in Software and Communication Systems," ser. Int. Conference series.

[5] D. Coetzee, A. Bhaskar, and G. Necula, "A model and framework for reliable build systems," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2012-27 arxiv.org/pdf/1203.2704.pdf, Feb 2012, URL: http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-27.html [retrieved: Feb, 2015].

[6] M. Shehata, A. Eberlein, and A. Fapojuwo, "Using semi-formal methods for detecting interactions among smart homes policies," Science of Computer Programming, vol. 67, no. 2-3, 2007, pp. 125–161.

[7] D. O. Keck and P. J. Kuehn, "The feature and service interaction problem in telecommunications systems: A survey," IEEE Transactions on Software Engineering, vol. 24, no. 10, Oct. 1998, pp. 779–796.

[8] N. Dunlop, J. Indulska, and K. Raymond, "Methods for conflict resolution in policy-based management systems," in Enterprise Distributed Object Computing Conference. IEEE Computer Society, 2002, pp. 15–26.

[9] M. Calder, M. Kolberg, E. H. Magill, and S. Reiff-Marganiec, "Feature interaction: A critical review and considered forecast," Computer Networks, vol. 41, 2001, pp. 115–141.

[10] P. Asirelli, M. H. ter Beek, A. Fantechi, and S. Gnesi, "A compositional framework to derive product line behavioural descriptions," in 5th Int. Symp. on Leveraging Applications of Formal Methods, Verification and Validation, ser. LNCS, vol. 7609. Heraklion, Crete: Springer, 2012, pp. 146–161.

[11] S. Reiff-Marganiec and K. J. Turner, "Feature interaction in policies," Comput. Networks, vol. 45, no. 5, 2004, pp. 569–584.

[12] M. Shehata, A. Eberlein, and A. Fapojuwo, "A taxonomy for identifying requirement interactions in software systems," Computer Networks, vol. 51, no. 2, 2007, pp. 398–425.

[13] M. Nakamura, T. Kikuno, J. Hassine, and L. Logrippo, "Feature interaction filtering with use case maps at requirements stage," in [28], May 2000, pp. 163–178.

[14] B. E. M. Weiss, A. Oreshkin, "Method for detecting functional feature interactions of web services," Journal of Computer Systems Science and Engineering, vol. 21, no. 4, 2006, pp. 273–284.

[15] Q. Zhao, J. Huang, X. Chen, and G. Huang, "Feature interaction problems in web-based service composition," in Feature Interactions in Software and Communication System X, S. Reiff-Marganiec and M. Nakamura, Eds. IOS Press, 2009, pp. 234–241.

[16] M. Kolberg and E. Magill, "Managing feature interactions between distributed sip call control services," Computer Network, vol. 51, no. 2, Feb. 2007, pp. 536–557.

[17] F. Siewe, A. Cau, and H. Zedan, "A compositional framework for access control policies enforcement," in Proceedings of the 2003 ACM workshop on Formal Methods in Security Engineering. NY, NY, USA: ACM Press, 2003, pp. 32–42.

[18] G. Mussbacher, J. Whittle, and D. Amyot, "Modeling and detecting semantic-based interactions in aspect-oriented scenarios," Requirements Engineering, vol. 15, 2010, pp. 197–214.

[19] A. Layouni, L. Logrippo, and K. Turner, "Conflict detection in call control using first-order logic model checking," in Proc. 9th Int. Conf. on Feature Interactions in Software and Communications Systems, L. du Bousquet and J.-L. Richier, Eds. France: IMAG Laboratory, University of Grenoble, 2007, pp. 77–92.

[20] Alloy Community, URL: alloy.mit.edu/community/ [retrieved: Feb, 2015].

[21] M. ter Beek, S. Gnesi, C. Montangero, and L. Semini, "Detecting policy conflicts by model checking uml state machines," in Feature Interactions in Software and Communication Systems X, International Conference on Feature Interactions in Software and Communication Systems, ICFI 2009, 11-12 June, 2009, Lisbon, Portugal. IOS Press, 2009, pp. 59–74.

[22] K. J. Turner, S. Reiff-Marganiec, L. Blair, J. Pang, T. Gray, P. Perry, and J. Ireland, "Policy support for call control," Computer Standards and Interfaces, vol. 28, no. 6, 2006, pp. 635–649.

[23] M. ter Beek, A. Fantechi, S. Gnesi, and F. Mazzanti, "A state/event-based model-checking approach for the analysis of abstract system properties," Sci. Comput. Program., vol. 76, no. 2, 2011, pp. 119–135.

[24] M. Danelutto, P. Kilpatrick, C. Montangero, and L. Semini, "Model checking support for conflict resolution in multiple non-functional concern management," in Euro-Par 2011 Parallel Processing Workshop Proc., ser. LNCS, M. A. et al., Ed., vol. 7155. Bordeaux: Springer, 2012, pp. 128–138.

[25] L. du Bousquet, F. Ouabdesselam, J.-L. Richier, and NicolasZuanon, "Feature interaction detection using a synchronous approach and testing," Computer Networks, vol. 32, no. 4, 2000, pp. 419–431.

[26] P. Cousot and R. Cousot, "Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints," in Proc. 4^{th} ACM Symp. Principles of Programming Languages, 1977, pp. 238–252.

[27] N. Siegmund, S. S. Kolesnikov, C. Kästner, S. Apel, D. S. Batory, M. Rosenmüller, and G. Saake, "Predicting performance via automated feature-interaction detection," in 34th International Conference on Software Engineering, ICSE 2012, June 2-9, 2012, Zurich, Switzerland, 2012, pp. 167–177.

[28] M. Calder and E. Magill, Eds., Feature Interactions in Telecommunications and Software Systems VI. IOS Press (Amsterdam), May 2000.

Developing a Repository for Component-Based Energy-Efficient Software Development

Doohwan Kim Jang-Eui Hong
Dept. of Computer Science
Chungbuk National University
Cheongju, Rep. of Korea
email: dhkim@selab.cbnu.ac.kr, jehong@chungbuk.ac.kr

***Abstract*— Software components are one reusable asset which can contain other kinds of software assets like requirement specifications, design patterns, source codes, documents, and so on. It can be used for designing software architecture or implementing a software system as an element like a building block. Therefore, software can be developed easily and quickly by assembling those building blocks. Focused on this nature of component-based development, energy-efficient software development can also be achieved with reusable software components. In particular, low-energy software has become a critical component for embedded and mobile software systems. Therefore, we have to consider energy efficiency to develop embedded software when developing the software based on reusable components. This paper, firstly, proposes how to represent the energy characteristics of the components and how to select a component for energy-efficient software development. We developed a component repository, ECoReS to support the selection of low-energy software components.**

Keywords-reusable software assets; component repository; energy-efficient software; component selection.

I. Introduction

Software reuse has become one of the general processes to develop software systems because reuse can provide huge benefits of error prevention, cost and time reduction, and even quality improvement [1]. The representative paradigm based on the reuse approach is known as CBSD (Component-based Software Development). CBSD can effectively support embedded software development because this kind of software frequently includes the same functions as other embedded software, which are in the product family. Therefore, the software can be developed faster and more reliable than developing it from scratch [2].

To elevate the benefits of software reuse, a repository that manages reusable assets on an organizational level is required. The purpose of a software component repository is to support the reuse of the components that have been acquired at the organization level. The repository also has to provide the functions of component registration, component retrieval, and component selection to software engineers for systematic reuse [3]. Therefore, many component repository systems have been developed that focus on managing and retrieving components to satisfy the functional requirements of developing software. However, embedded software development has to consider not only the functional requirements, but also various non-functional requirements because of limited resources and operational environments [4]. Therefore, these limitations must be considered as one of the characteristics, also known as the quality factors, of the software through the entire development process [5][6][7]. Low-energy consumption, as one of the characteristics of embedded software, has become a very important quality factor in portable or mobile systems like smart phones, MP3 players, and tablet PCs, because those systems are powered from limited energy sources such as a battery. However, there are just a few studies on the development of component-based low-energy software.

The major profits of component-based energy analysis can be considered from two sides: the first one is the reduction of feedback costs by early-phase estimation of energy consumption, and the second is the provision of high reliability of the estimation result. The higher abstraction level tends to lead to less accurate or coarse-grained analysis results [8]. However, reusable components have their own developed code which is managed in a component repository. Because the components can be used as the computational units of software architectural components, they make possible architecture-level analysis for requirements verification, which is a high abstraction level of software. Moreover, the component code will improve the accuracy of the analysis result. Therefore, if there is any method to support the energy analysis based on components in embedded software development, we can take the two advantages which conflict with each other, i.e., early phase estimation and its high reliable result. For these reasons, a component repository supporting low-energy software can be considered as one of the important infrastructures in the CBSD paradigm. Therefore, we developed a component repository, named Energy-based Component Retrieval System (ECoReS), which manages software components with their energy characteristics. The ECoReS can support fast and efficient embedded software development from the perspective of low-energy consumption, by providing the component selection based on energy characteristics.

The rest of this paper is organized as follows: the analysis of related work is explained in Section 2. The strategies to develop an energy-considered component repository will be discussed in Section 3. Section 4 describes the implementation of our proposed component repository, ECoReS. Section 5 describes conclusions and future research.

II. Related work

Each components repository can have different features according to policies of organization, application domain, and engineering environments [9]. The existing repositories have been providing various retrieval methods to find appropriate components. Therefore, each repository has its own distinct features, and specific structure of the repository. There are many repositories that are successfully supporting component reuse in the CBSD process [10][11][12]. Among them, we investigated recently proposed component repositories.

Z. Hai-mei et al. [10] emphasized the importance of component compositions as the issue to be managed by the component library. The authors designed a component library information model to improve reuse rate of managed components. Schema of that model included basic properties, classification information, and composition information. The key information of this research was the composition to enhance the reuse rate of stored and managed components in the library. However, the information model of the component library did not consider any information related to non-functional properties.

The research of C. Li et al. [11] focused on semantic-based component retrieval. They proposed an ontology-based component repository. Therefore, a software engineer who wants to develop component software can easily find software component using the ontology. However, the ontology did not include the terms of non-functional properties of software.

X. Shoukun et al. [12] developed a component library based on a component specification language named UCDL (Universal Component Description Language). By using the UCDL, the library manages the component information with the categories of basic information, classification information, interface specification, and feedback mark. However, this component library does not support the information of non-functional properties of components either.

The above studies implemented their repositories with different structures, and they provided distinct functions to maximize the reuse of components. However, these repositories can cause mistakes in the selection of suitable software components when the software engineer has to consider one or more non-functional requirements. Because the missing non-functional properties can lead to the re-development of large parts of the software [5], the engineer must consider the properties at the first step of component selection. Therefore, absence of the information of those properties can lead to inappropriate selection of software components. This incorrect selection can involve large re-developing costs, or even critical failure of the software.

Even though these component repositories provided convenient functions and they are well designed to support the reuse of components, they should be able to manage and provide information about the non-functional properties of the software components. Because the energy efficiency of software has become one of the most important non-functional properties of embedded systems, the perspective of low-energy must be considered in component-based software development. However, the previous studies and their repositories did not provide the distinguishing functions to support energy characteristics of the managed components. Our component repository provides the distinguishing functions for managing the energy characteristics of software components and supporting energy-efficient component selection.

III. Strategies for Low-Energy Software Development

As mentioned above, the component repository has to support low-energy software development by providing the reusable components. To support systematic reuse, we consider and define some strategies for describing energy characteristics of components, and for selecting a suitable component from our component repository.

A. Desiarable Useage Overview

Like general component repositories, the ECoReS also provides general features for component management and reuse. Additionally, certain distinguishing features are also provided by the ECoReS to support energy-efficient component selection. These features are reflected in the functions for component registration and component searching. When an engineer of an organization enrolls a new component in the ECoReS, it requests additional information that is related to the energy characteristics of the component. This information will be referred to by the software engineer who has to develop energy-efficient software. The availability of searching for the component characteristics in the ECoReS is also a different feature compared to other existing repositories. Because the functional requirement must be satisfied first when selecting a reusable component, the ECoReS also provides this functional requirements-based search. After that search, the software engineer can get a set of components which have the same functionality. We refer to these as “candidate components”. Given the main purpose of the ECoReS, energy-considered component selection will be done at the next step. The ECoReS provides the energy-comparing feature for actual selection from candidate components. Figure 1 shows an overview of the ECoReS including these features.

There are four specialized features in our repository, which are represented by the colored boxes in Figure 1. We also designed a feature, the “Function-based Search”, to provide easier search than just a general one. However, this feature is not colored because it is not the main concern of this research. As shown in the figure, the features of “add Component”, “Delete Component” and “Edit Component” are provided to manage components. Among them, “Add Component” has a specialized function that specifies the energy information. After using the “Function-based Search,” other specialized features will be supported. The software engineer will select the candidate components that have the same functionality to compare their energy efficiency. If the sequential order of searching based on function requirements, selecting candidates, and comparing candidate components were changed, the selection works will be meaningless. The feature, "Select the most suitable component" means the selection of a component that satisfies both functional requirements and energy-efficiency.

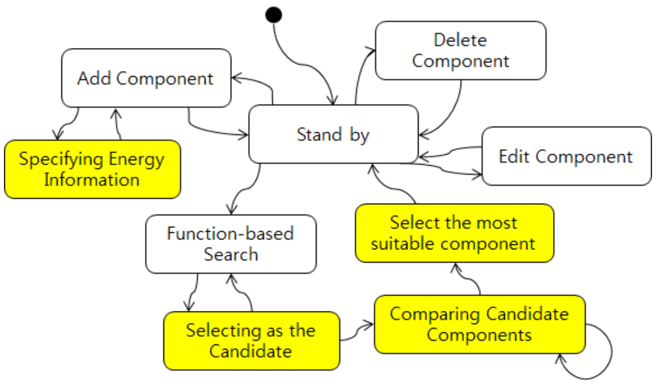

Add Component
Delete Component
Stand by
Edit Component
Specifying Energy Information
Function-based Search
Select the most suitable component
Selecting as the Candidate
Comparing Candidate Components

Figure 1. Usage overview for the ECoReS

B. *Manage Energy Characateristics by Interface*

A component can have several interfaces as the initiating entries of component behaviors. Those interfaces are the only way to request services of the components. Thus, those interfaces are identical to the basic units of the behavioral set that can be provided from components. This is a very critical notion that must be considered to manage the energy characteristics of reusable software components, because software never consumes energy by itself, but it consumes energy by controlling hardware devices when it is executed [13]. Therefore, the energy characteristics of components have to be defined by interface, and then have to be used to select components within the component repository.

C. *Consider the Effects of Interface Parameters*

Energy consumption of an interface can be changed by circumstances such as system conditions and given conditions of parameters. System conditions affect energy consumption on a small scale, while parameters can make a huge difference. For example, a specific parameter can be used as a flag variable which decides the branch of an inside interface. In other cases, some interfaces show exactly the same behaviors repeatedly when a parameter is given as a data stream or a similar one for iterative processing.

Because of these effects from the interference parameters, energy consumption of each interface can be variable. This is the reason why we have to define the energy characteristics of the components with the unit of interface. Moreover, the energy characteristics of components sometimes cannot be expressed as simple scalar values only, but certain ones have to be represented with a regression model. To define the energy model, the following recommendations are delineated.

- Thoroughly investigate the parameters of interfaces to determine if they can affect energy consumption, and how they may affect the energy consumption. The types of those effects are not the same for all possible cases. Therefore, this kind of investigation has to be done first to get an energy model.
- Collect sufficient data for the possible specific conditions of the interface parameters (for example, input data size, data value, etc.) to define an energy model. As mentioned above, the system condition is also a reason for the variance. Only a huge amount of sampling data can include this kind of variation. Therefore, we can define a proper energy model from a regression analysis using the large data set. This kind of model is already defined and used in the research of T. K. Tan et al. [8].

D. *Retrieve Components based on Facets*

It is difficult to find a suitable component from the repository that has many similar components functionally. In particular, if software developers would like to consider non-functional requirements as well as functional ones in component retrievals, finding a specific component that satisfies both requirements might be difficult.

Therefore, a component repository has to fulfill the needs of component retrieval with featured methods. In order to realize the methods, we provide a multi-dimensional facet-based retrieval technique to the ECoReS repository. Our repository provides two facets; the first is the domain facet which is the target domain of the software being developed, and the other is the functional facet which is responsible to the functionality to develop the software. The facets act like filters to show the list of components that could be selected. Although facet-based retrieval is not the main concern of this research, it is designed and implemented to help find a proper component by making a set of candidate components which have the same functionality before actual selection, focused on energy efficiency. Software engineers can limit the boundary of component retrieval by using these facets. Also, how many facets will be used to find a component can be determined with the trade-off analysis for an exact search and plentiful candidates.

E. *Compare Energy Characteristics*

With the facet-based retrieval, software engineers can find the components that are suitable for the functional requirements. However, there is still the remaining problem of selecting a proper component that is suitable for energy efficiency too. Because the result of facet-based retrieval can provide just a list of candidate components from the repository, software engineers have to compare the energy characteristics of the candidates.

Candidate components mean the components that are exchangeable with each other, according to Definition 1.

[Definition 1] Candidate Components: let CS be a set of components, and CC be a set of paired components. Then C_x and C_y are candidate components when satisfying;

$$CS = \{C_1, C_2, C_3, \ldots, C_n\}, \tag{1}$$
$$CC = \{C_x \in CS, C_y \in CS \mid Pre(C_x) = Pre(C_y) \wedge Post(C_x) = Post(C_y)\}, \tag{2}$$

Where, the Pre(Cx) means the pre-conditions (the input of the component) of Cx and the Post(Cx) means the post-conditions (the output of the component) of Cx. Also, $1 \le x \le n$, $1 \le y \le n$, and $x \ne y$.

The identification of candidate components means finding a set of components that satisfy the required functionalities. After that, the repository has to recommend a lower energy-consuming component among those candidate components, because the major purpose of our repository is selecting the most suitable component, satisfying not only functionality but also energy efficiency.

We considered this problem with the comparison of energy characteristics based on the interfaces, their energy models, and their parameters. Figure 2 shows the energy consuming patterns between two components when they are compared.

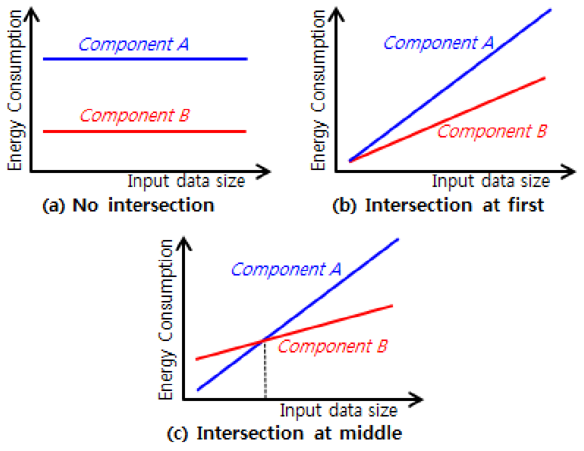


Figure 2. Energy consumption patterns between two components

Let us consider two components "Component A" and "Component B" that have the same function. However, they have different energy models for that function. For example, the energy model of the "Component A" is 2x+7, while the model of the "Component B" is 3x+2. As shown in the figure, the energy graphs of two components can be classified into three types like (a), (b) and (c), where all of energy models are linear regression models. In the case of (a) or (b), determining the more energy-efficient component between them is very easy. However, there is no component that is absolutely energy-efficient on the graph (c) in Figure 2.

In this case of the graph (c), "Component A" is more efficient in energy consumption until the input data size of A is less than the size of the crossing point. However, "Component B" is more efficient after exceeding the crossing point. The comparison of energy characteristics between candidate components from the perspectives of the energy models and input data size makes it possible to select a low-energy component, and also possible to develop component-based energy-efficient software.

IV. DESIGN AND IMPLEMENTION OF ECORES

Reusable software components must be managed and maintained on an organizational level by using a component repository or library [3]. The component repository provides several traditional functions such as the registration of components along with retrieval and selection of components to support the CBSD paradigm. The functions are basic and intrinsic for general component repositories.

However, our component repository, called ECoReS, provides not only the basic functions of general component repositories, but also the functions related to the energy efficiency of components. Those functions have to be considered and designed in a well-organized UI structure and seamless usage flow.

In this section, we explain how we considered the strategies for low-energy software development, what development environments were used to implement our component repository, and which functions are provided in our repository.

A. *Information for Component Specification*

Specifying a component to support easy reuse requires a lot of useful information for the components. However, there is a set of commonly required information such as the name, the usage, the list of interfaces, and the platform of a component for the specification [9][10][11][12]. This set of information is too general to examine further in component specification. Thus we only focus on the information that is required to describe the energy characteristics of components.

In the previous section, we discussed the energy models that describe the energy characteristics of component interfaces and interface parameters. However, some more information is required to describe the energy characteristics of components. This information is related to the platform in which the component is deployed. As mentioned above, software consumes energy by controlling the actions of hardware devices. Thus, the platform information must be covered in the component specification. These are the important platform specifications related to the energy characteristics:

1) *Hardware resources*: Hardware resources are actual energy consuming objects. Therefore, the information of the target (expected) platform of the component should be itemized in the specification. Among many of hardware resources, CPU clock speed and memory size are key information for energy characteristics [14][15].

2) *Operating System* (OS): Almost no application software can be activated without an operating system. The main purpose of OS is to control hardware resources (e.g., CPU, memory, etc.) and to provide system services to the application software like a middle layer broker. Therefore, the actual running environments of application software are controlled by the OS based on its scheduling policy, memory management policy, IO control, and so on. These policies can affect the energy consumption of software [15] and can be differently applied to its types (Android, IOS or something else) and versions.

3) *Compiler*: There are a number of available compilers. Even if the external behaviors of compiled codes (i.e., executable binary) for different source codes are the same, the compilers may have different optimization policies. An optimization policy can be differently applied during the code compilation process by setting different options even though the same compiler was used. Because the different optimization policy can generate different internal behaviors, the energy consumption of software is also different based on the different policies [16]. Therefore, we add the information of the compiler to specify the energy characteristics.

In addition to the energy models of component interfaces, the above information should be managed together. However, we can consider different tactics to manage that information because each part of the information represents different parts of a system. For example, the information of the platform for software is not changed during the lifetime of the software. Therefore, the influence of the platform on energy consumption will be decided at compile time. On the other hand, the information related to the real behaviors of the software can be decided at run time, i.e., which interface is called and how parameters are passed. The information can always change during real operation of the software depending on the requested user service.

We classified the information affecting the energy consumption into two types of factors: Indirect factors that are not changed during software operation, and direct factors that can be changed during the software operation. In the specification of component information, it is sufficient to describe the indirect factors of the component only once, since the factors have an equal effect every time on any interface and on any parameter. Unlike indirect factors, the direct factors (such as interfaces and their parameters) must be described multiple times in the specification, because they are differently affect to energy consumption based on which interface is called and how parameter is configured. TABLE I. shows these factors that affect energy consumption.

TABLE I. FACTORS AFFECTING ENERGY CONSUMPTION

Factors	Factor Types	Effecting Range
Hardware	Indirect factors	Whole component
OS		
Compiler		
Interface	Direct factors	Each interface
Parameter		

B. Development Environments

The ECoReS is developed to manage and retrieve software components with consideration of energy consumption. Ultimately, the goal of the ECoReS is to support component-based and energy-efficient software development. In the design of our repository, we separate it into DB-side and client-side because it can define N:M relationships.

Only one component repository is desirable in an organization to support various software projects because the centralized repository is easy to maintain and also easy to provide consistency for stored components, while multiple repositories are also valuable in distributed and collaborative development environments to elevate the flexibility and the variability of component-based development. The decision for the operational configuration of a component repository is dependent on the organization policy.

The client is developed by using JAVA with eclipse Rich Client Platform (RCP), Eclipse Modeling Framework (EMF), Java Data Base Connectivity (JDBC), as shown in TABLE II. Therefore, our repository system is possible to operate and use on any kinds of platforms.

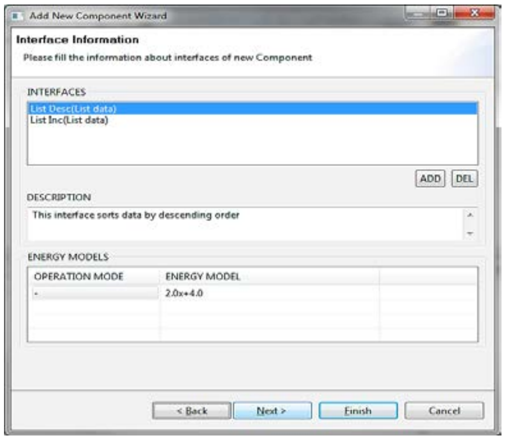
TABLE II. DEVELOPING ENVIRONMENTS OF ECORES

OS	MS Windows 7, 32bit
Language	JAVA(JDK 1.5)
Developing Tool	Eclipse 3.5(Galileo)
Platform	Eclipse Rich Client Platform
Plug-in	JDBC, EMF, etc.
DB	MySQL Server 5.5

C. Implementation of the Strategies

The common functional features of our repository are similar to other conventional component repository systems. However, the ECoReS has distinguishing features to support the strategies for energy-based component management, which are explained in Section III. This section is responsible for the implementation of those strategies.

The registration of components is a common and general function of component repositories except that the information of energy characteristics is also required. The component registration of the ECoReS provides a different widget to store the information about energy characteristics as shown in Figure 3.



Please fill the information about interfaces of new Component

INTERFACES
List Desc(List data)
List Inc(List data)

DESCRIPTION
This interface sorts data by descending order

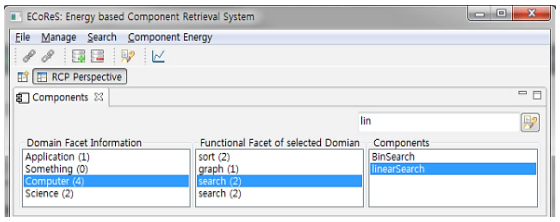
ENERGY MODELS

OPERATION MODE	ENERGY MODEL
-	2.0x+4.0

Figure 3. A Screen for energy characteristics of a component

The UI widget has the fields of “INTERFACES,” “DESCRIPTION,” and “ENERGY MODELS,” which should be filled in the “Add New Component Wizard” function. Because the indirect factors can be also identified as the commonly required information, we focused more on the energy model for each interface. Figure 3 shows an example of the "interface information" step of the component registration.

The facet-based retrieval is implemented with the composition of lists. Although the number of facets can be determined according to the domain hierarchy of the organizational business area, we define three facets such as domains, functions, and components level in our repository, as shown in Figure 4. This facet-based retrieval approach provides a quick and easy search to find a proper component in a functional manner and also helps software engineers think in top-down and systematic ways. Moreover, this approach can use the ontological concept to organize the facet structure.



Domain Facet Information	Functional Facet of selected Domian	Components
Application (1)	sort (2)	BinSearch
Something (0)	graph (1)	linearSearch
Computer (4)	search (2)	
Science (2)	search (2)	

Figure 4. A screen for facet-based component retrieval

After searching for candidate components that satisfy the functional requirements of the target software, then the software engineer can select a proper component based on energy efficiency. To compare energy efficiency, software engineers can simply set the check-box to compare the energy consumption with other components in the "properties" tab of a candidate component, as shown in Figure 5.

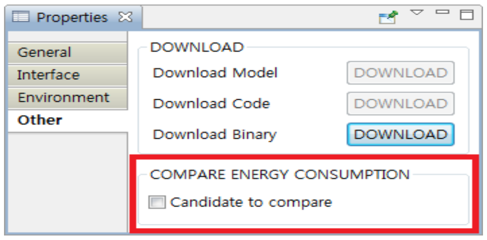


Figure 5. Selecting it as a candiate component

Comparing energy characteristics is the core feature of the ECoReS for selecting energy-efficient components. The comparison result of the energy characteristics between candidate components will be shown to help select the most suitable component for the software engineers. This result is shown in graph form to distinguish their energy efficiencies.

Selecting the specific interfaces of the components can be done when the comparison is activated. For example, there are two reusable components that are responsible for providing search algorithms. Even though the functions of two components are the same, their internal behaviors can differ. Therefore, the ECoReS will compare their energy efficiency based on component interfaces. Figure 6 shows the energy consumption graph of two components, "BinSearch" and "LinearSearch," which implemented a binary search algorithm and linear search algorithm, respectively.

In Figure 6, the "binSearch" consumes more energy than the "LinearSearch" at the first starting point. However, if the input parameter is bigger than 500 bytes, the "binSearch" is rather more energy-efficient than the "LinearSearch".

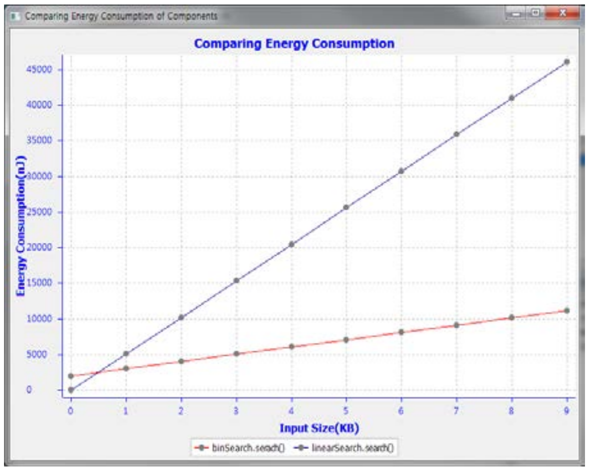


Figure 6. Energy consumption graph for two components

Due to this situation, software engineers who want to find an energy-efficient component have to consider the expected input data that was intended for processing by the component. In some cases in the above example, if we select to reuse the "binSearch" component when the input data size is always under 500 bytes, the selection will involve a worse result when the software is operated. The component always consumes more energy than the other component.

Therefore, we have to carefully predict and analyze the characteristics of the target system, to maximize the correctness of low-energy component selection.

V. CONCLUSION

CBSD has been broadly accepted as a reasonable and systematic paradigm to develop embedded software systems because embedded systems tend to be developed based on the product family approach [17]. The approach of reuse-based software development gives excellent benefits of time and cost reduction, and accuracy and reliability improvement as long as the approach applies well, and its infrastructure also runs well.

In this paper, we presented the design and implementation of a component repository, named ECoReS, which is a major infrastructure for the CBSD approach. Our repository specifically has focused on supporting component-based low-energy software development. Therefore, our proposed repository manages not only the general information of stored components, but also the energy characteristics of the components. It also provides functions like facets-based component retrieval, energy model management based on component interface, and energy consumption comparison.

We expect our component repository to be valuable in industrial and practical application development when a policy of energy-efficient software development is needed on an organizational level, or even on the subcontracting level.

In closing consideration, if any organization wants to combine or replace their existing component repository with the ECoReS, we expect that there will be three kinds of costs. For the first cost, the build or rebuild cost of repository will be needed. As mentioned, almost all component repositories have different structures. Even though the repository of the organization has a very similar structure with the ECoReS, minimum changes or re-builds for the structure are unavoidable. The second kind of cost deals with migration. After a total change of repository structure, migration must be followed. However, that migration can be omitted when the ECoReS is simply adapted to as-is system. The last cost is related to energy modeling. Because other repositories do not support the information about energy-efficiency, the energy modeling of the managed components should be done. Moreover, adaptation of the ECoReS without energy modeling is meaningless. We expect the cost related with energy modeling will be the largest of all.

For future studies, we are planning to upgrade the facet-based retrieval function with a powerful ontological scheme, and to establish the process and techniques for an architecture-based energy analysis framework, which cooperates with the ECoReS.

ACKNOWLEDGMENT

This research was supported by Basic Science Research Program through the NRF of Korea funded by the Ministry of Education (NRF-2014R1A1A4A01005566) and Next-Generation Information Computing Development Program through the NRF funded by the Ministry of Science, ICT & Future Planning (NRF-2014M3C4A7030503).

REFERENCES

[1] S. S. Yau, “Embedded Software in Real-time Pervasive Computing Environments,” in Proceedings of the 28th Annual International Computer Software and Applications Conference, pp. 406-407, 2004.

[2] X. Cai, M. R. Lyu, and K. Wong, “Component-Based Software Engineering: Technologies, Development Frameworks, and Quality Assurance Schemes,” in Proceedings of the 7th APSEC, pp. 372-379, 2000.

[3] J. Guo and Luqui, “A Survey of Software Reuse Repositories”, 7th IEEE International Conference and Workshop on the Engineering of Computer Based Systems, pp. 92-100, 2000.

[4] M. Daneva, M. Kassab, M. L. Ponisio, R. J. Wieringa, and O. Ormandjieva, “Exploiting a Goal-Decomposition Technique to Prioritize Non-functional Requirements,” In Proc. Of WER 2007, 10th International Workshop on Requirements Engineering, pp. 190-196, 2007.

[5] N. Siegmund, M. Kuhlemann, M. Pukall, and S. Apel, “Optimizing Non-functional Properties of Software Product Lines by means of Refactorings,” in Proc. Fourth International Workshop on Variability Modelling of Software-Intensive Systems (VaMoS'10), Vol. 37 (27-29 January 2010), pp. 115-122, 2010.

[6] M. Marzolla, “Simulation-based Performance Modeling of UML Software Architecture,” Ph.D Thesis, Ca’Foscari University, Italy, 2004.

[7] D. Kim, J. Kim and J. Hong, “A Power Consumption Analysis Technique Using UML-Based Design Models in Embedded Software Development”, Lecture Notes in Computer Science Volume 6543, pp. 320-331, 2011.

[8] T. K. Tan, A. Raghunathan, G. Lakshminarayana, and N. K. Jha, “High-Level Energy Macromodeling of Embedded Software”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 21, No. 9, pp. 1937-1050, Sep. 2002.

[9] G. Jones and R. Prieto-Diaz, “Building and Managing Software Libraries,” in Proc. on COMSAC 1988, pp. 228-236, 1998.

[10] Z. Hai-mei and G. Min, “A Component Library Information Model Supporting Component Composition”, in Porc, 2012 IEEE International Conference on Mechatronics and Automation, pp. 475-479, 2012.

[11] C. Li, X. Liu, and J. Kennedy, “Semantics-Based Component Repository: Current State Of Art and a CalCuation Rating Factor-based Framework,” in Proc. 32nd Annual IEEE International Computer Softare and Applications(COMSAC 2008), pp. 751-756, 2008.

[12] X. Shoukun, C. Xiaomei, and M. Zhenghua, “A Study of Local Component Library Based on UCDL,” in Proc. ICCSE ’09, pp. 904-907, 2009.

[13] K. Naik and D. S. L. Wei, "Software Implementation Strategies for Power-Conscious Systems", Mobile Networks and Applications, Vol. 6, Issue 3, pp. 291-305, 2001.

[14] C. L. Su, C. Y. Tsui, and A. M. Despain, “Low Power architecture design and compilation techniques for highperformance processors,” in Proceeding on IEEE COMPCON’04, pp. 489-498, 1994.

[15] D. Sarta, D. Trifone, and G. Ascia, “A Data Dependent Approach to Instruction Level Power Estimation,” IEEE Alessandro Volta Memorial Workshop on Low Power Design, pp. 182-190, 1999.

[16] M. E. A. Ibrahim, M Rupp, and S. E.-D. Habib, “Compiler-based optimizations impact on embedded software power consumption,” in Proceedings of the Conference NEWCAS, pp. 1-4, 2009.

[17] K. C. Kang, J. Lee, and P. Donohoe, "Feature-Oriented Product Line Engineering", IEEE Software, vol.19, no. 4, pp. 58-65, July/August 2002.

Automatic Generation of Sequence Diagrams and Updating Domain Model from Use Cases

Fabio Cardoso de Souza, Fernando Antonio de Castro Giorno
Master's Program in Software Engineering
Institute for Technological Research (IPT)
São Paulo, Brazil
e-mail: souzafc@yahoo.com, giorno@pucsp.br

***Abstract*—Software modeling allows for problem decomposition in a way that facilitates analysis and communication of the solution to developers and other interested parties. Models are widely used in engineering in general, but in Software Engineering modeling has often been left out due to the pressures to improve deadlines. A method and a tool that reduce the duration of this phase could help furthering the modeling phase. Use Cases are commonly utilized for functional specifications in Object-Oriented paradigm and the use of markups in Use Cases allow an automatic partial generation of Analysis Models, reducing the time of the modeling phase in this paradigm. This paper proposes a combination of rules for marking up Use Cases and one procedure for generating partial Sequence Diagrams with analysis classes (one Sequence Diagram for each Use Case) and the updating of the Domain Model with operations. A tool was built to prove the concept and two experiments were carried out.**

Keywords-Analysis Model; Use Case; Sequence Diagram; Model Driven Architecture.

I. INTRODUCTION

Software modeling permits the analyst to break the problem to be solved into parts which can be better analyzed. It also allows the formal communication of a functional and technical solution based on the demanded requirements. Model is a formal specification of the structure or function of a system [1]. A graphic representation can be used to provide a visual body for the model.

Despite being widely used in many areas of engineering, modeling has been left out in Software Engineering. According Rosenberg and Stephens [2], in practice, there never seems to be enough time to do modeling, analysis and design and there is always pressure from management to jump to code prematurely because progress on software projects tends to get measured by how much code exists, leading to problems in the quality of software.

Use Cases are commonly used for functional specification in Object-Oriented developments. According to Sommerville [3], Use Cases are an effective technique for eliciting requirements and they are increasingly used since the Unified Modeling Language (UML) became a standard for Object-Oriented modeling. Yet according to Rosenberg and Stephens [2], the Use Cases are created over a Domain Model since this offers the use of a common vocabulary. The utilization of markups in Use Cases can allow for the automatic partial generation of the Analysis Model, as demonstrated in the Mason and Supsrisupachai [20] work, where marked up Use Cases are automatic transformed into Sequence Diagrams.

The automatic partial generation could reduce the duration of the modeling phase, thus stimulating the adoption of this phase in Object-Oriented development projects, as suggested by a qualitative research [23] carried out with requirements analysts, system analysts and project managers. In this qualitative research, the majority of the interviewees agreed that software modeling improves the quality of the final product and most of them believe that the automatic generation of the partial Analysis Model can help the adoption of the modeling phase in software development projects. Due to space limitation, details of this research are omitted.

This paper presents a set of rules for marking up Use Cases and a transformation procedure that permits deriving Sequence Diagrams with analysis classes from the marked-up Use Cases. It also permits the updating of the Domain Model with operations identified in the Sequence Diagrams, leading to the Class Diagram. Class Diagram and Sequence Diagrams are the main diagrams in an Object-Oriented analysis model. The diagrams generated do not take into consideration details of a possible implementation, which must be done during the design phase. According to Booch et al. [4], the analysis must yield a statement of what the system does, not how. This research also presents a tool which implements the proposed procedure and with which the experiments were realized.

The rest of this paper is organized as follows. Section 2 presents concepts on which this research is based. This section also presents the State of the Art in the topics Model Driven Architecture and transformation of Use Cases into Sequence Diagrams. Section 3 presents a proposal for marking up Use Cases and a transformation procedure. Section 4 presents the tool and two experiments. The fifth and final section presents the conclusion and suggestions for future researches.

II. Concepts and State of the Art

This section starts presenting concepts related with Use Cases, Software Modeling and transforming requirements into software models approaches, and ends with state of the art on transformation subject.

A. *Requirements Specification with Use Cases*

Requirements Engineering provides appropriate mechanisms for [5]: understanding what the client wants; analysis of her/his needs; evaluation of feasibility; negotiation of a reasonable solution; specification of requirements in a unambiguous manner; validation of the specification and management of the requirements to be implemented.

Use Cases serves as functional specifications of requirements in Object-Oriented paradigm and the Analysis Model is created based on them. Use Cases provide the external behavior expected by the system with respect to the vocabulary in a Domain Model. Rosenberg [2] states that Use Cases describes a way by which the users interacts with the system and how the system responds. Pressman [5] notes that Use Cases does not tell how a system should realize the functionality. This emphasizes the importance of modeling.

According to Larman [6], Use Cases can be essential or concrete. Essential Use Cases do not consider mechanism details (like User Interfaces), while Concrete Use Cases consider them. In this paper, only Concrete Use Cases are contemplated.

Yet, according to Rosenberg [2], Use Cases should be written in the objects model context, referencing domain classes and boundary classes by their names. This recommendation is the base for this work as the objects constituents of the Analysis Model are the objects referenced in the Use Cases and existing in the Domain Model.

Use Cases makes explicit not only the objects involved in the system boundary but also the actors participating in the functionality and their actions. An actor is any entity that communicates with the system and is external to it, and may be a device, a system or a person. A main actor is that which interacts with the system in order to produce the result while secondary actors only support the system [5].

B. *Analysis Model*

Modeling is generally done in two levels of abstraction: Analysis Model and Design.

The Analysis Model - or Software Architectural Design - is used to identify, in a high level of abstraction, the components of the software, describing how the software is decomposed and organized into components [7]. In the case of Object-Oriented software, these components are Analysis Classes with their attributes and operations. In this paper, a partial Analysis Model is the expected result of the application of the proposed method.

The Class Diagram is the most important diagram of the Analysis Model and it describes the static vision of the system in terms of classes and relationships between them.

Jacobson [8] distinguishes the following types of classes used to give structure to Object-Oriented software: boundary, control and entity. According to him, boundary classes respond to information and behaviors related to system boundary; entity classes respond to information that are stored in the system and to behaviors surrounding these information; and control classes respond to behaviors which are not naturally incorporated into entities. These definitions are complemented by Bruegge and Dutoit [9], for whom, boundary classes represents interfaces between systems and actors, and control classes are in charge of realizing Use Cases.

The boundary and control classes, as well as their behaviors (their operations) are evident during analysis, in Analysis Model.

In this paper, these three types of objects are adopted in a way through which the Sequence Diagram can represent the software model with these three layers (boundary, control and entities).

Sequence Diagram is the second most important diagram of an Analysis Model and it is used to illustrate how objects interact with one another through messages, demonstrating the internal behavior of one system functionality (one Use Case).

The sequence of messages in a Sequence Diagram can use a pattern of communication between the objects, how, for example, the pattern presented by Heinemann and Denham [10], where messages should follow the flow "boundary ←→ control ←→ entity". This pattern is adopted in this work.

C. *MDD and MDA*

Model Driven Development (MDD) refers to the approaches based on models as the main products of a development [11]. According to Milicev [12], MDD raises the level of abstraction in a development.

Model Driven Architecture (MDA) is a MDD approach proposed by the Object Management Group (OMG) whose objective is to alleviate the problem of ruptures between design and code due to system migration from one platform to another [11].

MDA advocates four layers of model: Computation Independent Model (CIM), Platform Independent Model (PIM), Platform Specific Model (PSM) and Implementation Specific Model (ISM) as shown in Figure 1.

In the CIM layer there lies the process models and requirements that are independent of computing. In the PIM layer there lies the Analysis Model which is in the computing field, therefore totally independent of platform. In the PSM layer there lies the lower level models, which takes into consideration the platform where the system would be introduced. Finally, the ISM layer is the layer where the code is generated. The development focuses, on the MDA approach, is at a high level of abstraction, that is, in the CIM and PIM layers.

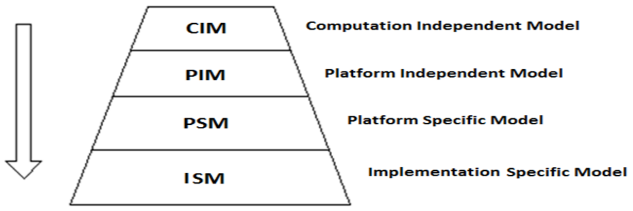

CIM
Computation Independent Model
PIM
Platform Independent Model
PSM
Platform Specific Model
ISM
Implementation Specific Model

Figure 1. Layers of MDA.

D. Related Work

In [13] a process for generating a model on the CIM layer from the requirements written in natural language was proposed. The requirements should be represented in Language Extended Lexicon (LEL) and in a scenario Model. LEL is a structure that permits representation of significant symbols in the universe of discourse, their synonyms and their behavior. The symbols can be: People, Objects, States, Events, among others. The process consists of a series of transformation rules over texts written in natural language contained in the LEL and in the scenarios.

In [14] a use case modeling approach was proposed in a way that elements of the Use Case are inserted into specific fields of a template, but there are no fields for components of the steps (sender object for example). Under this proposal, the steps should be restricted by a combination of grammar rules and rules for key words utilization. Based on this, the same authors [15] proposed a tool named aToucan (Automated Transformation of Use Case Model into Analysis Model). The tool aToucan reads the restricted steps of Use Cases and realizes the processing of natural language written in steps in order to obtain classes and relationships for the Analysis Model. The result is a generation of an intermediate Unified Modeling Language (UML) meta-model that is then transformed into a final Analysis Model. Only Class Models are mentioned in the obtained results.

In [16][17], it was proposed a set of marking-up rules and a set of syntactic structures in a manner an analyzer can extract the elements in order to generate a Sequence Diagram. The marking-up rules aim to permit the analyst to mark up occurrences of links, conditions and parallelism. The author names the marked-up Use Case with syntactic restrictions by Normalized Use Case.

The analyzer utilizes a dictionary to localize and store the elements in a catalogue applying syntactic rules. The catalogue is then used to obtain, in each message, the object sender, object receiver, operations and arguments that are registered in a file. There is no diagram generation. According to the author, the results needs to be refined by the analyst due to the confusion the analyzer can do while extracting concepts. The Use Case should be written in English natural language.

In [18] a set of transformation rules and a syntactic structure of the steps were also proposed. The steps should be written in this syntactic structure: “Who does What for Who”, being that the first ‘Who’ denotes the actor that starts the communication, the ‘What’ denotes the message to be transmitted, and the second ‘Who’ denotes the receiver of the message. The proposal contemplated a tool for editing Use Case and for generating the Sequence Diagram. The authors consider the method and the tool only as an instrument for learning.

In [19], it was proposed a tool for generation of Sequence Diagrams from Use Cases written in English. The tool uses a pre-existing component (Stanford Parser) to generate parts of speech tagged sentences and type dependencies. It then applies a proposed sentence structure rules and transformation rules to identify elements to generate the Sequence Diagram. The approach works only for the Simple Sentences in English.

In [20], Mason and Supsrisupachai proposed markups to indicate the primitives in a Use Case that derive elements to the respective Sequence Diagram. Only main scenarios of Use Cases are analyzed and each step of a Use Case needs to be marked up with an event type. A data dictionary is utilized as a reference of the Use Case elements. The marking up is made at each step of the Use Case on the elements: object sender, message, object receiver, actions and event timer. A tool was built for editing and marking up Use Cases and for generating the corresponding Sequence Diagram.

E. Consideration on State of the Art

In the MDA field, a lack of an official meta-model defined by the OMG for specification of Use Cases resulted in the presented proposals not fitting exactly into the MDA philosophy, which advocates, among other things, the use of UML and its meta-models as the origin and targets for transformations.

In direct Use Cases transformation into Sequence Diagrams, The Mason and Supsrisupachai [20] work offers a greater precision in the generation of a partial Analysis Model because the analyst previously identifies the elements in the Use Cases, as long as, he understands the problem and can deal with the imprecision of the natural language in an appropriate manner.

III. RULES FOR MARKING UP USE CASES AND THE TRANSFORMATION PROCEDURE

Mason and Supsrisupachai [20] work served as an inspiration for this proposal. As was previously mentioned, the use of markups in Use Cases is an efficient approach for partial automatic generation of Sequence Diagrams with analysis classes.

Some important differences in this work compared to the Mason and Supsrisupachai [20] work are:

This work proposes the updating of Domain Model with the operations identified during the method execution. It uses stereotypes to represent the types of classes according to their layers (boundary, control and entity). They also present a transformation procedure from Use Cases into Analysis Model and, finally, they define markups for actor, interface and guard condition. The set of markups was defined to allow, following the method, the generation of sequence diagrams considering stereotypes, actors (primary and secondary) and messages with condition guards. Even though this automatic generation is not enough for the analyst to start lower level design or code, it can be useful to the analyst since he does not need to start the modeling from scratch, thus reducing the duration of this phase. The mark-up process is considered to be made, by the analyst, against the Domain Model and trying to use, as much as possible, all the markups. For example, if in a specified step, the interface is not specified, and considering that there is a markup for interfaces, the professional must verify the possibility to explicit an interface in this step.

A. Marking up Rules

Table I below presents a set of markups proposed in this paper.

TABLE I. USE CASE MARKUPS

Markup	***Markup target***	***Markup format***
sdr	Sender object	[sdr Sender]
rcv	Receiver object	[rcv Receiver] or [rcv Receiver: name on Domain Model]
msg	Message	[msg Message] or [msg Message: label]
act	Internal action of the object (recursive message)	[act Message]
a1	Main actor	[a1 Actor]
a2	Secondary actor	[a2 Actor]
ifc	Human–machine or machine-machine interface	[ifc Interface]
grd	Guard condition	[grd condition]

The ‘msg’ markup allows an optional format with the use of a second argument (an optional label) which denotes that the label should be used on the diagram in the place of first argument (the event).

In the same way, “rcv” markup permits an optional format to specify the name of the receiver object when the name used in the step does not reflect the name in the Domain Model.

B. Transformation Process

According to Rosenberg and Stephens [2], Use Cases must be written in the context of the Domain Model, referencing the domain classes and boundary classes by their names. They recommend yet that the steps should be written with the structure: object – verb – object. Sequence Diagrams are behavioral models that illustrate how the objects interact with each other. These interactions are considered, initially (on the partial Sequence Diagram), a representation of the verbs specified on the Use Cases.

As mentioned above, the proposed procedure considers the types of object (boundary, control and entity), the actors and the messages between them with optional condition guard, in order to produce a partial Analysis Model. The following premises are considered in order to identify these elements in the Use Case text:

1. A Use Case step should contain only one message.
2. A step should be in one of the following configurations:
    - 2.1. “Xxx [a1 name] xxx [msg name] xxx [ifc name] xxx.”
    - 2.2. “Xxx [sdr System] xxx [msg name] xxx [ifc name] xxx.”
    - 2.3. “Xxx [sdr System] xxx [act name] xxx.”
    - 2.4. “Xxx [sdr System] xxx [msg name] xxx [a2 name].”
    - 2.5. “Xxx [sdr System] xxx [msg name] xxx [rcv name] xxx.”

Where ‘xxx’ represents free and non-obligatory texts and ‘name’ represents the name of an actor, object or message. A guard condition is optional and may occur in any of the above configurations.

Below is presented the procedure for transforming Use Cases into Sequence Diagrams and for updating the Domain Model with operations.

1. For each Use Case document:
    - 1.1. Is created a Sequence Diagram with the same name as the Use Case.
    - 1.2. Is added, into the diagram, the main actor, the «boundary» classes from ‘ifc’ markups without repetition, a «control» class with the same name as the Use Case, «entity» classes in the same sequence which they occur in the Use Case without repetition, and the secondary actors. «entity» classes are the other objects in Use Case which are neither actors, nor interface nor System.
    - 1.3. For each step in one expected configuration, the specific rules for messages creation must be observed. In any expected configuration, there may be a guard condition.
        - 1.3.1. “Xxx [a1 name] xxx [msg name] xxx [ifc name] xxx.”:
            - 1.3.1.1. One message is created from the main actor to the interface specified in the step.
            - 1.3.1.2. The focus is placed at the interface specified in the step in manner that the next message originates from it.
        - 1.3.2. “Xxx [sdr System] xxx [msg name] xxx [ifc name] xxx.”:
            - 1.3.2.1. If the control object has the focus:
                - 1.3.2.1.1. One message is created from the control object to the interface specified in the step.
                - 1.3.2.1.2. The focus is placed at the interface specified in the step in a way that the next message originates from it, unless there is a guard condition, because in this case, the guard condition may not occur, so the control object continues originating messages.
            - 1.3.2.2. If an interface has the focus:
                - 1.3.2.2.1. One message is created from the interface that has the focus to the interface specified in the

step. This represents a hyperlink from the first interface to the second, and such type of operation does not need to pass through the control object.

1.3.2.2.2. The focus is placed at the interface specified in the step in a manner that the next message originates from it, unless there is a guard condition, because in this case, the guard condition may not occur, so the first interface continues originating messages.

1.3.3. "Xxx [sdr System] xxx [act name] xxx.":

1.3.3.1. If the control object has the focus:

1.3.3.1.1. One recursive message is created in the control object.

1.3.3.1.2. The focus remains at the control object.

1.3.3.2. If an interface has the focus:

1.3.3.2.1. One message is created from the interface that has the focus to the control object.

1.3.3.2.2. One recursive message is created in the control object.

1.3.3.2.3. The focus is placed at the control object.

1.3.4. "Xxx [sdr System] xxx [msg name] xxx [a2 name] xxx.":

1.3.4.1. If the control object has the focus:

1.3.4.1.1. One message is created from the control object to the secondary actor.

1.3.4.1.2. The focus remains on the control object because it is not expected that a secondary actor can originate a message on the next step (secondary actors only supports the system and any activity that it can do is outside of the functionality scope).

1.3.4.2. If an interface has the focus:

1.3.4.2.1. One message is created from the interface that has the focus to the control object.

1.3.4.2.2. Other message is created from the control object to the secondary actor.

1.3.4.2.3. The focus is placed at the control object because it is not expected that a secondary actor can originate a message on the next step (secondary actors only support the system and any activity that it can do is outside of the functionality scope) and the control object originated the last message.

1.3.5. "Xxx [sdr System] xxx [msg name] xxx [rcv name] xxx.":

1.3.5.1. If the control object has the focus:

1.3.5.1.1. One message is created from the control object to the receiver object.

1.3.5.1.2. The focus remains on the control object because it is not expected that a receiver object (an entity by exclusion) can originates a message on the next step (Use Case do not explain the internal behavior of the functionality).

1.3.5.2. If an interface has the focus:

1.3.5.2.1. One message is created from the interface that has the focus to the control object.

1.3.5.2.2. Other message is created from the control object to the receiver object.

1.3.5.2.3. The focus is placed on the control object because it is not expected that a receiver object (an entity by exclusion) can originate a message on the next step (Use Case do not explain the internal behavior of the functionality) and the control object originated the last message.

2. The Domain Model is updated with operations identified in «entity» objects.

C. *Limitations*

The set of marking-up rules and the transformation procedure has the following limitations:

- Only main scenarios of Use Cases are considered;
- There is not treatment for inclusion and extension relationships at Use Cases;
- Only synchronous messages are considered;
- Message parameters are not considered;
- Loops and parallelism (concurrent processes) are not considered;
- Only Concrete Use Case is considered.

These limitations imply needs for adjustments and complements at the Analysis Model generated by the tool that implements the procedure, in order for the model to be useful for the next phases of the project (design phase and coding).

IV. Tool and Experiments

Below, we present the tool that implements the proposed procedure and two experiments.

A. *Tool*

A tool that automates the proposed procedure was developed using Java language and the Netbeans Integrated Development Environment (IDE). The tool is executed as a

Netbeans plug-in and it is presented as a tab on it, where the path to the Use Cases and Domain Model to be processed should be informed. Furthermore in this paper, file formats expected by the tool are presented. Figure 2 shows an overview of the transformation process.

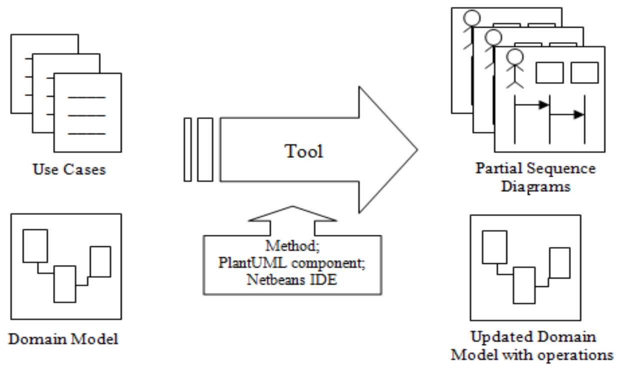

Use Cases
Tool
Partial Sequence Diagrams
Method; PlantUML component; Netbeans IDE
Domain Model
Updated Domain Model with operations

Figure 2. Transformation process overview.

Figure 3 presents an example of a marked-up part Use Case. For each generated diagram by the tool, a new tab is opened in the Netbeans IDE, containing the image of the diagram, as shown in Figures 4 and 5.

Use Case: Rent a Car.

Description: This Use Case describes the steps to make a car reservation on the Vehicle Rental web page.

Main Actor: [a1 Client]

Main Scenario:
1)**The** [a1 **Client**] [msg **request a reservation**: doReservation] **at the** [ifc **VehicleRentalPage**].
2)**The** [sdr **System**] [msg **request identification number**] **at the** [ifc **VehicleRentalPage**].
3)**The** [a1 **Client**] [msg **inform the identification number**: inputIdentificationNumber] **at the** [ifc **VehicleRentalPage**].
...

Figure 3. Example of a marked-up part Use Case.

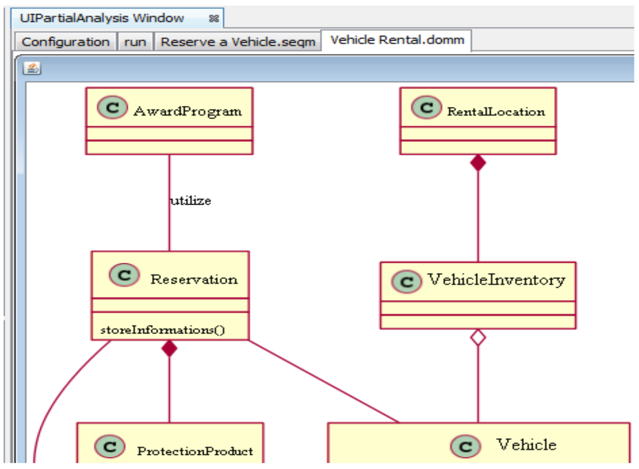


Figure 4. Class Diagram tab (partial view).

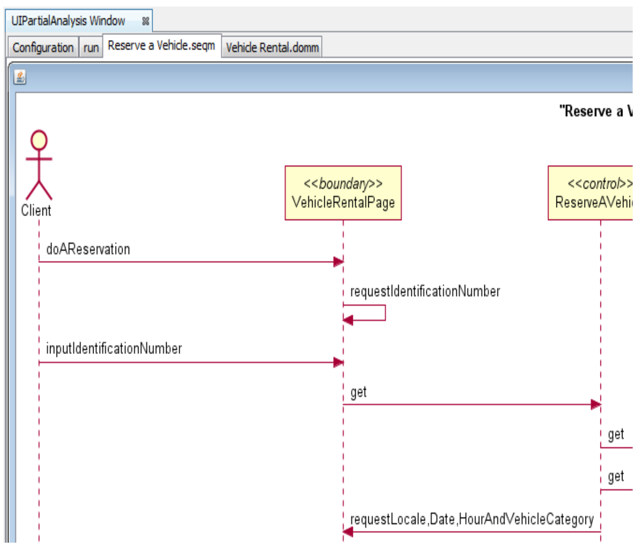


Figure 5. Sequence Diagram tab (partial view).

B. *PlantUML component and configurations*

The tool uses a pre-existing component known as PlantUML that permits generation of diagrams from stored commands in text formats. The tool creates PlantUML command files for each Sequence Diagrams to be generated from marked-up Use Cases and update with operations the PlantUML file related to the Domain Model. For this, the tool handles files with the following extensions:

- Files with “ucs” extension: File to be read and that contains a marked-up Use Case.
- File with “domm” extension: File to be read and updated and that contains PlantUML commands for Domain Model diagram generation.
- File with “seqm” extension: File to be created and that contains PlantUML commands for Sequence Diagram generation corresponding to Use Case with the same file name.

As soon as one file with PlantUML commands is generated or updated by the tool, immediately, the PlantUML component is activated for creating the respective diagram in the “png” format.

The tool, when in execution, alerts the analyst on cases of unidentified classes in the Domain Model, which however do not hinder generation of diagrams. The tool also alerts identified operations in Use Cases that already exists in the Domain Model. In this case, the tool does not include the operation in the Domain Model again.

The tool also transforms the names of objects in a manner by which words that compose it have their initials unified and transformed into capital letters. This pattern is known as “Camel Case”.

C. *Experiments*

The experiments were designed to verify if the application of the markups, associated with an automated method, could generate sequence diagrams with a reasonable margin of correctness, so that the adjustments to be made on the model, after its generation, would not cost more than it would if the Analysis Model was made from scratch.

For the experiments, one looked for materials containing Use Cases with the respective Sequence Diagram and Class Diagram or Domain Models. The first material is a tutorial [21] about analysis with Use Cases. The second material is a training example [22] about Analysis Model.

One problem found during the experiments is that the Use Cases of both materials did not explicitly specify the interfaces, and this would lead to a poor initial Analysis Model. To try and solve this problem, we define the interfaces in the steps during the markup process.

The evaluation of the results was done by looking for missing messages and objects in the diagrams generated by the tool/method (generated diagrams) compared to the diagrams presented in the materials (original diagrams). During the evaluation, other types of differences are detected, and they are listed in the Table II with their respective quantity of occurrences. One important difference that occurs in both experiments is that an operation, in the control object, gets an inadequate name when the focus is on an interface and a system realizes two or more subsequent steps. In this case, the name given to the operation, in the control object, is the name related to the first step of subsequent steps, and then, it does not reflect the meaning of all messages involved. This type of problem should be corrected after the diagram is generated, because the method/tool does not possess the mechanism to label, in an adequate manner and in this situation, control object operations.

One threat to external validity is about the skill of the analysts to specify the Use Cases considering the markups. This job must be done in a manner that the Use Cases represent, as complete as possible, all important objects and interfaces that must be present in the partial Sequence Diagrams. If this does not occur, the generated diagrams will be poor. We consider that it is an important concern and it can be mitigated by training the analysts on the markups elements orienting them to try to use the markups as much as possible on the Use Cases. A future research could evaluate this supposition accordingly.

Other threat to external validity is about the lack of Domain Model during the Use Cases specification, possibly leading to ambiguous objects while writing Use Cases and precluding part of the method (the update of the Domain Model with operations). We consider that the method need to have their use restricted to cases where de Domain Model is available during the requirements specification. One future research could evaluate the impact of the absence of the Domain Model during the Use Case specification using this method.

TABLE II. TYPES OF DIFFERENCES BETWEEN ORIGINAL DIAGRAMS AND GENERATED DIAGRAMS BY THE TOOL AND QUANTITY OF OCCURRENCES IN EXPERIMENTS

Type	*Description of difference*	*Exp.#1*	*Exp.#2*
1	The behavior allocation (the object where an operation is placed) in the original diagram is different from what was explicited in the step, and so, is different from the alocation in the generated diagram. This difference configures a modeling decision of the analyst and cannot be inferred by the tool, so needs to be adjusted in the generated diagram after generation.	3	0
2	There are behavior details in original diagrams that does not appear in the generated diagram. This difference is acceptable because is part of an analysis work to go beyond the interaction between the actor and the System and the generated diagram is only partial.	3	1
3	Non-utilization, in original diagrams, of the boundary-control-entity pattern. This difference is acceptable because the tool applies this flow pattern and the difference does not necessarily configure mistake.	1	1
4	Message omission in original diagrams. This difference is not a problem but a omission of the analyst in the original diagram.	0	1
5	Inadequate name of an operation in crontrol object. This occurs when the focus is on an interface and the System perform two or more operations. In this case, the first message will give the name of the operation in the control object, but it will not reflect the meaning of the operation that does more things than the first message suggest.	1	1

Considering only the types of differences that deserve adjustments (types 1 and 5), we have 5 differences in both generated diagrams compared to the original diagrams. Considering yet that the two generated diagrams have 30 messages, there is 83 percent of similarities between original and generated diagrams. Therefore, generated Analysis Model should be revised by the analyst after generation for behavior allocations and for operation's name on the control object. Beyond this, the generated Analysis Model should be complemented, given that the generation is only partial and because of the limitations of the method, cited above.

V. CONCLUSION

This paper presented a set of markups for Use Cases and a transformation procedure for automatic partial generation of an Analysis Model. The Mason and Supsrisupachai [20] work was the basis for this work once it defined some markups for primitives in a Use Case in order to originate a Sequence Diagram. This work defines some more markups (markups for guard-condition, actors and interface) and defines a procedure to create a partial Sequence Diagram with Analysis Classes, as well as complementing the Domain Model with the operations identified during the method execution.

The generated Analysis Model is composed of a partial Class Diagram and partial Sequence Diagrams (one per Use Case). The Class Diagram is the pre-existing Domain Model updated with the operations identified during the Sequence diagrams generation.

A tool was constructed based on the set of markups and the procedure in order to automate the generation of a partial Analysis Model.

The realized experiment demonstrated that the method/tool generates partial models with 83% of correctness, excluding differences that are not worth of adjustment. Considering this percentage, we believe that the implemented method could be used as a starting point for the Analysis Model since some improvements of the proposal can be made, as suggested below.

As a proposal to improve the tool, the model could be generated in XMI format, in a way that could be opened in a UML tool.

Another proposal to improve the tool is the creation of a tab for writing the marked-up Use Cases with an option for presenting texts with or without the markups, facilitating reading Use Cases when markups are hidden.

As a suggestion for future research, the procedure and the set of markups could consider alternative scenarios which will be transformed into fragments in the Sequence Diagrams. Extensions and Inclusions of Use Cases could also be considered.

Also, as a suggestion for future research, the transformation procedure and the set of markups could be extended to consider business rules written in Use Cases. A rule could be incorporated as an operation description when associated to a specific step or be incorporated as a note in the generated diagram when associated with Use Case as a whole.

REFERENCES

[1] J. T. Grose, G. D. Doney, and A. A. Brodsky, “Model Driven Architecture (MDA) and XMI,” in Mastering XMI. [S.l]: John Wiley & Sons, 2002, p. 329.

[2] D. Rosenberg and M. Stephens, “Introduction to ICONIX Process,” in Use Case Driven Object Modeling with UML: Theory and Practice. [S.l.]: Apress, 2007.

[3] I. Sommerville, Software Engineering, 9th ed.. [S.l.]: Addison-Wesley, 2011, p. 108.

[4] G. Booch, R. A. Maksimchuk, M. W. Engle, B. J. Young, J. Conallen and K. A. Houston, Object-Oriented Analysis and Design with Applications, 3th ed.. Boston, MA: Addison-Wesley, 2007, p.274.

[5] R. Pressman, “Requirements Engineering (RE) Tasks,” in Software Engineering: A Practitioner’s Approach, 6th ed.. [S.l.]: McGraw-Hill, 2004, p. 118.

[6] C. Larman, Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Iterative Development, 3th ed.. Upper Saddle River, NJ: Addison-Wesley, 2004, p. 145-146.

[7] IEEE. “Software Design,” in Guide to the Software Engineering Body of Knowledge. Los Alamitos, CA: [S.n.], 2004, p. 53.

[8] I. Jacobson, M. Christerson, P. Jonsson and G. Overgaard, Object-Oriented Software Engineering: A Use Case Driven Approach, 1st ed.. Edimburgh, UK: Addison-Wesley, 1992.

[9] B. Bruegge and A. H. Dutoit, “Analysis,” in Object-Oriented Software Engineering Using UML, Patterns and Java. Upper Saddle River, NJ: Pearson Prentice Hall, 2004, p. 177.

[10] G. Heineman and J. Denham, “Entity, Boudary, Control as Modularity Force Multiplier,” in Proc. 3rd Workshop on Assessment of Contemporary Modularization Techniques (ACoM.09), Orlando, FL, 2009, p. 42-47.

[11] L. Favre, Model Driven Architecture for Reverse Engineering Technologies: Strategic Directions and System Evolution, 1st ed.. Hershey, PA: IGI Global, 2010.

[12] D. Milicev, Model-Driven Development with Executable UML, 1st ed.. Indianapolis, IN: Wiley Publishing, 2009.

[13] N. Debnath, M. C. Leonard, M. V. Mauco, G. Montejano and D. Riesco, “Improving Model Driven Architecture with Requirements Models,” in Proc. 5th International Conference on Information Technology: New Generations (ITNG 2008), Las Vegas, NV, 2008, p. 21-26.

[14] T. Yue, L. C. Briand and Y. Labiche, “A Use Case Modeling Approache to Facilitate the Transition Towards Analysis Model: Concepts and Empirical Evaluation,” in Proc. Model Driven Engineering Languages and Systems (MoDELS 2009), Denver, CO, 2009, p. 484-498.

[15] T. Yue, L. C. Briand and Y. Labiche, “Automatically Deriving a UML Analysis Model from a Use Case Model,” Simula Research Laboratory, Oslo, Norway, Tech. Rep. 2010-15, Oct. 2010.

[16] L. Liwu, “A Semi-Automatic Approach to Translating Use Cases to Sequence Diagrams,” in Proc. Technology of Object-Oriented Languages and Systems (TOOLS’99), Nancy, France, 1999, p. 184-193.

[17] L. Liwu, “Translating Use Cases to Sequence Diagrams,” in Proc. 15th IEEE Int. Conf. on Automated Software Engineering (ASE’00), Grenoble, France, 2000, p. 293-296.

[18] L. Mendez, R. Romero and Y. P. Herrara, “UML Sequence Diagram Generator System from Use Case Description Using Natural Language,” in Proc. 4th Electronics, Robotics and Automotive Mechanics Conf. (CERMA’07), Cuernavaca, Mexico, 2007, p. 360-363.

[19] J. S. Thakur, A. Gupta, “Automatic Generation of Sequence Diagram from Use Case Specification,” in Proc. 7th India Software Engineering Conference (ISEC '14), 2014, Chennai, India.

[20] P. A. J. Mason and S. Supsrisupachai, “Paraphrasing use case descriptions and Sequence Diagrams: An approach with tool support,” in Proc. 6th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON 2009), 2009, Pataya, Thailand, p. 722-725.

[21] G. Evans. “Getting from use cases to code, Part-1: Use Case Analysis.” Internet: http://www.ibm.com/developerworks/rational/library/content/RationalEdge/jul04/TheRationalEdge_July2004.pdf, Jul. 13, 2004 [Feb. 22, 2015].

[22] J. White. “The Forgotten Step – Use Case Realization.” Internet: http://www.intertech.com/Blog/post/The-Forgotten-Step-Use-Case-Realization.aspx, Jan. 25, 2010 [Feb. 22, 2015].

[23] F. C. Souza. “Geração Automática de Diagramas de Sequência e Atualização do Modelo de Domínio a partir de Casos de Uso.” M.S. thesis, Institute for Technological Research, São Paulo, Brazil, 2011.

Framework for Developing Scientific Applications: Solving 1D and 2D Schrödinger Equation by using Discrete Variable Representation Method

Bojana Koteska and Anastas Mishev

Faculty of Computer Science and Engineering,
1000 Skopje, Republic of Macedonia
e-mails: {bojana.koteska,anastas.mishev}@finki.ukim.mk

Ljupco Pejov

Faculty of Natural Science and Mathematics,
1000 Skopje, Republic of Macedonia
e-mail: ljupcop@iunona.pmf.ukim.edu.mk

Abstract—**The absence of software engineering practices while developing scientific applications has negative impact on the quality of the applications. As a result, the probability for finding bugs in the application is higher, testing is more difficult and further code optimization and paralelization become an issue. In order to improve the developing process, in this paper, we propose a framework for developing scientific applications. The framework helps scientists to understand some of the basic concepts of software engineering and to change their current habits for developing scientific applications. Our goal is to adapt and modify some of the software engineering practices in every phase of the application development process. Aiming to use this framework in practice, we apply the recommendations for all phases while developing application for solving 1D and 2D Schrödinger equation by using the Discrete Variable Representation method (DVR). Using the framework resulted in better code organization, linked execution of the application modules for 1D and 2D equations, defining requirements and designing tests. As a final product we have an application organized in modules, documentation for each developing phase, comments in the code and executable tests.**

Keywords–Scientific application; Software Engineering; Framework; Schrödinger equation; Software quality.

I. INTRODUCTION

A scientific application is a software application that simulates activities from the real world and turns objects into mathematical models [1]. Scientific applications are designed to perform numerical simulations of natural phenomena in different scientific fields: computational chemistry and physics, informatics, mathematics, bioinformatics, etc. The execution of such applications that perform simulation of scientific experiments with large amount of data requires powerful supercomputers, high performance computing and Grid computing [2].

According to the Institute of Electrical and Electronics Engineers standard (IEEE Std) 610.12-1990, software engineering is defined as an application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software, that is, the application of engineering to software [3]. The main problem related to the development of scientific applications is the current development practice. In our previous paper [4], we conducted a survey among scientists - participants in the High Performance - South East Europe (HP-SEE) project and we found out that they do not use software engineering practices in the scientific application development process.

Scientific applications development differs from the development of commercial applications, especially in the stage of testing. Scientific applications are developed by the scientists themselves and used in the scientific research group which means that software can not be tested based on users' requirements, but the results can be compared to the results obtained from the real experiments or they are based on theory. High performance computing applications, such as scientific applications, can basically be the same with any kind of application, but only small number of additional changes in terms of planning, requirements elicitation, testing and development approaches are required [5]. It means that the same development practices cannot be fully applied, but there is a possibility to modify the practices and to make some adaptations.

The goal of this paper is to propose a development framework that will help scientists to change the current development practices and to show that software engineering can be included in the development process of scientific applications. The development framework provides basis for a complete scientific software development process and it also gives some quality recommendations that can be applied in the development process. The framework is generic which means that steps for developing can be used for different scientific applications. It provides a set of rules, recommendations and software engineering development practices. The main contribution of this paper is the description of the full development process of the scientific applications which tries to adapt and modify the existing software engineering practices for developing commercial applications.

The possibility of inclusion of the development practices proposed in the framework is validated with the development of an application for solving 1D and 2D Schrödinger equations by using the DVR numerical method. The application is developed by following the suggested development stages and recommendations which means that requirements are written, architecture is designed, code is organized in modules and optimized, testing is automated, etc.

The paper is organized as follows: related work is presented in the Section 2. Section 3 describes the current development practices used in scientific application development process. The framework is specified in the Section 4. The development process that includes the development stages proposed in the framework which are used for programming the application for solving 1D and 2D Schrödinger equations by using the DVR numerical method is given in Section 5. The conclusion and future work are provided in the Section 6.

II. Related Work

There are several papers that emphasize the need of software engineering when developing scientific applications, but no paper describes the full development process. In [6], the author presents the most modern software engineering practices relevant to scientific computing. He gives an overview of some software development models, choice of programming language, static analysis tools, dynamic testing procedures, version control systems, software quality and reliability. Here, there are some useful software engineering practices in the scientific applications development [7]: identification of resources required to develop the application; developing plan and schedule for completion of tasks and responsibilities; managing requirements, documenting them and constant control; making test plans and documentation; managing changes; managing the risks that may arise during development; constant quality control.

The quality of scientific applications can be improved by using generic programming, domain modeling and component based programming. For example, the creation of a meta model with established rules assists in the process of defining the application model needed to solve problems specific to that domain [8]. Software engineers can help scientists to realize the benefits of reusing by providing to them software frameworks. The framework will reduce the effort and time required for development, especially if scientists already have some knowledge about the used technologies, e.g., Message Passing Interface (MPI) [9]. Open Community Engagement Process is a model for software development which brings the software engineers and scientists together. The model define four steps: design, develop, refine, publish. This process is iterative and it follows incremental development approach and agile principles [10]. There is also a project for creating software infrastructure for scientific computing (ACTS) that provides a lot of free software tools which are divided into four categories: numerical calculations, code development, code execution and development libraries. These tools provide support for solving linear systems, optimization, obtaining analytical solutions, visualization, etc. [11]. Some practices that can improve the scientists' productivity and reliability of scientific applications are recommended in [12]. The survey we presented in [4], helped us to find some shortcomings in the development process of the scientific applications. Based on it, we proposed some practices for increasing the quality of the applications.

III. Scientists' development practices

Scientists usually learn programming independently or they are taught by other scientists. They believe that the process of software development is only the process of coding. The applications they are developing are intended for their scientific group or closer scientific community. The most important thing for scientists is to get scientifically correct results [9][13]. If the output results are correct, they are not very interested in making additional optimizations or parallizations. They are often guided by the thought that if a hypothesis is proven once, there is no need for reprogramming that section [5].

The scientists write codes in small teams without previous formal training [6]. The reason for the poor quality of high-performance computing applications lies in the lack of used software engineering formal methods and practices [9][14] . It may result in a larger number of errors, difficult understanding of the code written by the other scientists in the community, using additional unnecessary resources, problems with refactoring and optimizations later, etc. [13].

Scientists do not practice writing requirements or any kind of documents and they believe that requirements should only be discussed, but not written [13]. Later, when the testing is performed, there is no information what is done and what have to be done more. When the process of verification starts, scientists primarily give importance to the quality of the algorithm rather than its realization [9]. Usually the scientific problems have no precise solution and numerical methods are used to perform an approximation. Often, the verification means comparison with experimental results [6]. The software verification is based on professional judgment, such as comparison of the model output with other model outputs (benchmark) which will not always provide a consistent comparison and visual comparison between two pictures [15]. A primarily goal for scientists is to test the implemented theory, not the algorithm implementation. The testing is not the most important thing until an error that affect the correctness of scientific results appears [14].

The scientists in the HP-SEE project that participated in the survey [4] said that testing is a very important process, but mostly only the original developer is responsible for testing. Also, they are aware of their application errors, but they test manually, which means that they do not use any testing tools. Scientists think that writing testing documentation is very important process, but they describe the test cases freely. They answered that the biggest barrier for testing is time. Scientists are interested in graphs and/or reports that outline the state of system testing.

Taking into account all the problems mentioned above (difficult testing, no documentation, poor code comments, problems with optimization, etc.), we decided to propose a framework that will guide the scientists through the development process of the scientific applications.

IV. Framework for Developing Scientific Applications

This section describes the stages for developing scientific applications. We propose the incremental software development model which is elaborated in Software Engineering: 9th edition by Ian Sommerville [16]. We made small modifications of the model because there are two main differences between standard software development and scientific applications: The results from the scientific applications are verified by comparison to the results obtained from physical experiments; There are no customers that can evaluate the application since applications are developed only for the scientific community. We adapt this model by making changes in the process of evaluation, in test-driven design and short reports after each iteration.

This development process is characterized by fast delivery of increments and their evaluation which helps the cost of changing requirements to be reduced. Each increment of the development process contains 9 phases: **planning, requirements definition, system design, test cases design, coding,**

testing, evaluation, writing short report. It is shown in Figure 1.

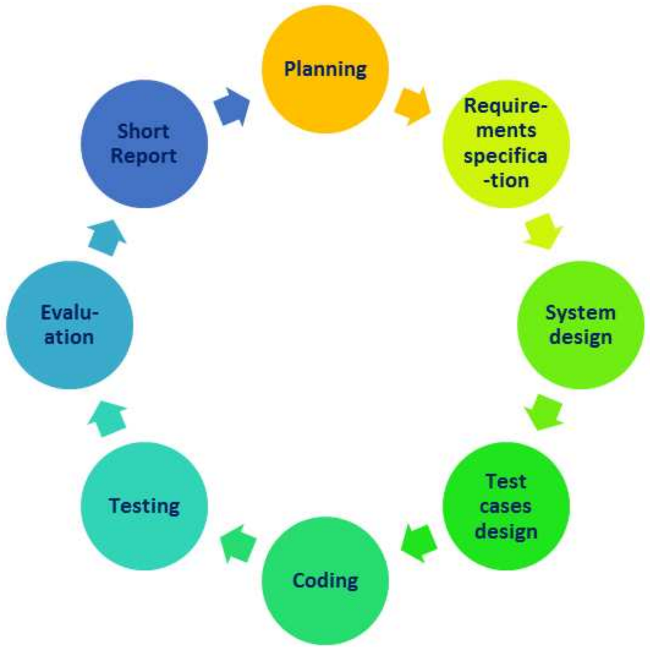


Figure 1. Scientific application development process

A. Planning

Each increment begins with the planning phase. The plan is a non-formal document which contains a list of activities that should be performed in the next iteration. For example, scientists should think about what parts (modules) have to be coded in that iteration, how they will be tested and evaluated. If there are more scientists who develop the application, the tasks should be assigned appropriately.

B. Requirements Specification

This is the development phase where requirements should be written formally. The document containing formal requirement specification is needed for better evidence of completed and future activities. We divide the requirements in two categories: functional requirements and nonfunctional requirements.

A **functional requirement** is a requirement that describes an application functionality (also inputs and outputs of a function). A **nonfunctional requirement** is a requirement that specifies the system behavior (constraints).

Each requirement should be consisted of the following fields: Id - unique requirement identifier; Name - short name that describes the requirement; Requirement type - functional or nonfunctional requirement; Version - current version of the requirement (as a number); Description - description of the functionality that needs to be realized (if functional) or description of tests and evaluation of the application functionalities or any hardware and software requirement; History of changes in each version of the requirement which is needed because of frequent requirements' changes, especially when solution goes in the wrong direction. The requirement specification will provide a better overview of the features that need to be tested, and less problems, such as delayed tasks realization, no working plan, late and incomplete testing, errors, etc.

C. System design

System design is the part where the hardware and software systems requirements are specified in order to establish a system architecture (for example, if the application needs multicore processor, Grid, high-performance computing to be run or some libraries that provide parallel execution). In this section, compilers, integrated development environments (IDEs), platforms and operating systems should be also specified.

D. Test cases design

Test cases designs are mostly related to functional requirements which means that well specified requirements can contribute to better design of test cases. Test cases need to be defined by using standardized forms. Also, boundary values and source code analysis could be considered as a relevant information. Test cases where the correct value cannot be specified should contain a range of values.

Each test case description should include the following fields: Id - a unique test case identifier; Name - name of the test case; Requirement id - specific id of the requirement tested with this test case; Goal - a description of the goals of the test case; Preconditions - conditions that must be met in order to perform the test case (results from other tests or some additional conditions); Execution environment of the test case; Expected results; Actual results; Test case status (pass or fail); History of the changes in each test case version. Code and branches coverage techniques could be useful for generating test cases.

E. Coding

The best approach is to define more independent modules because they can be used as generic functions in many other applications from the same or similar scientific domain. If any part of the code is repeated it definitely should be written as a separate function. The declaration and definition of unused variables should be avoided. Also, release of the memory should be performed always when possible. Comments must be written through the code and, if possible, some optimizations can be done. Parallelization can be performed by using some libraries (for example, OpenMPI [17], if the code is written in C).

F. Testing

When test cases are created and code is written it is time to make and run tests. There are various methods that can be used to provide assurance that the software is error free. White-box testing includes tests designed to check the source code. If possible, tests should be created before the software is developed. Only non redundant test cases should be selected, the other should be eliminated. The following criteria are good indicators that testing process is completed: all tests are performed successfully without errors, the criteria for source code coverage and test models are satisfied and validation by analytic solutions is achieved.

Tests automation and tools that provide source code testing or functional testing are very important and can greatly improve and speed up the process of testing. To improve the quality of applications, the current practice of manual testing should be changed. There are many frameworks for creating and running tests (for example, frameworks for C code: Check, CUnit, AceUnit, CuTest, etc.).

We recommend white box testing for each module, and then integration testing to check the interconnection between the modules and functionality of a system as a complete product. The possible conflicts with the already created tests must be resolved by changing the tests.

G. Evaluation

The results can be evaluated only by a limited number of users (usually scientists themselves) because the accuracy of the results is often based on theory and experiments. Scientists who have developed applications that solve similar problems can help to find errors. Developing experience and learning techniques for guessing or past errors can help a lot in the test design. Found errors should be sorted by priority, for example, if the error affects the further development or it is a critical error, it should be marked as error with higher priority and corrected as soon as possible.

H. Writing Short Report

A short report is document that describes the finished tasks in the current iteration. The tasks done in each phase should be listed. This is very important when a new member joins the team because he/she will know what is done and what have to be done. Scientists do not have practice for generating documents, but creation of documents can help to further improving and upgrading the application.

V. Solving 1D and 2D Schorödinger Equations by using Discrete Variable Representation Method

Discrete Variable Representation methods are widely used in different scientific domains, such as chemical physics, molecular quantum dynamics, etc. DVR's are described as a representation whose basis functions are localized about discrete values of the variables. DVR's are also approximations of coordinate operators by their values at the DVR points which are assumed diagonal. DVR methods are acceptable for many problems because they simplify the calculation of the kinetic energy matrix elements of the Hamiltonian matrix and also potential matrix elements which are the value of the potential of the DVR. DVR's provide efficient numerical solutions to quantum dynamical problems. When the DVR's product in multi-dimensional systems is calculated, operation of the Hamiltonian on a vector is fast and the Hamiltonian matrix is sparse (with many 0's) [18].

Schrödinger equation is a partial differential equation that describes the dynamics of system at atomic and molecular level. A time independent equation is represented with the following formula:

$$H * \Psi = E * \Psi \tag{1}$$

where H is the Hamiltionian operator, Ψ is the wave function of the quantum system and E is the energy of the state Ψ. A time dependent Schrödinger equation has the form:

$$i * \hbar \frac{\partial \Psi}{\partial t} = H * \Psi \tag{2}$$

where H is the Hamiltionian operator, Ψ is the wave function of the quantum system, $\hbar$ is the Planck Constant divided by 2Π and $\frac{\partial}{\partial t}$ is a partial derivative with respect to time t.

We have to develop scientific application for solving 1D and 2D Schrödinger equation. We will organize the development process in increments. There are already some codes for solving 1D and 2D Schrödinger equation in Mathematica, but we want to use C language because our goal is to use this module as an independent part in more complex application written in C. Detailed descriptions of the requirements and modules are given in the following subsection.

A. Increment 1

1) Planning:

- define the inputs and outputs of the module for solving 1D and 2D Schrödinger equation
- define developing environment, software and hardware needed for developing and execution of the application
- split the algorithm in modules
- define inputs and outputs for each module
- create tests for modules by choosing any tool for test automation
- develop all submodules needed for the module for solving 1D and 2D Schrödinger equation
- perform tests and evaluate the results
- correct errors if any

2) Requirement Specification: In this subsection, we will provide only the list the requirements with their IDs, but they also have to be specified as described above.

Functional requirements:

1) Module for multiplication of two 2D arrays (input - two 2D arrays of type double and number of rows/columns of the array (matrices are quadratic) of type integer, output - one 2D array of type double).
2) Module for making a diagonal 2D array(input - 1D array of type double and number of rows/columns of the array of type integer, output - diagonal 2D array of type double).
3) Module for multiplication of a scalar and a 2D array (input - 2D array, scalar and number of rows/columns of the array, output - 2D array).
4) Module for making an identity 2D array (input - the number of rows/columns of the array, output 2D array).
5) Module for addition of two 2D arrays (input-two 2D arrays and number of rows/columns of the array (matrices are quadratic), output - one 2D array).
6) Module for making a transposed 2D array (input - the number of rows/columns of the array, output 2D array).
7) Structure for a file row which contains three double numbers.
8) Structure for a file which contains array of elements of type "structure for a file row" and number of rows of the file (integer).
9) Module for reading data from file (input - char array(file path), output- element of type "structure for a file".

10) Module for sorting rows in file (input - element of the type ”structure for a file”, output - element of the type ”structure for a file”).
11) Module for calculating eigenvalues (input - 2D array of type double and number of rows/columns of the array(integer), output- 1D array of type double).
12) Module **thcheby** for calculating array of x-values, y-values(2D), transformation matrices for x and y values in finite basis representation (FBR) (input - number of points for x values- integer, number of points for y values - integer, minumim and maximum values for x and y-double, output - 1D array for x points of type double, 1D array for y points of type double, 1D array of x points in FBR of type double, 1D array of y points in FBR of type double, 2D array for transforming x points from FBR to DVR of type double, 2D array for transforming y points from FBR to DVR of type double. y-values are only needed for solving 2D Schrödinger equation.

Nonfunctional requirements:

1) Algorithm for array sorting should have complexity smaller then $O(n^2)$.
2) Application should be scalable (for example, when input data size increases, dynamic memory allocation must be used).
3) Memory used by the objects should be released when they are not used anymore.

3) System Design: The application for solving 1D and 2D Schrödinger equation can be run on a single processor machine. A C compiler is needed for a code compailing. A code editor should be installed also. The application can be run on any operating system. The results are provided to the standard output. In order to perform the testing, the CuTest framework for testing C codes should be installed also [19].

If the user wants to solve 1D or 2D Schrödinger equation, he/she has to provide only the input parameters to the module for solving these equations and file path.

4) Test Cases Design: In order to test the system for each specified functional requirement, we defined test case, as specified in the Section 4. Writing test cases before writing the code will help us to identify the needed input, predicted results and conditions. This is the part where only text document is written and later in the testing phase we will use the CuTest framework to write and run the unit tests. For example, the test case for the 12-th functional requirement (**thcheby** module) specified above, has the following conditions:

1) **deltax** is the difference between the maximum(**xmax**) and minumum value of x(**xmin**);
2) **deltay** is the difference between the maximum(**ymax**) and minumum value of y(**ymin**);
3) **nx1** is the number of x points increased by 1;
4) **nxy** is the number of y points increased by 1;
5) The i-th member of the array of x values (**ptsx**) has the value $((i+1)*deltax*1.0)/nx1+xmin$
6) The i-th member of the array of y values (**ptsy**) has the value $((i+1)*deltay*1.0)/ny1+ymin$
7) The i-th member of the array of x values in FBR (**fbrtx**) has the value $(((i+1)*\Pi)/deltax)^2$;
8) The i-th member of the array of y values in FBR (**fbrty**) has the value $(((i+1)*\Pi)/deltay)^2$;
9) The element at the position (i,j) of the transformation matrix for x values (**Tx**) has the following value $\sqrt{2.0/nx1}*\sin((i+1)*(j+1)*\Pi/nx1)$
10) The element at the position (i,j) of the transformation matrix for y values (**Ty**) has the following value $\sqrt{2.0/ny1}*\sin((i+1)*(j+1)*\Pi/ny1)$

For each condition a status should be written also (PASSED/FAILED).

5) Coding: The code is organized in modules. The coding of a module is performed after the specification of the test case for that module. Arrays are defined by using pointers and memory is released always when possible. The calculation of eigenvalues is performed by using the GNU Scientific Library(GSL) library. The complexity of the algorithm is $O(n^2)$. Sorting was implemented by using the quick sort method.

One of the most important things that scientists do not practice is writing comments through the code. We add comments for describing the modules, variables, cycles and statements. Another useful thing in programming is the concise naming of the variables and methods.

6) Testing: Testing was performed by using the CuTest system which is designed for writing, administering, and running unit tests in C [19]. A test case is passed if all conditions for that test case are satisfied. In order to check the correctness of the code, assertions must be added. For example, in order to check the correctness of the test case for the 14-th functional requirement, we have to write several test functions and to call them in the main function. In Figure 2, function for testing the members of the array of x values in FBR (**fbrtx**) is shown, where i-th member has the value $(((i+1)*\Pi)/deltax)^2$;, as described in the subsection test cases above.

```
void TestFbrtxMembers(CuTest *tc)
{
struct return_objects result=
thcheby(10, 1, 5, 10, 2, 6);
double *ac=result.fbrtx; //actual result
int i;
double *ex=
malloc(10*sizeof(double));//expected result
for(i=0;i<10;i++)
{
ex[i]=square(((i+1)*M_PI)/(5-1));
CuAssertTrue(tc,abs(ex[i]-ac[i])<0.00001);
}
```

Figure 2. Descriptive Caption Text

7) Evaluation: Evaluation was performed by comparing the results from our application to results from the provided code in Mathematica by the Upssala University. When comparing experimental results, another way to test this application to test the convergence of the results with increasing the density of grid points which is equivalent to increasing the number of basis functions. Several errors in the modules were found and they were corrected.

8) Writing Short Report: In the first increment, all modules specified in the requirements section needed for calculating were written and tested. The errors were corrected and all tests passed successfully. Also, all nonfunctional requirements were taken into consideration.

B. Increment 2

1) Planning:

- adapt the input for 1D and 2D equation
- integrate all modules into one module for solving 1D and 2D Schrödinger equation
- create tests for the module for solving 1D and 2D Schrödinger equation
- perform tests and evaluate the results
- correct errors, if any

2) Requirement Specification: **Functional Requirements**

- Module for making the interpolation function (**PES**) and energy list (**pel**)
- Module for calculating 1D and 2D Schrödinger equation: Input - Potential energy values computed on 2D grid of points (E, x, y) or 1D grid (E,x) read from a separate file, the number of DVR points, minimum and maximum values of x(1D) and x, y(2D), interpolation function and mass; Output - Frequencies of various vibrational transitions $(n, m)- > (l, k)$ to be compared with experiment, but also the convergence with respect to computation-related parameters to be tested, - Vibrational wavefunctions on 2D grid of points (ψ, x, y) or the square-modulus of ψ on 2D grid of points $(|\psi|^2, x, y)$, i.e. $((\psi * \psi), x, y)$.

3) System Design: Same as specified in Increment 1.

4) Test Cases Design: Two test cases were created. The conditions included in the test case for the first module in this increment are for interpolation function which should be implemented by using the Hermite interpolation method and some details about making the pel array. The second test case only checks the output results because all modules that are called in this module are checked in the previous increment. The only important thing here is the proper modules integration.

5) Coding: In this increment, only the developed modules were called in proper order in the module for calculating 1D and 2D Schrödinger equation. The module for calculating the 2D Schrödinger equation was developed. Also, this module is appropriate for solving 1D Schrödinger equation by eliminating the calculations which include the second coordinate y. The most convincing part here was programming of the interpolation method which is an integrated function in Mathematica.

6) Testing: Testing was also made by using the CuTest framework. Two tests were created and run.

7) Evaluation: The results were compared to the results from the program written in Mathematica.

8) Writing Short Report: Time needed for this increment was shorter because all modules were tested and programmed in the previous increment. An algorithm for Hermite interpolation was programmed and some errors that were found in the integration function were corrected.

VI. Conclusion and Future Work

In this paper, we proposed a framework for developing scientific applications. The framework describes the phases of the development process. In order to prove the developing model, an application for calculating 1D and 2D Schrödinger equation was developed. The results show that the framework is suitable for developing this application because at the end we have understandable code organized in modules, documentation and generated tests which are shortly described in the Section 5. The existing solutions for this problem do not have comments through the code or any documentation and it is very hard to understand the code. There are no proposed developing stages for scientific applications which will guide scientists to write documents and to plan the development process. This framework will help scientists to change the current development practices and will provide better overview of the application for the scientists who will join the project later. The framework is a guide for developing scientific applications because it explains the developing steps in details. The independent modules or complete program as a module can be used also in other scientific applications. Although there are many scientific libraries, usually a scientific research group needs modules with specific input, output and parallel programming methodologies which are originally developed by the group. Our goal is to test the framework of the large scientific applications and to check if this development process can be applied for different scientific applications. Also, we want to provide a set of documented modules written in C that are used in many different scientific applications which will be available for the scientists to use in their applications. Our future work is oriented to developing more complex high performance computing (HPC) scientific application which will require powerful distributed computing resources. If needed, some modifications of the existing model will be made in the future.

References

[1] PCMag. Definition of scientific application. [Online]. Available: http://www.pcmag.com/encyclopedia/term/50872/scientific-application [retrieved: Mar., 2015]

[2] C. Vecchiola, S. Pandey, and R. Buyya, "High-performance cloud computing: A view of scientific applications," in Pervasive Systems, Algorithms, and Networks (ISPAN), 2009 10th International Symposium on, Dec 2009, pp. 4–16. [Online]. Available: http://dx.doi.org/10.1109/I-SPAN.2009.150

[3] IEEE standard glossary of software engineering terminology. The Institute of Electrical and Electronics Ehgineers, New York, NY, USA. [Online]. Available: http://www.idi.ntnu.no/grupper/su/publ/ese/ieee-se-glossary-610.12-1990.pdf. [retrieved: Mar., 2015]

[4] B. Koteska and A. Mishev, "Software engineering practices and principles to increase quality of scientific applications," in ICT Innovations 2012, ser. Advances in Intelligent Systems and Computing, S. Markovski and M. Gusev, Eds., vol. 207. Springer Berlin Heidelberg, 2013, pp. 245–254. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-37169-1_24

[5] R. Baxter, "Software engineering is software engineering," in Proceedings of the First International Workshop on Software Engineering for High Performance Computing System Application. Edinburgh, Scotland, United Kingdom: IEE, 2004, pp. 14–18. [Online]. Available: http://dx.doi.org/10.1049/ic:20040411

[6] C. Roy, "Practical software engineering strategies for scientific computing," in Proceedings of the 19th AIAA Computational Fluid Dynamics Conference. Red Hook, NY, USA: Curran Associates, Inc, 2009, pp. 1473–1485. [Online]. Available: http://dx.doi.org/10.2514/6.2009-3997

[7] D. E. Post and R. P. Kendall, "Software project management and quality engineering practices for complex, coupled multiphysics, massively parallel computational simulations: Lessons learned from asci," International Journal of High Performance Computing Applications, vol. 18, no. 4, 2004, pp. 399–416. [Online]. Available: http://dx.doi.org/10.1177/1094342004048534

[8] F. Hernández, P. Bangalore, and K. Reilly, "Automating the development of scientific applications using domain-specific modeling," in Proceedings of the second international workshop on Software engineering for high performance computing system applications. New York, NY, USA: ACM, 2005, pp. 50–54. [Online]. Available: http://dx.doi.org/10.1145/1145319.1145334

[9] V. R. Basili et al., "Understanding the high performance computing community: A software engineer's perspective," IEEE Software, vol. 25, no. 4, 2008, pp. 29–36. [Online]. Available: http://dx.doi.org/10.1109/MS.2008.103

[10] L. Christopherson, R. Idaszak, and S. Ahalt. Developing Scientific Software through the Open Community Engagement Process . [Online]. Available: http://dx.doi.org/10.6084/m9.figshare.790723 [retrieved: Mar., 2015]

[11] O. Marques and T. Drummond, "Building a software infrastructure for computational science applications: lessons and solutions," in Proceedings of the second international workshop on Software engineering for high performance computing system applications. New York, NY, USA: ACM, 2005, pp. 40–44. [Online]. Available: http://dx.doi.org/10.1145/1145319.1145332

[12] G. Wilson et al. Best Practices for Scientific Computing. [Online]. Available: http://arxiv.org/abs/1210.0530 [retrieved: Mar., 2015]

[13] J. Segal. Models of scientific software development. [Online]. Available: http://oro.open.ac.uk/17673/1/SegalICSE08R.pdf [retrieved: Mar., 2015]

[14] ——, "Scientists and software engineers: A tale of two cultures," in Proceedings of the Psychology of Programming Interest Group. UK: University of Lancaster, 2008, pp. 44–51. [Online]. Available: http://www.ppig.org/papers/20th-segal.pdf

[15] R. Sanders and D. Kelly, "The Challenge of Testing Scientific Software," in Proceedings of the Conference for the Association for Software Testing, July 2008, pp. 30–36. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.464.7432&rep=rep1&type=pdf

[16] I. Sommerville, Software Engineering, 9th ed. Harlow, England: Addison-Wesley, 2010.

[17] OpenMPI. Open MPI: Open source high performance computing. [Online]. Available: http://www.open-mpi.org/ [retrieved: Mar., 2015]

[18] J. C. Light and T. Carrington Jr, "Discrete-variable representations and their utilization," Advances in Chemical Physics, vol. 114, 2000, pp. 263–310. [Online]. Available: http://dx.doi.org/10.1002/9780470141731.ch4

[19] Cutest: C unit testing framework. [Online]. Available: http://cutest.sourceforge.net/ [retrieved: Mar., 2015]

Exploring Test Composition: Towards Reusability in Combinatorial Test Design

Anna Zamansky
University of Haifa
Haifa, Israel
Email: annazam@is.haifa.ac.il

Eitan Farchi
IBM Research
Haifa, Israel
Email: farchi@il.ibm.com

***Abstract*—Combinatorial test design (CTD) is an effective test planning technique that reveals faulty feature interaction in a given system. CTD takes a systematic approach to formally model the system to be tested, aiming to minimize the number of test cases while ensuring coverage of given conditions or interactions between parameters. Since the system model and its test space in real-life cases are usually enormous, the process of creation of new tests is very expensive. This naturally leads to the need for exploring ways in which reuse methodologies can be incorporated into CTD practices. In this paper, we extend the standard CTD framework to a reuse-oriented setting by incorporating the notion of test *composition*. This notion arises naturally in sequential testing scenarios, where the output of one test is used as the input of the next test. Based on the proposed framework, we propose a reuse-oriented reformulation for the CTD problem, with the *composability* of test plan being the main consideration.**

***Keywords*–combinatorial test design; pairwise testing**

I. Introduction

As software systems become increasingly complex, verifying their correctness is even more challenging. Formal verification approaches are highly sensitive to the size of complexity of software, and might require extremely expensive resources. Functional testing, on the other hand, is prone to omissions, as it always involves a selection of what to test from a potentially enormous space of scenarios, configurations or conditions that is typically exponential in nature.

The process of test planning refers to the design and selection of tests out of a test space aiming at reducing the risk of bugs while minimizing redundancy of tests. *Combinatorial Test Design* (CTD) ([1], [2]) is an effective test planning technique, in which the space to be tested, called a *combinatorial model*, is represented by a set of parameters, their respective values and restrictions on the value combinations [3]. The approach of CTD can be applied at different phases and scopes of testing, including end-to-end and system-level testing and feature-, service- and application program interface-level testing.

The standard general formulation of the CTD problem consists of finding a small (and ideally minimal) set of test cases (a subset of the space to be tested), which ensures coverage of given conditions, or interactions between variables (such as pairs, three-way, etc.) The most common special case of the CTD problem is pairwise testing, in which the interaction of every pair of parameters must be covered. This is justified by experiments showing that a test set that covers all possible pairs of parameter values can typically detect between 50 to 75 percent of the bugs in a program [4].

The process of building combinatorial models for CTD is a laborious and error-prone task, which involves finding a set of parameters and values that define the test space and correctly identifying all valid value combinations. One potential pitfall of this process is omissions, i.e., failing to include an important parameter, value or combination of parameter values in the test space. Another pitfall is failing to correctly define the restrictions so that they capture the intended cobminations. Providing support for the test space definition process is a crucial factor in a successful application of CTD techniques to a wide range of testing domains. IBM's tool FoCuS [5] aims to automatically assist the tester in the solution of CTD problems by constructing the test space efficiently, while considerably reducing the risk of omissions. In [3] recurring patterns in CTD models were studied. These patterns capture cases that often repeat in different CTD models of different types and from different domains. Solutions for how to translate them into parameters, values and restrictions already exist, and thus can fascilitate *model reuse*.

In this paper, we address another type of reuse in CTD, namely *test reuse*. Addressing this problem is important, as the test space in real-life cases is usually enormous, creating new tests is very expensive. In this work-in-progress we explore the idea of *reuse-oriented* CTD based on the notion of *composition of test plans*. We propose a formal framework in which composition of test plans can be defined. This leads to a natural notion of the design space, which is the inductive closure of test plans under composition. Based on the proposed framework, we propose a reuse-oriented reformulation for the CTD problem, with the *composability* of test plan being the main consideration.

II. The CTD Framework

CTD is a powerful technique for functional testing. The most well-known versions of this approach aim at testing all value combinations of a given size t of the system's parameters. This problem of finding a minimal set of tests that cover all combinations of size t can be reduced to the mathematically equivalent problem of finding a covering array of strength t [6].

In this paper, we take a more general approach along the lines of [7], where the CTD problem is considered as a subset selection problem. Below we provide a generalization of the basic definitions of the framework, in which we make a separation between the input and output parameters of a test plan. This will be instrumental in what follows for defining the notion of composition.

We assume a finite set of system parameters $\mathbf{Par} = \{\mathbf{A}_1, \ldots, \mathbf{A}_n\}$. We treat a parameter as a finite set of its possible values. For any value $a \in \mathbf{A}_i$, we say that a has type $\mathbf{A}_i$, denoted by $type(a) = \mathbf{A}_i$. Subsets of parameters are called p-collections and are denoted by $\mathcal{A}, \mathcal{B}, ...,$.
We start by defining the notion of an (unordered) test, which is basically a collection of values the combination of which needs to be tested:

Definition 1: Let $\mathcal{A} = \{\mathbf{A}_1, \ldots, \mathbf{A}_m\}$ be a p-collection.

1) A **selection** for $\mathcal{A}$ is an element $\mathcal{S} \in \mathbb{P}(\bigcup_1^m \mathbf{A}_i)$ such that for distinct $a, b \in \mathcal{S}$, $type(a) \neq type(b)$. Write $\mathsf{Selections}_{\mathcal{A}}$ for the set of all selections for $\mathcal{A}$.
2) Call a selection $\mathcal{S}$ for $\mathcal{A}$ an **unordered test** when for every $\mathbf{A} \in \mathcal{A}$, there is some $a \in \mathcal{S}$, such that $type(a) = \mathbf{A}$ (so that $|\mathcal{S}| = |\mathcal{A}|$). Write $\mathsf{Tests}_{\mathcal{A}}$ for the set of all unordered tests for A.

Example 1: $\mathcal{A} = \{\mathsf{FileOps}, \mathsf{PathName}, \mathsf{OS}\}$ be a p-collection (taken out of a larger superset of parameters), where

$$\mathsf{FileOps} = \{open, close, read, write\}$$

$$\mathsf{PathName} = \{relative, absolute\}$$

$$\mathsf{OS} = \{unix, windows\}$$

Then $\mathcal{S}_1 = \{open, relative, unix\}$ and $\mathcal{S}_2 = \{close, windows\}$ are selections for $\mathcal{A}$, while the former is also an unordered test.

Using sets of selections, we can easily capture various interaction modes, such as pairwise testing (as well as others):

Example 2: For a p-collection $\mathcal{A} = \{\mathbf{A_1}, \ldots, \mathbf{A_m}\}$, denote by $2Pair(\mathcal{A})$ the set of all selections for $\mathcal{A}$ of size 2.

Thus we think of tests just as sets, specifying which values interact with which. However, when actually running tests, other types of information become important. One of them is a separation between the *input* and *output* parameters, which defines possible orders of execution of tests as sequences of consecutive test runs. Another issue is the *order* of parameter appearance. Thus, when defining the notion of *executable test set*, i.e., those tests that can actually be run in the system, we incorporate the above types of information as well:

Definition 2: Let $\mathcal{A} = \{\mathbf{A}_1, \ldots, \mathbf{A}_m\}, \mathcal{B} = \{\mathbf{B}_1, \ldots, \mathbf{B}_k\}$ be p-collections.

- For $e = (a_1, \ldots, a_r) \in (\mathbf{A}_1 \times \ldots \times \mathbf{A}_m) \times (\mathbf{B}_1 \times \ldots \times \mathbf{B}_k)$ (where $r = m + k$), we define the collapse of e by $col(e) = \{a_1, \ldots, a_r\}$.
- An element $e \in (\mathbf{A}_1 \times \ldots \times \mathbf{A}_m) \times (\mathbf{B}_1 \times \ldots \times \mathbf{B}_k)$ is called an **ordered test** if $col(e)$ is an unordered test.
- An executable test set from $\mathcal{A}$ to $\mathcal{B}$, denoted by $\mathcal{E} : \mathcal{A} \to\ B$, is a set of ordered tests.

Definition 3: Let $\mathcal{A}, \mathcal{B}, \mathcal{C}$ be p-collections.

- A **pre-test plan** from $\mathcal{A}$ to $\mathcal{B}$, denoted by $P : \mathcal{A} \to \mathcal{B}$ is a triple $P = (\mathcal{E}, \mathbf{R}, \mathbf{T})$ where:
  - $\mathcal{E} : \mathcal{A} \to \mathcal{B}$ is an executable test set.
  - $\mathbf{R}$ is a set of selections for $\mathcal{A} \cup \mathcal{B}$ called *coverage requirements*,
  - $\mathbf{T}$ is a set of selections for $\mathcal{A} \cup \mathcal{B}$, such that $\mathbf{T} \subseteq col(\mathcal{E})$,

  A **test plan** from $\mathcal{A}$ to $\mathcal{B}$ is a pre-test plan $P = (\mathcal{E}, \mathbf{R}, \mathbf{T})$ which satisfies that for every $\mathcal{R} \in \mathbf{R}$, there is some $\mathcal{T} \in \mathbf{T}$, such that $\mathcal{R} \subseteq \mathcal{T}$ (in words: every coverage requirement $\mathcal{R}$ in $\mathbf{R}$ is 'covered' by some test $\mathcal{T}$ in $\mathbf{T}$.)

Example 3: Let $\mathcal{A}$ be the p-collection from Example 1. Let $\mathcal{B} = \{\mathsf{PackageSize}, \mathsf{AckRequired}\}$ be a p-collection, where

$$\mathsf{PackageSize} = \{1KB, 2KB, 3KB\}$$

$$\mathsf{AckRequired} = \{yes, no\}$$

Let $\mathcal{C} = \{\mathsf{Mode}, \mathsf{Protocol}\}$ be a p-collection, where

$$\mathsf{Mode} = \{on, off\}$$

$$\mathsf{Protocol} = \{UDP, TCP\}$$

We can define, e.g., the following test plans $P_1 : \mathcal{A} \to \mathcal{B}$ and $P_2 : \mathcal{B} \to \mathcal{C}$: $P_1 = (\mathcal{E}_1, \mathbf{R}_1, \mathbf{T}_1)$, where:

$$\mathcal{E}_1 = (\mathsf{FileOps} \times \mathsf{PathName} \times \mathsf{OS}) \times (\mathsf{PackageSize} \times \mathsf{AckRequired})$$

$$\mathbf{R}_1 = \{\{relative, unix, yes\}\}$$

$$\mathbf{T}_1 = \{\{open, relative, unix, 2KB, yes\}\}$$

$P_2 = (\mathcal{E}_2, \mathbf{R}_2, \mathbf{T}_2)$, where:

$$\mathcal{E}_2 = (\mathsf{PackageSize} \times \mathsf{AckRequired}) \times (\mathsf{Mode} \times \mathsf{Protocol})$$

$$\mathbf{R}_2 = \{\{yes, TCP\}\}$$

$$\mathbf{T}_2 = \{\{3KB, yes, on, TCP\}\}$$

We are now ready to state the standard formulation of the CTD problem in terms of our framework: *given the set of executable tests $\mathcal{E} : \mathcal{A} \to \mathcal{B}$ and coverage requirements $\mathbf{R}$, find a set of tests $\mathbf{T}$, such that $P = (\mathcal{E}, \mathbf{R}, \mathbf{T})$ is a valid test plan.* In the case of pairwise testing, e.g., we set $\mathbf{R} = 2Pair(\mathcal{A} \cup \mathcal{B})$, where the latter is taken from Example 2.

Many algorithms and tools exist for solving various instances of the CTD problem ([8]). They can be classified into three categories ([9]):

- *algebraic*: providing a solution by a mathematical construction. These approaches usually lead to optimal results, but are however highly inefficient in practice (see, e.g., [10]).
- *greedy*: applying some search heuristic to incrementally build the solution. An efficient algorithm of such kind based on Binary Decision Diagrams [11] is given in [7], other examples are [12], [13].
- *meta-heuristic*: applying genetic or other bio-inspired search techniques (see, e.g., [14], [15]).

III. Introducing Composition

The intuition behind composition of test plans is very simple. Suppose a user has designed tests plans for several parts of the system. Now, he may want to test several parts sequentially, running one after the other and using the output of one as input for the other. Instead of constructing tests from scratch, he can reuse existing tests for the system parts via *composition*. To define in precise terms the notion of composition of test plans, we start by defining a composition of each of their parts: executable test sets and selections (and so also of tests).

Definition 4: Let $\mathcal{A} = \{\mathbf{A}_1, \ldots, \mathbf{A}_m\}$ and $\mathcal{B} = \{\mathbf{B}_1, \ldots, \mathbf{B}_k\}$. Let $\mathcal{S}_A, \mathcal{S}_B$ be selections for $\mathcal{A}$ and $\mathcal{B}$ respectively. Let $\mathcal{E}_1 : \mathcal{A} \to \mathcal{B}$ and $\mathcal{E}_2 : \mathcal{B} \to \mathcal{C}$ be executable test plans.

- Define $\mathcal{E}_2 \circ \mathcal{E}_1 = \{(a, c) \mid \exists b \in \mathbf{B}_1 \times \ldots \times \mathbf{B}_k.(a, b) \in \mathcal{E}_1 \wedge (b, c) \in \mathcal{E}_2\}$.
- If for every $a \in \bigcup_{1 \le i \le m} \mathbf{A_i} \cup \bigcup_{1 \le i \le k} \mathbf{B_i}$ it holds that $type(a) \in \mathcal{A} \cap \mathcal{B}$ implies $a \in \mathcal{S}_A \cap \mathcal{S}_B$, we say that $\mathcal{S}_A \circ \mathcal{S}_B$ is well-defined and equals to $\mathcal{S}_A \cup \mathcal{S}_B \setminus \mathcal{S}_A \cap \mathcal{S}_B$.
- Define $S_A \circ S_B = \{\mathcal{S}_A \circ \mathcal{S}_B \mid \mathcal{S}_A \circ \mathcal{S}_B \text{ is well defined}, \mathcal{S}_A \in S_A, \mathcal{S}_B \in S_B\}$.

The above definition captures both compositions of coverage requirements and of tests (as both of them are sets of selections). Note also that composition of 2-pair coverage requirements leads to 2-pair coverage requirements:

Proposition 1: Let $\mathcal{A}, \mathcal{B}, \mathcal{C}$ be p-collections. Then

$$2Pair(\mathcal{A} \cup \mathcal{B}) \circ 2Pair(\mathcal{B} \cup \mathcal{C}) = 2Pair(\mathcal{A} \cup \mathcal{C})$$

We can now define the following natural notion of composition of test plans:

Definition 5: Let $P_1 = \langle \mathcal{E}_1, \mathbf{T_1}, \mathbf{R_1} \rangle : \mathcal{A} \to \mathcal{B}$ and $P_2 = \langle \mathcal{E}_2, \mathbf{T_2}, \mathbf{R_2} \rangle : \mathcal{B} \to \mathcal{C}$ be test plans. The composition of P_1 and P_2 is defined by the pre-test plan

$$P_2 \circ P_1 = \langle \mathcal{E}_2 \circ \mathcal{E}_1, \mathbf{T_2} \circ \mathbf{T_1}, \mathbf{R_2} \circ \mathbf{R_1} \rangle$$

To capture the set of all possible runs of sequences of tests for which the output of one test is used as the input of the next test, we can now define the notion of the design space, which contains pre-test plans:

Definition 6: Let $P_1 : \mathcal{A}_1 \to \mathcal{B}_1, \ldots, P_n : \mathcal{A}_n \to \mathcal{B}_n$ be test plans from A_i to B_i respectively. For $i \ge 0$, define the sets $\mathbf{DS}_i(P_1, \ldots, P_n)$ of pre-test plans as the smallest sets satisfying:

- $P_1, \ldots, P_n \in \mathbf{DS}_0(P_1, \ldots, P_n)$.
- If $P_i, P_j \in \mathbf{DS}_k(P_1, \ldots, P_n)$ and $\mathcal{B}_i = \mathcal{A}_j$, then $P_i \circ P_j \in \mathbf{DS}_{k+1}(P_1, \ldots, P_n)$.

We denote $\bigcup_{i \ge 0} \mathbf{DS}_i$ by $\mathbf{DS}$.

IV. Reuse-Oriented Test Design

Given existing test plans, e.g., for $\mathcal{A} \to \mathcal{B}, \mathcal{B} \to \mathcal{C}, \mathcal{B} \to \mathcal{D}$, one can compose them to obtain new test plans for $\mathcal{A} \to \mathcal{C}$ and $\mathcal{A} \to \mathcal{D}$. The challenge is, however, dealing with the fact that composing valid test plans does not necessarily lead to valid test plans. Indeed, the operation of composition does not preserve validity of test plans, as demonstrated by the following example:

Example 4: Consider again the test plans P_1 and P_2 from Example 3. Their composition is the pre-test plan $P_2 \circ P_1 = (\mathcal{E}, \mathbf{T}, \mathbf{R})$, where:

$$\mathcal{E} = (\mathsf{FileOps} \times \mathsf{PathName} \times \mathsf{OS}) \times (\mathsf{Mode} \times \mathsf{Protocol})$$

$$\mathbf{T} = \emptyset$$

$$\mathbf{R} = \{\{relative, unix, TCP\}\}$$

This pre-test plan is obviously not a test plan, as there is no test meeting the requirement in $\mathbf{R}$.

We now come to the key idea of reuse-oriented CTD problem based on composition: designing basic tests with the aim of effectively composing them at later stages when required.

Definition 7: Two test plans $P_1 = \langle \mathcal{E}_1, \mathbf{T_1}, \mathbf{R_1} \rangle : \mathcal{A} \to \mathcal{B}$ and $P_2 = \langle \mathcal{E}_2, \mathbf{T_2}, \mathbf{R_2} \rangle : \mathcal{B} \to \mathcal{C}$ are *composable* if $P_2 \circ P_1$ is a valid test plan.

It is easy to see that for every pair of executable test sets $\mathcal{E}_1, \mathcal{E}_2$ and coverage requirements for $\mathcal{A} \to \mathcal{B}$ and $\mathcal{B} \to \mathcal{C}$ respectively, there always exist valid test plans $P_1 = \langle \mathcal{E}_1, \mathbf{T_1}, \mathbf{R_1} \rangle : \mathcal{A} \to \mathcal{B}$ and $P_2 = \langle \mathcal{E}_2, \mathbf{T_2}, \mathbf{R_2} \rangle : \mathcal{B} \to \mathcal{C}$, which are composable.

The notion of composability can be naturally extended to any number of test plans, as well as to the inductive closure under composition $\mathbf{DS}$ of a given depth:

Definition 8: Let $P_1 : \mathcal{A}_1 \to \mathcal{B}_1, \ldots, P_n : \mathcal{A}_n \to \mathcal{B}_n$ be test plans from A_i to B_i respectively. The design space $\mathbf{DS}$ is composable up to depth k if for every $P \in \mathbf{DS}_i$ and $P' \in \mathbf{DS}_j$ for $i, j \le k$, such that $\mathcal{B}_i = \mathcal{A}_j$, $P_i \circ P_j$ is composable.

The notion of composability defined above gives rise to new interesting formulations of reuse-oriented versions of the CTD problem, such as the following: given executable test sets $\mathcal{E}_1, \mathcal{E}_2$ coverage requirements and $\mathbf{R_1}$ and $\mathbf{R_2}$ for $\mathcal{A} \to \mathcal{B}$ and $\mathcal{B} \to \mathcal{C}$ respectively, find $\mathbf{T_1}, \mathbf{T_2}$, such that $P_1 = \langle \mathcal{E}_1, \mathbf{T_1}, \mathbf{R_1} \rangle : \mathcal{A} \to \mathcal{B}$ and $P_2 = \langle \mathcal{E}_2, \mathbf{T_2}, \mathbf{R_2} \rangle : \mathcal{B} \to \mathcal{C}$ are *composable* test plans. This can also be extended to composability of a design space of a fixed depth in a natural way.

Similarly to the standard CTD problem, the above problems may be solved using appropriate algebraic, greedy or meta-heuristic algorithms.

V. Summary and Further Work

In this paper, we define a formal framework, in which we formulate an algorithmic problem of reuse-oriented CTD based on composition. Composition of tests naturally arises in sequential testing scenarios, where the output of one test is used as the input of the next test. An evaluation of our framework is the next natural step. It will be done by implementing a greedy algorithm for solving the composition-based CTD problem defined in this paper, by extending the algorithm of [7], based on Binary Decision Diagrams.

Natural operations on test plans other than composition can be further considered for a systematic test reuse (e.g., the standard $join$ operation from database theory). Another direction for future research is investigating which data structures are most suitable for providing efficient solutions to the type of algorithmic problems formulated in this paper.

REFERENCES

[1] C. Nie and H. Leung, "A survey of combinatorial testing," ACM Computing Surveys (CSUR), vol. 43, no. 2, 2011, p. 11.

[2] J. Zhang, Z. Zhang, and F. Ma, "Introduction to combinatorial testing," in Automatic Generation of Combinatorial Test Data. Springer, 2014, pp. 1–16.

[3] I. Segall, R. Tzoref-Brill, and A. Zlotnick, "Common patterns in combinatorial models," in Proceedings of the IEEE Fifth International Conference on Software Testing, Verification and Validation (ICST). IEEE, 2012, pp. 624–629.

[4] S. R. Dalal, A. Jain, N. Karunanithi, J. Leaton, C. M. Lott, G. C. Patton, and B. M. Horowitz, "Model-based testing in practice," in Proceedings of the 21st International Conference on Software engineering. ACM, 1999, pp. 285–294.

[5] http://researcher.watson.ibm.com/researcher/view_group.php?id=1871.

[6] A. Hartman and L. Raskin, "Problems and algorithms for covering arrays," Discrete Mathematics, vol. 284, no. 1, 2004, pp. 149–156.

[7] I. Segall, R. Tzoref-Brill, and E. Farchi, "Using binary decision diagrams for combinatorial test design," in Proceedings of the 2011 International Symposium on Software Testing and Analysis. ACM, 2011, pp. 254–264.

[8] M. Grindal, J. Offutt, and S. F. Andler, "Combination testing strategies: a survey," Software Testing, Verification and Reliability, vol. 15, no. 3, 2005, pp. 167–199.

[9] M. B. Cohen, M. B. Dwyer, and J. Shi, "Interaction testing of highly-configurable systems in the presence of constraints," in Proceedings of the 2007 International Symposium on Software testing and analysis. ACM, 2007, pp. 129–139.

[10] N. Kobayashi, T. Tsuchiya, and T. Kikuno, "Non-specification-based approaches to logic testing for software," Information and Software Technology, vol. 44, no. 2, 2002, pp. 113–121.

[11] S. B. Akers, "Binary decision diagrams," IEEE Transactions on Computers, vol. 100, no. 6, 1978, pp. 509–516.

[12] D. M. Cohen, S. R. Dalal, M. L. Fredman, and G. C. Patton, "The AETG system: An approach to testing based on combinatorial design," IEEE Transactions on Software Engineering, vol. 23, no. 7, 1997, pp. 437–444.

[13] K. Tai and Y. Lie, "A test generation strategy for pairwise testing," IEEE Transactions on Software Engineering, vol. 28, no. 1, 2002, pp. 109–111.

[14] M. B. Cohen, P. B. Gibbons, W. B. Mugridge, and C. J. Colbourn, "Constructing test suites for interaction testing," in Proceedings of the IEEE 25th International Conference on Software Engineering, 2003, pp. 38–48.

[15] K. J. Nurmela, "Upper bounds for covering arrays by tabu search," Discrete applied mathematics, vol. 138, no. 1, 2004, pp. 143–152.

Smart City Applications TestBed

Towards a service based TestBed for smart cities applications

Danilo Silva, Felipe Ferraz
CESAR – Recife Center for Advanced Studies and Systems
Recife, Brazil
e-mail: {dls, fsf}@cesar.org.br

Felipe Ferraz, Carlos Ferraz
Informatics Center
Federal University of Pernambuco
Recife, Brazil
e-mail: {fsf3, cagf}@cin.ufpe.br

***Abstract*—Cities are facing a new challenge related to their population; it is the first time in history that most part of human population is now living in metropolis. Within this scenario, a city needs to deploy new solutions, presenting systems that answers to demands related to Security, Health, Resources, Government, Education and other urban daily systems to its citizens. In order to keep the creation of such solutions, it is vital to present developers with means to validate their projects. Focusing on this situation, this paper proposes the creation of a configurable testbed, where web services represent different systems of a smart city that could be consumed by applications in order to validate its implementation and features.**

Keywords - Smart City; TestBed; Solutions; Platform;

I. INTRODUCTION

With the fast increase of global urban population, cities now face risks in an unprecedented scale. Air pollution, lacking transportation infrastructure, uncertain economic landscapes, violence and unemployment – to name a few of the referred risks- are issues that nowadays can be addressed through the proper use of information and communication technology (ICT) 0

Batty et al. [2] reinforce the notion that technology applied to the concept of smart cities can help building a future in which a city will be interconnected, and relations between citizen and services and services to services will create an environment with innovative possibilities. Dirks et al. [3] stress the importance of software solutions that help society interact more efficiently with essential systems of a city. The "systems" of the city, in the context used by the referred authors, are: water and energy supply, transportation, security and healthcare [4][5].

Despite the fact that smart city applications can help society interact more efficiently with the available systems in the city, a great obstacle faced by developers, when creating new smart city solutions, stem from the difficulty of validating and testing their solutions [6][7]. This paper proposes a solution as an environment that could help on the construction of smart city solutions, with services and a data generator.

This paper is presented as follows: Introduction describes the problem. Section II depicts about smart cities and their concepts; Section III presents differences between TestBed and simulators, also describes related works; Section IV explains the proposed solution; Section V has details of the proposed TestBed platform architecture, and finally; section VI, presents conclusions and future works.

II. SMART CITIES

Most concerns around the concept of smart cities arise because cities have become home of more than 50% of the population [3][4][6]. Even more, the cities being the home of the largest part of the population, they are the center of the modern economies [8].

Despite the many challenges that branch from the increase of urban population, the amount of people that now live in big urban centers is still rising [9][10]. In fact, since 2008, for the first time in human history, the bigger part of the global population is now living in big cities [11][13].

In a previous work [5], it was possible to represent a city as an operational center that has a set of main services, which are vital for the city maintenance. Those services are categorized in the following way: Education, Security, Transportation, Energy and Water Supply, Healthcare, Government.

Following the same line of development, a smart city is a representation of a system of systems. Such characteristic is due to the interoperability present in the systems that build the infrastructure and environment of the city. They tend to cooperate in a way that new solutions for the daily urban life are more possible and easier to follow.

III. TESTBED

A TestBed is a platform for experimentation of large projects [14]. It allows testing applications with difficulty in accessing inaccessible domains – such as the case with many smart city applications [15]. Since smart city applications depend on city data to operate, this ends up hindering many efforts to create new solutions. Often city data is not readily available, or, when they are available, they are not in a format that is easily consumable by the applications [14]-[16]. The objective of this paper is to create a testbed, represented by different web services that represent city systems that will aid developers to create and test smart city applications in different urban landscapes [14][17].

A. Simulators vs. Testbeds

A simulator represents a special case of testbed. The main difference is that simulators generate data in real time while testbeds do not – in this case, data is pre-loaded in the storage of the solution [7][15].

When a simulator is used as a data generator for a smart city application, the simulator will generate city data during runtime, which has the downside of unpredictability since subsequent tests will yield different results. Using a testbed has the advantage of allowing consistent ability of repeating the tests since the data it provides for the smart city application is not prone to change.

B. Related works

By relating tests environment with smart cities solutions, it is possible to list some projects, such as: SmartGridLab [17], I3ASensorBed [12] and SmartSantander [18]-[22]. The following section presents a brief explanation about some of those solutions:

SmartSantander - is a project emerged after the identification requirement for an experimental Internet of Things (IOT) platform, which occurred during a Real World Internet congress, in 2009 [20]. Based on those requirements, a proposal to create a testbed in Europe emerged. The objective was to support research and experimentation of architectures, technologies, services and applications for IOTs in the context of SmartCities [19].

I3ASESORBED - The purpose of I3ASENSORBED is to create an experimentation testbed for different types of demographics -not only for universities but also for anyone who intends to create applications or to improve communication protocols for Smart Cities [12].

NetworkedCITY– An initiative derived from inside the IAAC –"*Institute for Advanced Architeture of Catalonia*" that combines physical computation, data visualization, and real time computing through interoperable devices, applications and models [23].

LOG-a-TEC – Conceived in JoJožef Stefan Institute in Slovenia, the initiative is an IOT testbed for small cities in Europe. Based on wireless sensor technology, its focus is in infrastructure of energy management and services [24].

Testbed for smart technologies in London – Intel, along with Imperial College London and London College, with the intent of promoting sustainable and connected cities, started an initiative that would turn London into a testbed for smart technologies [25].

IV. PROPOSED SOLUTION

The result of this work is a Smart City Testbed. A solution composed by two main components, an API set and a data generator.

The API set is composed of systems commonly found on the environment of the cities. On the following list, we detail the main objective of each of those systems [5].

Education: It is composed by services responsible for managing classes, grades and available courses [3][18] those services can help citizens and the government deploy or pay more attention to specific areas of expertise.

Healthcare: Is composed by units of treatment, Medical History of the patients with focus in providing a unique point of medical registers and hospitals specialties [4][10].

Transportation: Promotes improvements in mobility in the urban environment. This system receives great attention in vehicle traffic research [7], and proposes mechanisms related to Traffic Light control and Traffic management.

Government: Allows more transparency in the way the City is managed, for example, how tax money is spent by its politics [11].

Resources: Provides a way that enables to control the expenses of the city resources, such as water and energy [17].

Security: Related to public safety issues such as reporting crime and tracking violent areas [16].

To choose and define the mentioned types, we followed the approach proposed in a previous work [5], and back on this proposal, we summarized the most common types of solutions for smart cities. In addition, to determine the main goal of each system on this paper, we summarized some expectations and comments of related works.

Finally, Table I presents a compilation of the systems mentioned before along with different areas and testbed Services.

TABLE I. TYPES OF SYSTEMS AND LIST OF SERVICES

System Type	Area/Focus	Testbed Services
Education	Unified Grades	Register Grades
		View School Record
		Register Absent Days
	Courses	Register Courses
		View Courses
Healthcare	Medical Records	New Entry
		View Medical History of the Patient
		Assess Service
	Hospitals	Add Hospital
		Search for special treatment units
		Rate Hospital
Transportation	Control Traffic Lights	Open/Close Traffic Lights
		Detect malfunctioning of Traffic Lights
	Traffic control	Inform Traffic Jams
		View Traffic
Government	Taxes	Add Expenses
		View Expense History
	Registering of Occurrences	Report Occurrence
		Check Occurrence
Resources	Consumption	Register Consumption
		Consumption per Individual
		Consumption per Region
	Interruption of Service	Turn Service Off/On
Security	Register Occurrences of Crime	Report Crime Occurrence
		View History of Crime Occurrences
	Violence level per Area	Add new Area
		Areas by Crime Occurrence
		Assign Police to Area

The model created for the testbed respects three different levels, the first one, already presented, is represented by systems; each system is divided accordingly to sub areas or focus groups. The area represents level two, where one can find some specific solutions. For instance, Educational System presents unified Grades and Available courses. Lastly, layer three presents services built in each area, for example, Educational System, which has a unified grade solution, that offers services related to: Register Grades, View School records, and register absent days.

V. ARCHITECTURE

In this section, the testbed architecture will be described with more details. In addition, the components that were used as well as the reason for their adoption will be explained.

A. *Application server*

Jetty [26] is the application server chosen because it presents the advantage of being easy to set and integrate in Eclipse IDE. Another advantage is the fact that it is possible to embed Jetty inside of the application (inside of a main class, opposed as being packaged in a .war file). The main class is responsible for initializing the application server and exporting RESTful endpoints (created using JAX-RS of Apache CXF project).

B. *Persistence Layer*

On the persistence layer, the Java Persistence API (JPA) [27] implemented by Hibernate ORM [28] was used. HSQLSDB [29] is the database management system used for the solution, it is the selected option because it allows memory execution along with our solution, creating a fast and flexible solution. It presents downsides related to its simplest implementation, however, it is possible to change the persistence layer and DataBase by changing a specific layer.

However, the manual utilization of the persistence framework generates a repetition of the code, which is potentially dangerous due to the lack of appropriate persistence session management and JDBC transactions (Java's database connectivity API).

To overcome this code generation issue, we adopted the Spring framework [30] due to its dependency injection, persistence session management and JDBC transactions. By using Spring it was possible to minimize the code generation, speed up the development, reduce the amount of errors, and raise the overall quality of the solution.

C. *Architecture Overview*

The way each individual component communicates with one another is depicted in Figure 1. As it can be seen, the proposed solution follows the 4+1 architecture [31] of Philippe Kruchten.

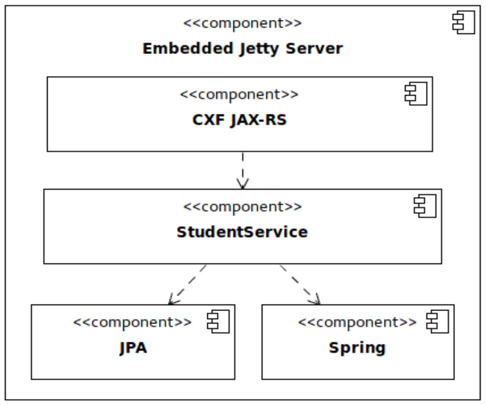


Figure 1. Architectural components

D. *Generation of Data*

As briefly discussed in this section, a mechanism to generate data was created. This mechanism allows the creation of various scenarios based on input parameters.

The parameters that the data generator accepts are the following:

- **City spot** - latitude and longitude: those values serve as a reference to geographical locations of the city.
- **City size:** values 1, 2, and 3 define, respectively, small, medium, and large cities. This parameter is used with the city spot to determine the size of the radius, which helps determining if a spot is part of the city.
- **Amount of inhabitants**: works as a parameter to determine how many inhabitants the city possesses.
- **Amount of accesses per inhabitant of urban systems:** a number that defines the amount of accesses that each system receives by the citizens of the city.
- **System rank index**: the system (Resources, Security, Education, Healthcare, etc.) receives values from 1 to 5 that ranks them from most to least accessed.

Table II serves as an example of input values, passed to the generator in order to create a base scenario.

TABLE II. VALUES USED TO CREATE A BASE SCENARIO

Latitude and Longitude	52,15 and 0,18
City size	1
Inhabitants	100
Amount of accesses	100
Resources, Security, Education, Healthcare, Transportation, and Government	1,2,3,1,2,3

In an early round of tests, Table II represents a city with a central point near Cambridge, and its inhabitants will perform 10.000 daily accesses to the services API.

To know how many accesses will be performed, one can sum up the "system rank" indexes of the systems. In the example previously presented, the result of the sum of indexes is 12. It means that, for instance, the Resources system will be responsible for 1/12 of the total amount of accesses, whereas Government will be responsible for 3/12.

Table III represents the total amount of accesses received by each service.

TABLE III. AMOUNT OF SIMULATED ACCESSES BY SYSTEM TYPE

Resources	834
Security	1.667
Education	2.500
Healthcare	834
Transportation	1.667
Government	2.500
Total	10.002

The creation of a data generator able to create realistic data was not the focus of this work. Because of that, it is likely to have a scenario where a student enrolls in two classes (courses) that happen at the same time. Likewise, it is possible to report crimes in the middle of the ocean.

VI. CONCLUSION

Based on research pertaining testbeds and smart cities, it is possible to notice that there is a great demand for testing smart city solutions, and that most of the existing testing strategies are based on the use of sensors and IOTs.

The advantage of the proposed solution is the possibility to create testing scenarios where validation of ideas and solutions for smart cities can be performed without having to rely on sensors and IOTs.

As future works from this research, it is intended to improve the data generator creating realistic scenarios and to develop a feature that enables the re-execution of previously created scenarios.

REFERENCES

[1] T. Nam and T. a. Pardo, “Conceptualizing smart city with dimensions of technology, people, and institutions,” Proc. 12th Annu. Int. Digit. Gov. Res. Conf. Digit. Gov. Innov. Challenging Times - dg.o ’11, 2011, p. 282.

[2] M. Batty et al., “Smart cities of the future,” Eur. Phys. J. Spec. Top., vol. 214, no. 1, Dec. 2012, pp. 481–518.

[3] S. Dirks, C. Gurdgiev, and M. Keeling, “Smarter cities for smarter growth: How cities can optimize their systems for the talent-based economy”, IBM Inst. Bus. Value, 2010. Available: http://public.dhe.ibm.com/common/ssi/ecm/en/gbe03348usen/GBE03 348USEN.PDF [Accessed: 15-Feb-2015].

[4] S. Dirks, M. Keeling, and J. Dencik, “How Smart is Your City?: Helping Cities Measure Progress,” IBM Inst. Bus. Value, 2009.

[5] F. Ferraz et al., “Towards a Smart City Security Model Exploring Smart Cities Elements Based on Nowadays Solutions.”, ICSEA 2013, The Eighth International Conference on Software Engineering Advances, pp. 546–550, 2013.

[6] W. M. da Silva, A. Alvaro, G. H. R. P. Tomas, R. a. Afonso, K. L. Dias, and V. C. Garcia, “Smart cities software architectures: a survey,” Proc. 28th Annu. ACM Symp. Appl. Comput. - SAC ’13, 2013, pp. 1722–1727.

[7] V. V. Filho, “RTSCUP : Testbed For Multiagent Systems Evaluation”, Thesis, Federal University of Pernambuco, 2008.

[8] J. M. Hernández-muñoz, J. B. Vercher, L. Muñoz, and J. A. Galache, “Smart Cities at the Forefront of the Future Internet”, 2011, pp. 447–462.

[9] S. Dirks and M. Keeling, “A vision of smarter cities: How cities can lead the way into a prosperous and sustainable future,” IBM Inst. Bus. Value. June, 2009.

[10] P. Hall, “Cities of tomorrow: An Intellectual History of Urban Planning and Design in the Twentieth Century”, Oxford: Blackwell Publishing, 3rd Edition, 2002.

[11] M. Naphade, G. Banavar, C. Harrison, J. Paraszczak, and R. Morris, “Smarter Cities and Their Innovation Challenges,” Computer (Long. Beach. Calif)., vol. 44, no. 6, Jun. 2011, pp. 32–39.

[12] T. Olivares, F. Royo, A. M. Ortiz, and I. Mines-telecom, “An Experimental Testbed for Smart Cities Applications”, Proceedings of the 11th ACM international symposium on Mobility management and wireless access, 2013, pp. 115–118.

[13] A. Caragliu, C. Del Bo, and P. Nijkamp, “Smart Cities in Europe,” J. Urban Technol., vol. 18, no. 2, Apr. 2011, pp. 65–82.

[14] E. Barreiros, A. Almeida, J. Saraiva, and S. Soares, “A Systematic Mapping Study on Software Engineering Testbeds,” 2011 Int. Symp. Empir. Softw. Eng. Meas., Sep. 2011, pp. 107–116.

[15] P. Hanks, S. Pollack, M. Cohen, “Benchmarks, Testbeds, Controlled Experimentation and the Design of Agent Architectures,” AI Mag. 13(4), 1993.

[16] A. Hahn, A. Ashok, S. Sridhar, and M. Govindarasu, “Cyber-physical security testbeds: architecture, application, and evaluation for smart grid,” IEEE Trans. Smart Grid, vol. 4, no. 2, 2013, pp. 847–855.

[17] G. Lu and D. D. W. Song, “SmartGridLab : A Laboratory-Based Smart Grid Testbed,” no. I, 2010, pp. 143–148.

[18] L. Sanchez, V. Gutierrez, J. A. Galache, P. Sotres, J. R. Santana, and J. Casanueva, “SmartSantander : Experimentation and Service Provision in the Smart City”, WPMC 2013 16th International Symposium on, 2013.

[19] “SmartSantander.” [Online]. Available: http://www.smartsantander.eu/. [Accessed: 01-Apr-2015].

[20] “SmartSantanderRA: Santander Augmented Reality Application.” [Online]. Available: http://www.smartsantander.eu/index.php/blog/item/174-smartsantanderra-santander-augmented-reality-application. [Accessed: 02-Apr-2015].

[21] [“Participatory Sensing Application.” [Online]. Available: http://www.smartsantander.eu/index.php/blog/item/181-participatory-sensing-application. [Accessed: 30-Mar-2015].

[22] “Smarter Travel.” [Online]. Available: http://www.smartsantander.eu/index.php/smart-travel. [Accessed: 01-Apr-2015].

[23] “NetworkedCITY.” [Online]. Available: http://www.iaac.net/smart/?portfolio=networkedcity. [Accessed: 01-Apr-2015].

[24] “LOG-a-TEX - A Smart City IoT testbed.” [Online]. Available: http://www.livingbitsandthings.com/in-the-spot-light/8-technology/45-log-a-tec-a-smart-city-iot-testbed#.UzhUeNxq5fM. [Accessed: 02-Apr-2015].

[25] “Intel turns London into testbed for smart technologies.” [Online]. Available: http://news.techworld.com/green-it/3359881/intel-turns-london-into-testbed-for-smart-technologies/. [Accessed: 30-Mar-2015].

[26] “Jetty.” [Online]. Available: http://www.eclipse.org/jetty/. [Accessed: 02-Apr-2015].

[27] “JPA - Java Persistence API.” [Online]. Available: http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html. [Accessed: 30-Mar-2015].

[28] “Hibernate ORM.” [Online]. Available: http://hibernate.org/orm/. [Accessed: 01-Apr-2015].

[29] “HSQLDB.” [Online]. Available: http://hsqldb.org/. [Accessed: 02-Apr-2015].

[30] “Spring Framework.” [Online]. Available: http://projects.spring.io/spring-framework/. [Accessed: 02-Apr-2015].

[31] P. Kruchten, “Architectural Blueprints — The ‘ 4 + 1 ’ View Model of Software Architecture”, IEEE Software 12 (6), vol. 12, November, 1995, pp. 42–50.

Towards A Smart-City Security Architecture

Proposal and Analysis of Impact of Major Smart-City Security Issues

Felipe Silva Ferraz[1,2], Carlos Candido Barros Sampaio[1,2], Carlos André Guimarães Ferraz[1]

[1]Informatics Center
Federal University of Pernambuco
Recife, Brazil
{fsf3, ccbs, cagf }@cin.ufpe.br

[2]CESAR
Recife Center for Advanced Studies and Systems
Recife, Brazil
{fsf, ccbs }@cesar.org.br

Abstract—**Concepts and solutions related to smart cities have been the focus of related studies and improvements for several years. Part of the motivation to the growing body of research in this field comes from the eminent need for solutions to address the present situation of urban environments. For the first time in human history, more than 50% of the population now live in big cities and not in the countryside. In this matter, computer networks and systems have an important role towards the construction of solutions that enable cities and citizens to maintain a continuous and agile use of those environments. At the same time that new solutions come forth offering readily available, integrated, and reliable information to citizens, new challenges related to information security arise. In this context, this paper explores a set of information security issues in the environment of a smart city and proposes a new approach (City Security Layer) based on the use of different and unique identifiers for each entity (citizen or sensor) involved in the relations of a city to its systems.**

Keywords—security; smart city; architecture; identification

I. INTRODUCTION

Beginning in the early 2000, a major proportion of the human population started to move from small towns to live in big cities [1]; this change in the world's urban structures has led to an unprecedented consumption of natural resources and added an enormous load on city systems [1][2].

To attempt to address this situation, cities have started to put efforts in creating more sustainable and green environments [1][3][4], and to offer its citizens with more and diverse services coming from existing systems of Education, Health, Public Safety, Resources, Government, and Public Transport [5]. To do so, investments in both time and money have been made to increase IoT adoptions to provide the city with more detailed and precise information [6][7][8] and services interoperability, guaranteeing increasing systems evolutions [5][9]. Combined with other definitions, such urban environments that are now extremely connected and highly technological are known as smart cities [10].

Pursuing interoperability in such a heterogeneous environment, new challenges arise, such as performance in the face of enormous amount of generated and transferred data [6][11][12][13]. Another is services availability to assure both citizens and the city with access every time they have a need [14][15][16]. Information security also becomes a concern and how to ensure that sensitive data like a patient's medical records, a driver or vehicle's location or an engineer's structural plan, are dealt with appropriate and expected confidentiality [17][18][19][20].

Information security is an important challenge yet to be properly and fully addressed in the construction of smart cities. At the same time that it is necessary to develop the means to maintain data trafficked by such cities private, integrated, and available upon access, it is also necessary to provide the city systems with ways for the same data to be shared and to be equally protected.

Furthermore, Sen et al. [17] states that there are information security issues related to privacy in the role of a smart city. Thus, it is vital to deal with this situation because the data shared among a smart city environment is as sensitive as the citizen itself, but affirming that privacy issues are the main problem in terms of security would be over simplistic [18][21]. There are other kinds of issues that may pose as a threat to the entire urban system if they are not properly addressed [22][23]. In this scenario, it is vital to develop different architectures, protocols, and other policies that will allow citizens to better manage and access their data [24][25].

Standards such as OpenID [26][27], OAuth [28][29], and SAML [11][30] appear as good choices to provide cities with the means to integrate its services in an environment, with authorization and authentication capabilities. However, their strength relies on offering a unique ID within an environment that is responsible for sharing permission rather than them offering tools to manage individual IDs as separate information.

Rather than just providing an environment with authentication and authorization, it is important to provide citizens with the means to manage their own identity across a heterogeneous system [9][25], without compromise; the environment interoperability and the citizens' privacy and anonymity, ergo identity management, is a key enabler for smart cities' evolution and maintenance.

This paper presents a communication protocol, based on identification management that aims to increase security in smart cities' environment, providing entities (citizens, services, and sensors) with a mechanics to interact with systems using unique IDs for each system. This paper is divided as follows: Section II will briefly introduce the concepts of smart cities, followed by Section III dealing with security analyses on smart cities. Section IV will depict 9 security issues that may affect smart cities' solutions, while Section V will describe the CSL (City Security Layer). Sections VI and VII will end this paper, analyzing the impact of the proposed approach.

II. SMART CITIES

Today's level of urbanization has reached an unprecedented economical and social growth different from other ages; large cities now have the most part of the world's population and an increasing share of the world's most skilled, educated, creative, and entrepreneurial women and men [1]. More than 50% of people on the planet now live in large cities. According to the United Nations, this number will increase to 70% in less than 50 years [1]. This so-called *city growth* or emerging of urban life is driving the city infrastructure into a stress level never before seen as the demand for basic services increases and is exponentially overloaded [5].

Cities are becoming increasingly empowered technologically as their core systems (i.e., Education, Public Safety, Transportation, Energy and Water, Healthcare and Services) are instrumented and interconnected, enabling new ways to deal with massive, parallel, and concurrent usage. In the same pace, new challenges in the field of information security rises and must be properly addressed.

Tracing the genealogy of the word 'smart' in the label 'smart city' can contribute to an understanding of how the term 'smart' is being used in this field. In marketing language, smartness is centered on user perspective. Because of the need for appeal to a broader base of community members, 'smart' serves better than the more elitist term 'intelligent'. Smart is more user-friendly than intelligent, which is limited to having a quick mind and being responsive to feedback. Smart city is required to adapt itself to the user needs and to provide customized interfaces [36]. In the urban planning field, this defines smartness.

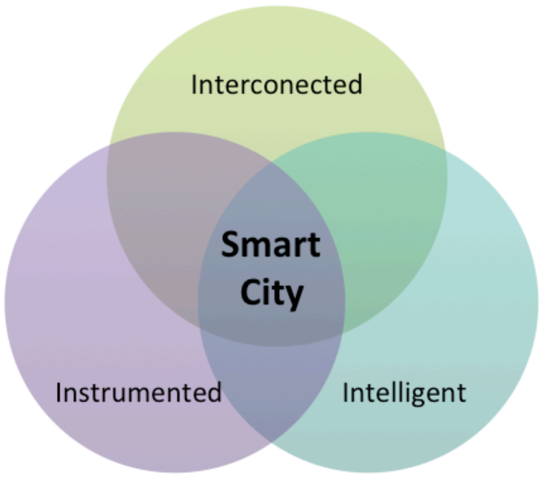


Figure 1. Smart city concept based on intelligence, connection, and instrumentation

Another perspective, represented in Figure 1, points out three main characteristics towards a smart city definition; it is an environment that is instrumented, interconnected, and intelligent [32].

Each one has a meaning:

Instrumented: means a city covered by a set of sensors that could be both physical and social. Through those sensors, the cities' core systems have access to real-time and reliable information. This relates directly with the IoT concepts.

Interconnected: means a vast set of systems working together to offer information from different points and sources. A correct combination of interconnected and instrumented systems creates a connection from the physical world to the real world.

Intelligent: refers to an instrumented and interconnected environment that makes the best use of information obtained from different sensors and systems, to offer a better life to the citizen.

Offering just one or any combination from those three concepts creates a scenario where a vital part will be missing. To illustrate, a system may have the means to extract the best from a set of information, but it does not have data to analyze. A system may also have the data to analyze but does not have ways to pass through other points of the environment its discoveries and information.

However, offering an environment so broadly constructed could have the side effect of creating a different set of scenarios where security information flaws could be created and explored.

III. SMART CITIES' SECURITY ANALYSES

Apart from the number of studies and protocols related to information security, the amount of vulnerabilities in connected applications has increased in the past few years [14]. In this matter, smart-city systems will demand a specific treatment to address its specific information security challenges [18].

According to [5][17][18][19][36], smart-city solutions depend on a high degree of connectivity, so that their systems (such as Education, Government, Traffic, Security, Resources, and Health) can create an interoperable network, offering citizens with more powerful, accurate, and innovative [35] services. For this reason, one of the biggest challenges facing smart-city development is associated to information security in the scope of interoperable systems [1]. Information security is a critical issue due to the increasing potential of cyber attacks and incidents against critical sectors in a smart city.

Information security must address not only deliberate attacks, such as from disgruntled employees, industrial espionage, and terrorists, but also inadvertent compromises of the information infrastructure due to user errors, equipment failures, and natural disasters. Vulnerabilities might allow an attacker to penetrate a network, gain access to control software, and alter load conditions to destabilize the system in unpredictable ways. To protect a smart city in a proper way, a number of security problems have to be faced according to a specific design/plan.

Believing that a traditional security approach based on privacy keeping, authorization, and authentication concept can simply be added into a city's critical infrastructure to make it safer as a city becomes smarter, is far from the real scenario. To deal with new paradigms related to smart cities is necessary to think about in terms of new architectures and not only to improve services and current solutions [36].

This class of services is fundamental to the success of the future city, and represents a topic of such complexity that it is beyond the scope of this paper to cover in detail. As an illustration, let us explore the design of identity services for the future city – which is required to maintain privacy while maintaining security.

The integration of the identity of *the citizen* across multiple systems and services, and the ability to provide a joint-up response to the needs of life events, comprises the goal of allowing the citizen to manage their own identity. This also includes what information is released about them to whom or when, while anonymous, aggregate data are made more widely available [25].

Thus, identity management is a key enabler for future cities. A unified identity system albeit one that can integrate with multiple identity providers and different forms of authentication and identification is needed to handle the extensively 'wired' nature of the city and the density of data transactions, systems, and solutions diversity [25].

Citizens or entities will use their identities to gain access to services and systems, and through benefits that they offer. This is a way to integrate to several solutions (systems and services), entities and service will eventually repeat their identification artifact in different moments and situations.

Ideally, every citizen and/or entity shall have a number of identities, each of which is made up of a number of attributes, which are either exposed, or used to validate a claim without exposing the information. The use of multiple identities limits exposure of truly important credentials, minimizing risk of abuse and identity theft, while allowing for the exposure of less critical information that is helpful for participants in the city ecosystem such as retailers, building operators, service providers, and governments [25].

Not only will citizens be in charge of their identities, but also the information that constitutes them, and when this information could be exposed. The proposed solution is intended to build a trusting relationship between the city, the services/systems, and the citizens. This will allow the acquisition and flow of information that are helpful to all participants without compromising their identity.

IV. Security Issues in the context of a Smart City

Previous studies brought into attention the need to make further improvements related to information security on smart-city environments [5][17][18][19][36]. Based on this need, this section will describe a set of 9 security issues that an urban system and a city may be under the risk of [22][23].

It is important to add to this train of thought that even though the technical solutions applied in those environments handle questions, such as Code Injection [37], Cross Site Scripting [37], Cross Site Request Forgery [37], Buffer Overflow [37], and so on, the issues posted in this work explore concepts related to the nature of a smart-city system. Our approach is to present a set of issues that, regardless of the technical solution applied, may be a threat to urban cities (or a smart-city system) in a different level.

Urban systems are composed by *Citizens* using *Solutions,* which could be *Platforms, Frameworks*, and *Applications*; all of those built on *Technologies* to receive and use *Data*. Urban system security issues or security issues, in brief, in the context of a smart city are situations that can pose as problems to the entire infrastructure of a smart city [22][23].

In the following, 9 security issues will be described; the focus will be the explanation of scenarios and situations that could be a potential threat to an urban environment and its systems.

A. Access to information from applications

According to Sen et al. [17], packet transfer must be studied to apply efforts on adding security to improve data privacy and integrity.

From a network and access perspective, devices have the means to access a packet, or a set of packets, in different ways and locations using different amounts of effort. For instance, to reduce latencies during data transfer, local copies or cache values of those packets could be created, and from there, the mentioned data could be retrieved not only from the network or during a transfer, but also from a local device.

To illustrate further, a sensor connects to a server to identify and authenticate user A and retrieves its permissions. During this process, user B could intercept the packet in different points of the network or of the device, and gain a set of information from user A and the service it is accessing.

B. Information tracking

It is important to have an interoperable and interconnected environment for systems to interact with one another like in [5][11]–[15]. It is also extremely important that, for instance, the information used by system B and that are originally created by system A, cannot be tracked back to its origin. This means that even though system A has provided a set of information to B, a user from system B should not realize that this information is from another part or user.

As an example, let us assume that system A provides information to a solution B.

Let us suppose that A is a system of criminal reports; B is another solution that uses those criminal reports to define the most suitable place to open a new commercial building, based on criminal records. The information used by B, which was provided by A, must not be unveiled or disclosed. This situation could destroy the anonymity in A and compromise, for instance, witnesses and victims of crimes.

C. Citizen tracking

Solutions for smart cities make use of different sensors (physical or social); those sensors are used to collect data from several city systems, and based on this, it is possible for urban systems to have a better city management.

To avoid further problems, such sensors must be under the control of a responsible entity to preserve the integrity of its functionality and generated data.

Among the possible problems raised by this feature is that it may be open or subject to unauthorized citizen tracking, discovery of movement patterns, and may cause 'flooding' of directional advertisement/merchandising.

D. User/Citizen data loss

Smart systems, within the context of smart cities, may use devices, such as smart phones, tablets, and other gadgets to gather a wide range of data and information. Depending on the data type handled by such devices, it is possible to have personal and sensitive data, such as messages, pictures, appointments, bank account numbers, contacts details, and others.

This issues deals with the concept that applications are saving precious data in the device, and if are not well treated, those data could be lost or compromised, creating significant problems to the citizen.

This could be achieved by adding a mechanism related to client cryptographic storage [5], system isolation, and even solutions related to authorization and authentication mechanics.

E. *Crossed access to information in data centers*

For this scenario, we deal with situations related to unauthorized access to information by exploiting flaws on the server side.

If by any means data security is violated, for instance, while they are under storage, analysis, and management procedures, the entire system may be compromised.

For example, when accessing information related to students' educational systems, a given entity (application) can recover criminal records, from a non-specific connection related to this citizen even though the solution should only be using Educational Services. This situation may occur because both systems share a common area or permissions that must be respected to avoid this kind of behavior.

F. *Crossed access at the client side*

Description (E), *Crossed access to information in data centers*, details a situation related to unauthorized access in the server side.

Issue (F), in this current topic, brings forth a subject related to unauthorized access at the client side, for instance in a mobile that holds sensitive information.

This is different from issue (D), which is concerned about *every* information saved and that is not properly stored, and is liable to undesired access within the context of a device.

For instance, system A saves in a device values related to paid fees, and system B uses the same mechanism to store information regarding the user financial account. If the device does not provide A and B with the correct isolation, it is possible that through A, an attacker can gain access to values presented in B, and even more, it is possible that a malicious third part system may be installed, and then gain access to both systems information [36].

G. *Lack of in-depth security*

According to OWASP TOP 10, one of the top-ten risks to web application is related to code injection. Also according to OWASP, sanitizing input values and removing undesired text are measures that can be used to avoid this issue and other security flaws [37].

This flaw, (G), relates to systems that do not validate data in different layers, and are compromised in any level, by data coming from other services.

On the other hand, in-depth security relates to the concept of adding several security measures in different layers of a solution [38].

In an interconnected environment, like in an urban system, if a system C does not provide the entered data with proper sanitization, other solutions that do not use concepts of in-depth security may also be affected.

In other words, if system A provides the user with a rich UI environment, and has several validations and sanitization in this part, if the back-end structure does not apply the same criteria, whenever a system C sends data to system A, system A may use this data. It means that if C has a malicious code inputted, it might be transferred to A once A misses its defense in terms of in-depth security check.

H. *Viral effect in urban environment*

A smart city uses an interoperable environment to provide solutions with the opportunity to interact with other systems, exchanging data, and creating more value to its citizens[36].

If the border of these relations is not well defined, the systems may face a scenario where a value is changed in system A and when system B uses this changed value, it may corrupt the information used in system B.

For instance, let us assume that this environment is made of a set of systems (named A, B, C… Z), we can foresee a situation where A provides B with an infected value while B may provide or transfer to C, D ... Z systems the same infected value. For an attacker to infect the entire environment, only a small portion of the system needs to be infected and then the contamination may spread throughout the entire system.

In issue (G), our main concern is with the lack of protection in every layer, and how this could be a problem that an urban scenario is highly connected. In the present issue, our concern is with the consequences of issues like issue (G), if the system is highly connected and lacks protections in several parts, the consequences of an attack can be exponentially increased, infecting the entire solution through the infection of a small part.

I. *Infection traceability and recovery*

The amount of data used and stored by a smart city has reached unprecedented levels. Moreover, the connection between systems has created a system of systems structure that provides those solutions with data coming from different services.

Issue (H) presents a viral threat related to a set of data that can share or provide another service with different data, creating, at some level, a self-sustained system. From that point of view, this issue presents a consequence for issue (I). Due to the amount of data and interconnected system, it is possible for an infection to maintain its origin undetected and beyond data recovery.

Using as an example, system A, with terabytes of data, exchanging values with a System B, that feeds systems C and D with updated and new values, processed from A data. D, on other hand, keeps passing some fine-grained data back to system A. If A suffers an infection having its data compromised, B will be fed with infected data, spreading the infection to other systems, like C and D. As soon as the infection could be detected, recovery processes may not be an option because the amount of data compromised is too big to be restored to a previous form. In addition, due to the relation between systems, the infection source may not be detectable.

V. SMART CITY SECURITY LAYER

OAuth, SAML, and OpenID are architectural solutions focused on identification and assets protection. Those assets could be any type of information or entities such as documents, data, and photos, among others. Through its adoption, it is feasible to create mechanisms in which it is possible to pass the responsibility of security measures to a third party, that could be a known server (Facebook, Google, and others), or to implement the same approach in an in-company solution.

A. Objective

Smart CSL's main objective is to be a layer, where the change of a sent identifier for another identifier is made. The new identifier will be generated from the combination of an ID and the accessed service.

Through that mechanism, it will be possible to make an entity keep its identity secret from a service and unique within the whole environment. This can still be done even though the same entity can access different sets of services, as the creation of the ID is made from the combination of two other identifiers; the resultant ID will be different for the different service coming from the same entity.

B. General view

The City Security Layer (CSL) is a mechanism based on the concept of change identifiers involved in a system relation. The following are the basic components and flow of CSL.

Entity: which is a component that is requesting information to a service; an Entity can be anything from a citizen, to a sensor or a service that is interoperating with another one.

Service: represents any service contacted from an entity.

Communication Layer: represents a contact point from entity to service, responsible for changing the identifier sent by the entity into the correct ID to be used within the service.

ID Service: is a component responsible for storing and managing information to generate the correct ID.

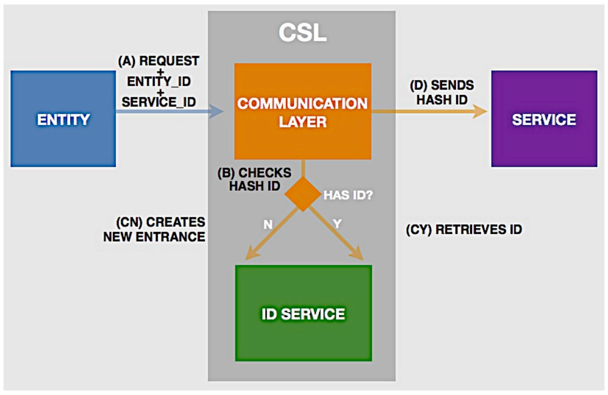


Figure 2. CSL basic flow.

C. Comparing with related works

CSL's main strengths rely on the fact that for each entity using a specific service, a unique and new ID will be presented to the service. For each system adopting this approach, the identifier sent by the layer will be interpreted as a real individual, protecting the real user through the protection of its real ID. On other hand, CSL presents no other feature, like OAuth, SAML, and OpenID, related to authentication and authorization.

Finally, CSL has presented an interesting contribution mostly due to the fact that it only relies on ID changing and management, providing the environment with means to use different IDs. However, it is also important to use an extra layer or process to take care of authorization and authentication under the scenario of an urban environment.

VI. ANALYSES OF ISSUES

Analyzing through this scope, Table 1 shows a compilation of CSL impacts when compared against the 9 issues previously studied.

TABLE I. CSL COVERAGE

ISSUES	COVERAGE
1. Access to information from applications	✓
2. Information Tracking	✓
3. Citizen Tracking	✓
4. User/Citizen data loss	✗
5. Crossed access to information in data centers	✓
6. Crossed access in client side	✗
7. Lack of Security in Depth	✓
8. Viral effect in urban environment	✗/✓
9. Infection traceability and recovery	✓

A. Access to information from applications

Assuming the behavior that each packet will be sent through the net with different user IDs per system, even though an 'eavesdropper' can capture many of those packets, this issue will be addressed; hence, the amount of senders will now be considerably bigger, making it even harder try to understand who is who.

B. Information tracking

For issue (B), the same toughness to identify each user will isolate information, at the same pace, it will also influence issue (C), isolating also the citizen, because information about the entity will be protected, as a consequence, the entity will also be untraceable.

C. Citizen tracking

Through the adoption of CSL, information will be hard to track; ergo, the citizen will also be hard to track because their data will be hard to track.

D. User/Citizen data loss

As mentioned in OAuth, OpenID, and SAML section, the main strength in the three standards relies on interoperability and authentication, and for that issue, they do not impact (D). The same way goes to CSL, its concept proposes to change the way identifiers are sent and used by systems; therefore, this issue is not impacted by CSL.

E. Crossed access to information in data centers

Issue (E) is addressed by CSL from the point of view that even though an attacker can compromise a system, and gather information about citizen A and also access other systems, this attacker will have the perception that the systems databases are composed only of different entities, but in practice, for each system/service, an entity will be presented differently.

F. Crossed access in client side

The consequence behind these issues is directly related to an application A accessing information from application B in the client side, without the authorization to do so. In CSL, this has no impact because identifier changes are made not at the client side.

G. Lack of in-depth security

CSL adoption will add an extra layer, responsible for creating and maintaining different IDs for different users. If citizens are to access a service, their IDs will pass for an adaptation to retrieve the real ID. Even if the requesting party is a service, retrieving information about a specific entity, it will also be submitted to CSL approaches, enhancing security in the environment as a single piece.

H. Viral effect in urban environment

The basic idea from issue (G) applies to issue (H); it will be more difficult to explore breaches due to the existence of an extra layer, but, issue (H) deals with a further consequence, which is the creation of a viral effect. This effect could be produced in different forms and with different types of data, not only a citizen ID, and that said, issue (G) is only partially addressed.

I. Infection traceability and recovery

The adoption of CSL will increase security, mostly because it will add an extra security layer that will provide the city systems with the means to keep its entities and citizens using different IDs for different services. As a consequence, this will promote systems isolation.

Even though they are isolated, they are not disconnected, that said, issues (G) and (I) are addressed completely by the CSL proposal. Issue (G) was presented with an extra layer; for issue (I), better traceability of the origins of an infection will be possible.

VII. CONCLUSION AND FUTURE WORK

For the first time in human history, humanity is facing a unique situation where more than 50% of the population now live in big cities. For that to work out, there is an urgent need to evolve information technology systems to solutions that provide citizens with more and detailed information about different subjects of their daily use.

At the same time that new solutions rise, new challenges also develop, and among those, information security plays an important role, and not only due to citizens' privacy issues, as it is a subject that may go beyond citizens and impact entire systems.

Solutions like OpenID, SAML, and OAuth play an important part in guaranteeing user security and single sign on. Unfortunately, lay all expectations in one of those 3 standards may not address and solve all problems. In this scenario, this paper proposed the creation of an architectural solution, called city security layer, an architecture based on a cryptography that proposes the creation of different and unique IDs for each system relating to each citizen. This way, it is possible to address more security issues than with the 3 mentioned standards.

CSL still needs further studies and evolutions to be considered as a final solution. In this matter, the next steps for this project are to develop an environment to better validate the proposed approach, and conduct some stress and performance tests to CSL implementation to guarantee its reliability and applicability.

REFERENCES

[1] S. Dirks and M. Keeling, "A vision of smarter cities: How cities can lead the way into a prosperous and sustainable future," *IBM Inst. Bus. Value. June*, 2009.

[2] IBM. Ibm smarter healthcare. http://ibm.co/bCJpHX, 2012. "[Online] Available: 19-March-2015".

[3] C. G. Kirwan, "Urban Media : A Design Process for the Development of Sustainable Applications for Ubiquitous Computing for Livable Cities," pp. 7–10.

[4] P. Jollivet, "Crowd sourced security, trust & cooperation for learning digital megacities: valuing social intangible assets for competitive advantage and harmonious development," *IET Int. Conf. Smart Sustain. City (ICSSC 2011)*, pp. 52–55, 2011.

[5] F. Ferraz, C. Sampaio, and C. Ferraz, "Towards a Smart City Security Model Exploring Smart Cities Elements Based on Nowadays Solutions," *ICSEA 2013*, pp. 546–550, 2013.

[6] R. van Kranenburg and A. Bassi, "IoT Challenges," *Commun. Mob. Comput.*, vol. 1, no. 1, p. 9, 2012.

[7] M. Chen, "Towards smart city: M2M communications with software agent intelligence," *Multimed. Tools Appl.*, vol. 67, no. 1, pp. 167–178, Feb. 2012.

[8] M. Fazio, M. Paone, A. Puliafito, and M. Villari, "Heterogeneous Sensors Become Homogeneous Things in Smart Cities," *2012 Sixth Int. Conf. Innov. Mob. Internet Serv. Ubiquitous Comput.*, pp. 775–780, Jul. 2012.

[9] Y. Wang and Y. Zhou, "Cloud architecture based on Near Field Communication in the smart city," in *2012 7th International Conference on Computer Science & Education (ICCSE)*, 2012, no. Iccse, pp. 231–234.

[10] T. Nam and T. A. Pardo, "Conceptualizing smart city with dimensions of technology, people, and institutions," *Proc. 12th Annu. Int. Digit. Gov. Res. Conf. Digit. Gov. Innov. Challenging Times - dg.o '11*, p. 282, 2011.

[11] T. Tran and C. Wietfeld, "Approaches for optimizing the performance of a mobile SAML-based emergency response system," *2009 13th Enterp. Distrib. Object Comput. Conf. Work.*, pp. 148–156, Sep. 2009.

[12] I. B. M. Global, B. Services, and E. Report, “Smarter cities for smarter growth, How cities can optimize their systems for the talent-based economy.”

[13] Z. Fan, Q. Chen, G. Kalogridis, S. Tan, and D. Kaleshi, “The power of data: Data analytics for M2M and smart grid,” *2012 3rd IEEE PES Innov. Smart Grid Technol. Eur. (ISGT Eur.*, pp. 1–8, Oct. 2012.

[14] J. M. Gonçalves, “Privacy and Information Security in Brazil? Yes, We Have It and We Do It!,” *2010 Seventh Int. Conf. Inf. Technol. New Gener.*, pp. 702–707, 2010.

[15] R. Giaffreda, “Enabling Smart Cities through a Cognitive Management Framework for the Internet of Things,” no. June, pp. 102–111, 2013.

[16] F. Hu, M. Qiu, J. Li, T. Grant, D. Tylor, S. Mccaleb, L. Butler, and R. Hamner, “A Review on Cloud Computing : Design Challenges in Architecture and Security,” pp. 25–55, 2011.

[17] M. Sen, A. Dutt, S. Agarwal, and A. Nath, “Issues of Privacy and Security in the Role of Software in Smart Cities,” *2013 Int. Conf. Commun. Syst. Netw. Technol.*, pp. 518–523, Apr. 2013.

[18] A. Bartoli, J. Hernández-Serrano, and M. Soriano, “Security and Privacy in your Smart City,” *cttc.cat*, pp. 1–6.

[19] W. M. da Silva, A. Alvaro, G. H. R. P. Tomas, R. a. Afonso, K. L. Dias, and V. C. Garcia, “Smart cities software architectures,” in *Proceedings of the 28th Annual ACM Symposium on Applied Computing - SAC ’13*, 2013, p. 1722.

[20] Z. Chen, W. Fan, Z. Xiong, P. Zhang, and L. Luo, “Visual data security and management for smart cities,” *Front. Comput. Sci. China*, vol. 4, no. 3, pp. 386–393, Aug. 2010.

[21] S. Report, “Software Security Assurance,” 2007.

[22] F. S. Ferraz and C. A. G. Ferraz, “More Than Meets the Eye In Smart City Information Security: Exploring security issues far beyond privacy concerns,” in *IEEE computer science, UFirst-UIC 2014*, 2014.

[23] F. S. Ferraz and C. A. G. Ferraz, “Smart City Security Issues: Depicting information security issues in the role of a urban environment,” in *IEEE Cloud Computing Initiative, UCC 2014*, 2014.

[24] G. Suciu, A. Vulpe, S. Halunga, O. Fratu, G. Todoran, and V. Suciu, “Smart Cities Built on Resilient Cloud Computing and Secure Internet of Things,” *2013 19th Int. Conf. Control Syst. Comput. Sci.*, pp. 513–518, May 2013.

[25] L. PlainIT, “Cities in the Cloud, a living planit introduction to future cities technologies.”

[26] A.-V. Anttiroiko, P. Valkama, and S. J. Bailey, “Smart cities in the new service economy: building platforms for smart services,” *Ai Soc.*, Jun. 2013.

[27] M. Noureddine and R. Bashroush, “An authentication model towards cloud federation in the enterprise,” *J. Syst. Softw.*, vol. 86, no. 9, pp. 2269–2275, Sep. 2013.

[28] OAuth, “OAuth 2.0.” [Online]. Available: www.oauth.net.

[29] B. Leiba, “OAuth Web Authorization Protocol” pp. 1–3, 2012.

[30] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, “Security and Cloud Computing: InterCloud Identity Management Infrastructure,” *2010 19th IEEE Int. Work. Enabling Technol. Infrastructures Collab. Enterp.*, pp. 263–265, 2010.

[31] M. Naphade, G. Banavar, C. Harrison, J. Paraszczak, and R. Morris, “Smarter Cities and Their Innovation Challenges,” *Computer (Long. Beach. Calif).*, vol. 44, no. 6, pp. 32–39, Jun. 2011.

[32] IBM. Ibm smarter healthcare. http://ibm.co/bCJpHX, 2012. ” [Online] Available: 19-March-2015”

[33] C. Balakrishna, “Enabling Technologies for Smart City Services and Applications,” *2012 Sixth Int. Conf. Next Gener. Mob. Appl. Serv. Technol.*, pp. 223–227, Sep. 2012.

[34] N. Mitton, S. Papavassiliou, A. Puliafito, and K. S. Trivedi, “Combining Cloud and sensors in a smart city environment,” *EURASIP J. Wirel. Commun. Netw.*, vol. 2012, no. 1, p. 247, 2012.

[35] M. Batty, K. W. Axhausen, F. Giannotti, A. Pozdnoukhov, A. Bazzani, M. Wachowicz, G. Ouzounis, and Y. Portugali, “Smart cities of the future,” *Eur. Phys. J. Spec. Top.* vol. 214, no. 1, pp. 481–518, Dec. 2012.

[36] I. Verbauwhede, “Efficient and secure hardware,” Datenschutz und Datensicherheit - DuD, vol. 36, no. 12, pp. 872–875, Nov. 2012.

[37] OWASP, “OWASP Top 10 - 2013 : The most critical web application security risks,”. https://www.owasp.org/index.php/Top_10_2013-Top_10, 2013 "[Online] Available: 19-March-2015”

[38] M. Schumacher, E. Fernandez-Buglioni, D. Hybertson, F. Buschmann, and P. Sommerlad, *Security Patterns : Integrating Security and Systems Engineering (Wiley Software Patterns Series)*. John Wiley & Sons, 2006.

Towards an Automatic Test Case Decomposition by Means of System Decomposition

Marcel Ibe and Andreas Rausch

Clausthal University of Technology
Email: marcel.ibe@tu-clausthal.de, andreas.rausch@tu-clausthal.de

***Abstract*—Quality assurance takes a huge effort during a software development project. Especially the generation of test cases and test data but also the execution, analysing the results and the maintenance consumes a lot of resources. Model-based testing tries to reduce the effort by automating several testing activities. In this paper, an approach for test case decomposition by means of system decomposition is introduced. The system's structure is described by composition structure diagrams, the systems behaviour by state charts. By transferring structural decomposition steps and use of the additional behavioural descriptions, existing test cases can be adapted to the refined system description automatically.**

Keywords–Model-Based-Testing, Test-Case-Decomposition, Sequence Diagram

I. INTRODUCTION

Testing is one of the most widely used practices to ensure high quality of software systems. At random the real behaviour of a system or a component is compared with the desired one by in advanced defined test cases. It takes up a very big share of the effort of a software development project [1]. Model-Based Testing (MBT) uses models to support testing activities, for example, by generating test cases automatically. By means of a test case specification a finite set of test cases (a test suite) is selected, that will be executed on the system [2]. According to the test case specification, the resulting test suite can contain a very large number of test cases. Since the number of test cases in a test suite is one of the factors that have a significant influence on total testing costs [3], one would like this number to be as small as possible. However, an afterward reduction of the test suites size can be both a hard problem [4] [5], as well as have an adverse effect on the quality of the test suite [6]. Several studies suggest that manually derived test cases provide an alternative to automatically generated ones. For example, Pretschner et al. [7] have found that not the size of a test suite but rather the basis of the test case generation reveals about the fault-finding ability. They observed that test suites containing a much higher number of automatically generated test cases detect only a few more errors than fewer, hand-crafted test cases. Marques et al. [8] compared manual ad hoc tests with automatically generated ones. They confirmed the observation, that manually derived test suites are usually smaller but not less effective in finding bugs. The study even gave evidence that manually derived test suites could find more major bugs. This suggests that a small test suite containing manually derived test cases provides an alternative solution to automatic test case generation. Although, the manual test case derivation has the main disadvantage of higher effort, it is nevertheless feasible in limited range of the software development process like user acceptance testing. If the strong fault-finding ability with the small test suite size could be transferred to other testing levels, the total costs of testing could be reduced.

The contribution of this paper is to introduce an approach that allows reusing manually derived test cases at different testing levels by automatically decompose these test cases analogously to the decomposition of the system under test (SUT). In this way, test suites for component or unit testing can be created that also have a strong fault-detection capability but contain only a small number of test cases.

In this paper, we present an approach for decomposing test cases by means of system decomposition to create new test cases for testing the SUT at different levels of decomposition. The next section introduces the overall approach, a running example and specifies the requirements to test case decomposition at the example. In Section 3, the test case decomposition is described in detail. Section 4 gives an overview over related work. In the last section, the results and plans for future work are presented.

II. OVERALL APPROACH

In this section, we introduce our approach for decomposing test cases. Figure 1 shows schematically how the test case decomposition can be applied during a software development project. Based on the requirements of the customer a first specification of the system is created manually. At the same time, test cases for user acceptance testing are derived. These test cases are black box tests that do not consider the internal structure but only the systems behaviour. During the development, the specification is getting more and more detailed by decomposing the system into components and describing the behaviour of these new components. Now, instead of creating new test cases for testing these components the already existing test cases can be used by enriching them with the new information about the internal structure and behaviour of the system. For every decomposition step at the systems specification these decompositions are transferred automatically to the test cases and so they are adapted and able to test the new defined system components. After starting implementing the specified components of the system the decomposed test cases can be used to test the components against the specification.

For applying our approach, a structural description of the SUT and its internal structure consisting of components and their ability to interact with each other is needed. We use an adapted version of UML Composition Diagrams to describe the internal structure of the systems and its components. For the behavioural description of the SUT and its internal

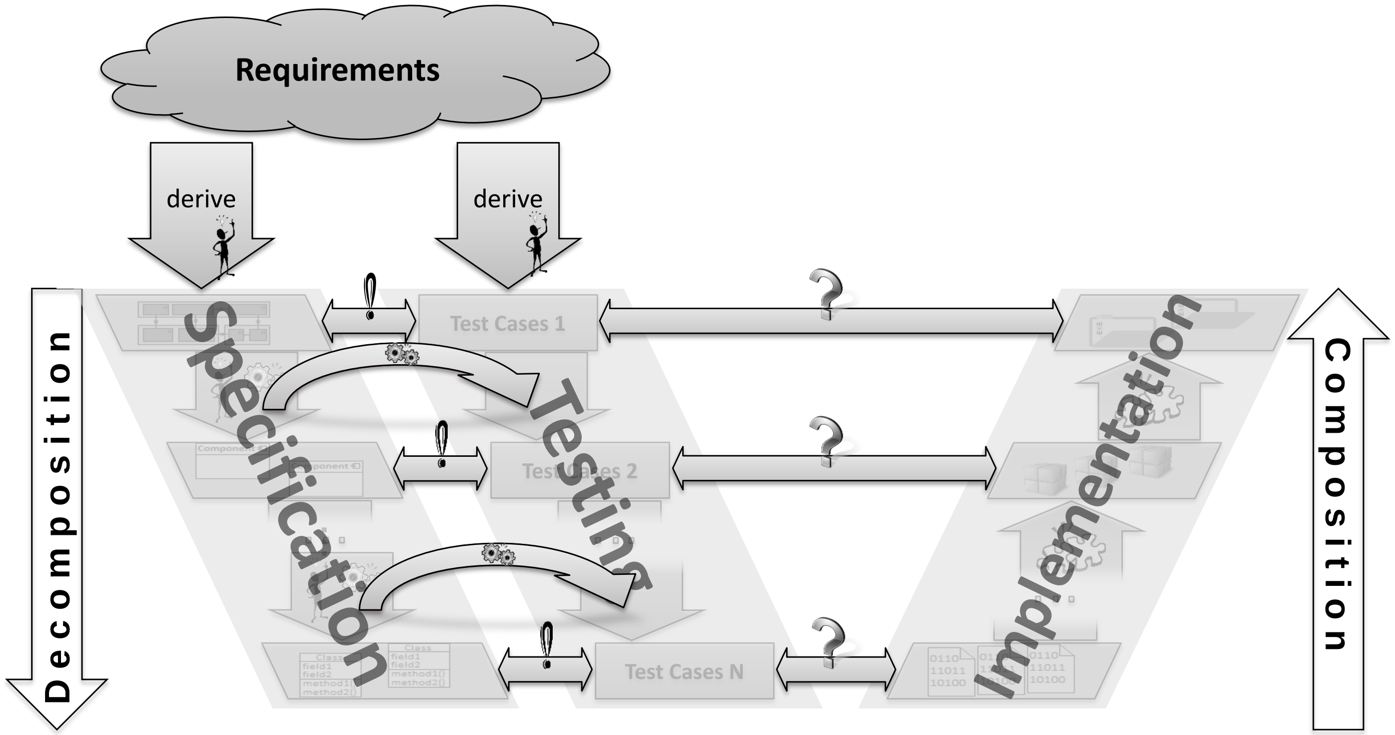


Figure 1. Schematic illustration of the test case decomposition approach applied during a software development process

components UML State Machines are used as graphical representation. At last we use an adapted version of UML Sequence Diagrams to describe the test cases.

In the next subsections, we introduce a running example and explain the diagrams used for describing the system's structure, behaviour and test cases in detail.

A. Running Example

To illustrate our approach, we use a small example of a central locking system (CLS) for cars which is inspired by the example used by Krüger et al. [9] and is shown in Figure 2. The upper image of Figure 2 shows that the CLS defines an interface with the four signals *lock*, *unlock*, *locked* and *unlocked* which are used for the communication of the CLS with its environment. The lower image of Figure 2 shows the state-based behaviour of the CLS. There are the initial state *unlocked* and the state *locked*. The CLS can switch from unlocked to locked state when receiving the signal *lock*. Additionally the signal *locked* is sent and vice versa when receiving the signal *unlock* and sending the signal *unlocked*.

B. Structural Description

The graphical representation of the structural description of the SUT and its internal components and subcomponents is based on UML Composition Structure Diagrams [10] (CSD). The structural decomposition of the CLS is shown in the left image of Figure 3. The CLS consists of the parts *control* and *motors*. The *control* part contains exactly one instance of the *control* component and the *motor* part contains two to five instances of the *motor* component. The parts can communicate with each other via the *MotorControl* interface, which is provided by the *motor* component and defines the

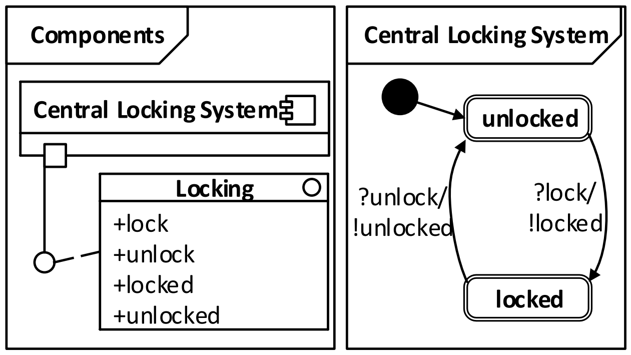


Figure 2. Structural (upper) and behavioural (lower) description of the central locking system

three signals *up*, *down* and *ready*. Furthermore, the *control* part can communicate with the environment of the CLS via the *Locking* interface, which is provided by the CLS itself and defines the signals *lock*, *unlock*, *locked* and *unlocked*. The *motor* and *control* components could be decomposed in the same way.

C. Behavioural Description

The behavioural description of the SUT and the components defined in the CSD is based on UML Statecharts [10] (SC). In addition to the components, the signals to be used in the SC are defined by the interfaces in the CSD.

The upper part of the right image of Figure 3 shows the behaviour of the CLS, which was already explained in Subsection II-A. In the lower part SCs, for *control* component

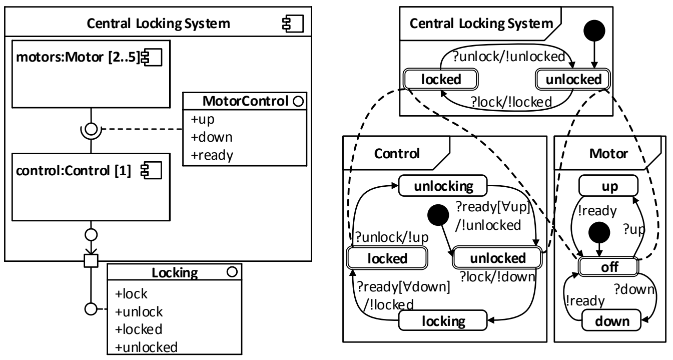


Figure 3. Structural (left) and behavioural (right) descriptions of the CLS and its components

and *motor* component are shown. For *control* component the four states *unlocked*, *locking*, *locked* and *unlocking* are defined. The double framed states *unlocked* and *locked* are so-called stable states. That means, the component may stay in this state for an unlimited period of time. *Unlocked* state is the initial state of this component. After receiving the signal *lock* being in this initial state it sends the *down* signal and switches to *locking* state. This state can only be left if every receiver of the *down* signal responses with a *ready* signal. Then this component switches to *locked* state after sending the *locked* signal. Switching the states to *unlocking* and *unlocked* states are performed analogous with the *unlock*, *up*, *ready* and *unlocked* signals. For *motor* component there are three states defined: The initial stable state *off* and *up* and *down* states. After receiving *up* respectively *down* signal, *motor* component switches to *up* respectively *down* state. Since, they are both no stable state *motor* component has to leave them and switch back to *off* state and consequently sends a *ready* signal.

The statecharts shown in Figure 3 describe the behaviour of components at two different decomposition levels of the SUT: The CLS at the top-most level and one level below *control* and *motor* component. For tracing the refinement between two levels a relation that assigns a set of states of the lower level to every stable state of the upper level has to be defined. If a SC of a lower level has both stable and not stable states the assigned set of states must contain at least one state from this SC. Else the relation to this lower level SC is optional. This relation will be used later to ensure that the components of a decomposed test case are in states that corresponds to a state of the component which was tested by the former test case. If there is no such relation for one or more SCs, the final states of the components in the decomposed test case do not have to be considered. In the CLS example *locked* state from *control* component and *off* state from *motor* component are assigned to *locked* state from CLS and *unlocked* state from *control* component and *off* from motor component are assigned to *unlocked* state from CLS.

D. Test Case Description

For describing the test cases, diagrams are used that are based on UML Sequence Diagrams [10] (SD). Every test case is described by one SD and consists of exactly one lifeline, which represents an instance of the component to be tested by this test case, and signals (messages) that are sent between the

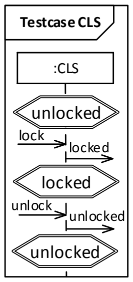


Figure 4. Graphical representation of a test case for CLS component

lifeline and its environment during this test case. This lifeline also contains the states that the corresponding component is in. The sent messages correspond to the signals that are sent or received during the transitions in the SCs and can cause a state change of the lifeline.

Figure 4 shows one test case for the CLS. Starting with the CLS lifeline in *unlocked* state receiving the *lock* message from its environment which causes sending the *locked* message and a switch to *locked* state. Then receiving the *unlock* message causes the CLS to send the *unlocked* message and a state change back to *unlocked* state. This behaviour corresponds to the SC for the CLS shown in Figure 2.

III. Test Case Decomposition

In this section, the test case decomposition is described in detail and its application is shown at the running example. The test case decomposition contains of two stages: Test case extension and test case partition. During the first stage, the test case is extended using the information of the systems decomposition from the CSD and the SCs of the new subcomponents. Hence, the test case is enriched with information about the internal communication between the subcomponents. Within the second stage the extended test case is partitioned into several test cases that respectively test one of the subcomponents of the component to test. These two stages of the test case decomposition are described in detail during the next subsections.

A. Test Cases Extension

In the first stage, a given initial test case is getting extended. That means that the structural decomposition and the new information about the behaviour of the decomposed components are transferred to the initial test case. The initial test case contains exactly one lifeline, which represents an instance of the component to be tested, the messages that are sent between the lifeline and its environment and the lifelines states during the test case. At first the new structural information is added to the initial test case by replacing the initial lifeline by lifelines for all instances that compose the initial component as describes in the CSD. After that the new behavioural information are added. This is done by retaining the initial messages from and to the environment and add the new messages which are sent between the new lifelines. Figure 5 shows the test case extension as pseudocode.

Using the example test case illustrated in Figure 4 we perform the test case extension. Figure 6 shows the test case at several intermediate steps during and after its extension. The algorithm gets as input the test case tc to be extended

```
   Data: test case $tc$ to be extended, lifeline $l_{tc}$ of component to
         be tested by $tc$, list of messages $M$ sent in $tc$
   Result: extended test case $tc$
1  Replace $l_{tc}$ by lifelines $L_{ext}$ for subcomponents
   List of free messages $M_f$ := $\emptyset$
   foreach $m \in M$ do
2      if $l_{tc}$ receives $m$ then
3          Mark $m$ as free incoming Message
4      else
5          Mark $m$ as free outgoing Message
6      $M = M \setminus m$
       $M_f = M_f + m$
7  while transition with ANY-Trigger available or $M_f \neq \emptyset$ do
8      while transition $t$ with ANY-Trigger available do
9          fire transition $t$
           $M_n$:= list of messages received due to fire $t$
           $M_f = M_n + M_f$
           update state of corresponding lifeline

10     if $M_f \neq \emptyset$ then
11         $m_i \in M_f$ first free incoming message
           $L$:=set of lifelines, that can receive $m$
           if $L \neq \emptyset$ then
12             foreach lifeline $l \in L$ do
13                 Create copy $m'$ of $m_i$
                   Bind $m'$ to $l$ and mark $m'$ as bound
                   $M_n$:= list of messages received due to $m'$
                   Update state of $l$
                   $M_f = M_n + M_f$
14         else
15             foreach free outgoing message $m_o \in M$ do
16                 if $m_i == m_o$ then
17                     Bind $m_o$ to sending lifeline $l$ of $m_i$
                       Update state of $l$
                       $M_f = M_f \setminus m_o$
18         Delete $m_i$ from $tc$ and $M_f$
```

Figure 5. Pseudocode for test case extension

containing the lifeline l_{tc} and a list of messages M that are sent between l_{tc} and its environment. After its execution, the algorithm returns the extended test case tc.

At the first step the algorithm replaces the lifeline l_{tc} by a set L_{ext} with lifelines in their initial state for every instance of a subcomponent as defined in the CSD of the component to test. For variable multiplicities of a part in the CSD, the lowest valid value greater zero is selected as number of instances. After this step the test case is adapted to the new structural information. Now, the empty list M_f is defined to collect all *free messages*. *Free messages* are messages, that do not have a sending (*free outgoing message*) or receiving (*free incoming message*) lifeline. In the following loop (lines 3 - 9) all messages in M are marked either as *free incoming* or *free outgoing message* depending on whether they were received or sent by lifeline l_{tc} and added to the end of M_f. The former order of the messages is retained. The current state of the test case is shown in the upper left image of Figure 6. The former lifeline *:CLS* was replaced by a *control* lifeline and two *motor* lifelines as two is the lower bound of part motors (see Figure 3). The *lock* and *unlock* messages were marked as *free incoming message* so one of the available lifelines can receive them. The *locked* and *unlocked* messages were marked as *free outgoing messages* and a lifeline has to be found that sends these messages.

The next loop (lines 10 - 31) is executed while there are transitions left that can be fired, more precisely transitions without a trigger (ANY-trigger) are available from the current states of the lifelines in L_{ext} or there are free messages left. First, all transitions with ANY-Trigger are fired (lines 11 - 15). Thereafter, new messages that are received due to firing these transitions are added at the top of M_f and the states of the corresponding lifelines are updated. If there is no transition with ANY-trigger left the first free incoming message $m_i \in M_f$ is bound to suitable lifelines in L_{ext}. Thereto, the corresponding statecharts of the lifelines in L_{ext} are searched for transitions that have the current state of the lifeline as source and m_i as trigger. For every possible receiver lifeline $l \in L_{ext}$ a copy m' of m_i is created and bound to l, i.e., l now receives the copy m' and m' is marked as bound (no longer a free message). Due to firing a transition, new free incoming messages M_n can be sent and are added at top of the list M_f after updating the state of lifeline l. If there are no possible receiving lifelines all remaining free outgoing messages m_o are compared to m_i and if there is a m_o with the same signal as m_i, m_o is bound to the lifeline sending m_i and m_o is removed from M_f. At the end the message m_i is deleted from the test case tc and the list M_f. In our example, there are no possible transitions with ANY-trigger, but there are free incoming messages. The first one that is chosen is the *lock* message. A copy of this message can be bound to the *control* lifeline because it is in *unlocked* state and there is a transition from *unlocked* state with trigger *lock* in the corresponding statechart. As a result of firing this transition, the new free incoming message *down* is added at top of M_f and *control* lifeline switches to *locking* state. Since, there are no more possible receiver lifelines the *lock* message can be deleted from the testcase. The upper central image of Figure 6 illustrates the current state of the test case.

The new *down* message can now be bound at the two *motor* lifelines. So there are two copies of this message, which are bound to these lifelines, causing them to switch to *down* state and the original message can be deleted. The current state of the test case is shown in upper right image of Figure 6. Now, there are two ANY-trigger transitions available which send the new free incoming *ready* messages and switch the *motor* lifelines back to *off* states. After binding these two *ready* messages to *control* lifeline it receives a new *locked* message and switch the lifelines state to *locked* state which is shown in the lower left image of Figure 6. Now, there is no possible receiver lifeline for this message so the algorithm looks for an equal free outgoing message. Since there is a free outgoing *locked* message this free outgoing message can be bound to *control* lifeline and delete the free incoming message *locked*. Hence, *control* lifeline sends the new *up* message (Figure 6 lower image). The following steps are analogue to them after adding the *down* message. After switching the *control* lifeline back to *unlocked* state there are no possible transitions with ANY-trigger and no free incoming messages left and the test case extension has finished. To check, whether the extension is correct it is checked if the final states of the lifelines of the extended test case are in a state that is in the set related to the

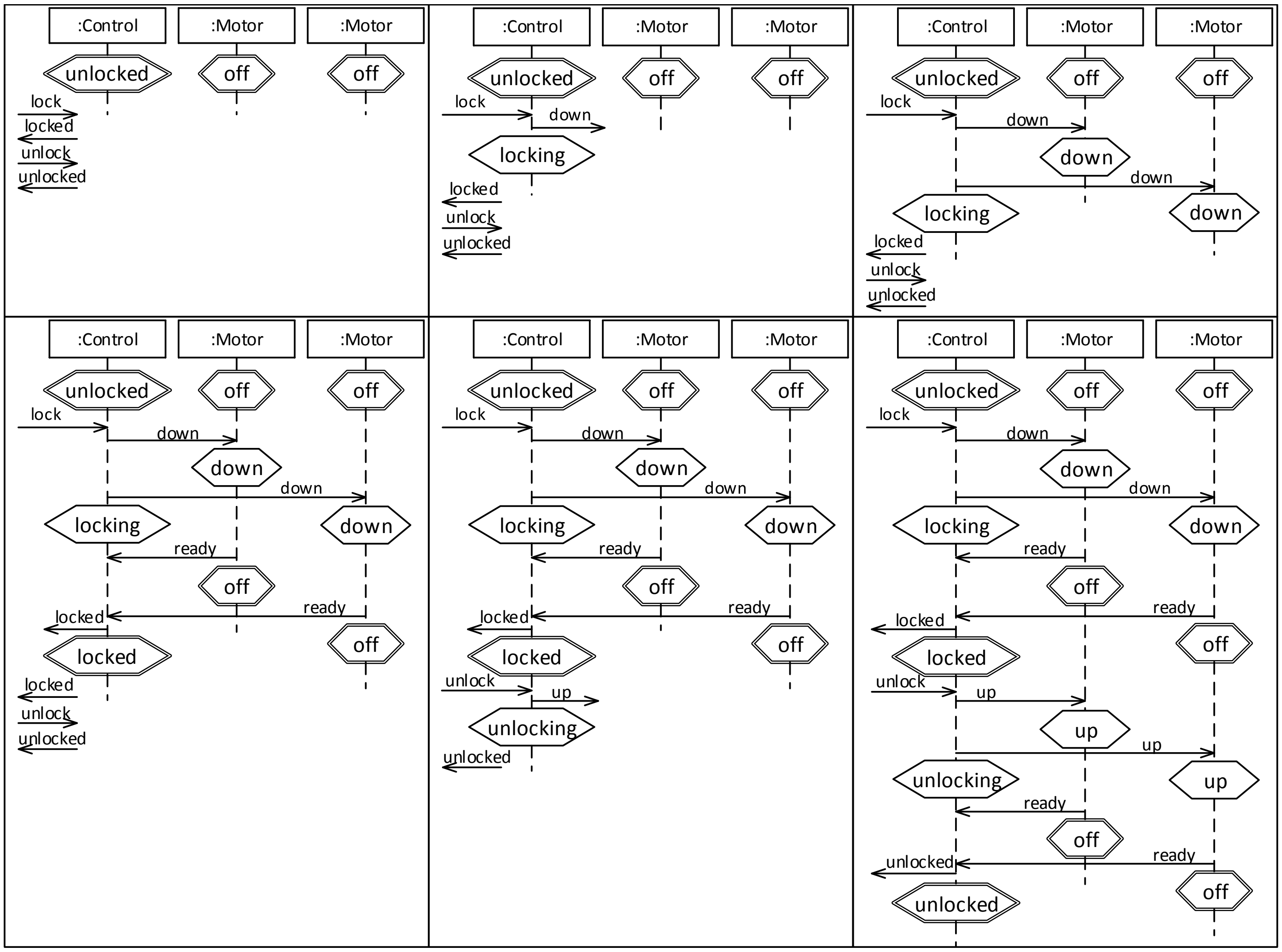


Figure 6. Extension of the sample test case

final state of the initial test case.

B. Test Case Partition

The test case that was extended by the algorithm presented in the last section has to be partitioned into several test cases for the several subcomponents. First the test case is vertically partitioned, i.e., for every lifeline only this one lifeline and only these messages, which are sent or received by this lifeline are considered. After this step there exists one test case for every lifeline that occurs in the extended test case. Now, these test cases can be partitioned horizontally at stable states, i.e., if there is a stable state that is not the initial or final state of a test case, this test case is split up there. The two new test cases contain only these messages that are sent before respectively after reaching the stable state. The stable state additionally becomes the final state of the first and the initial state of the last new test case. Since it is possible that several identical test cases are created, duplicated test cases can be refused. In our example the extended test case would be partitioned into six new test cases; two for every lifeline with each horizontal splitting at the middle stable state *locked* respectively *off*. The left image of Figure 7 shows the partitioning as dotted lines. Now, there are two times two identical test cases for the motor

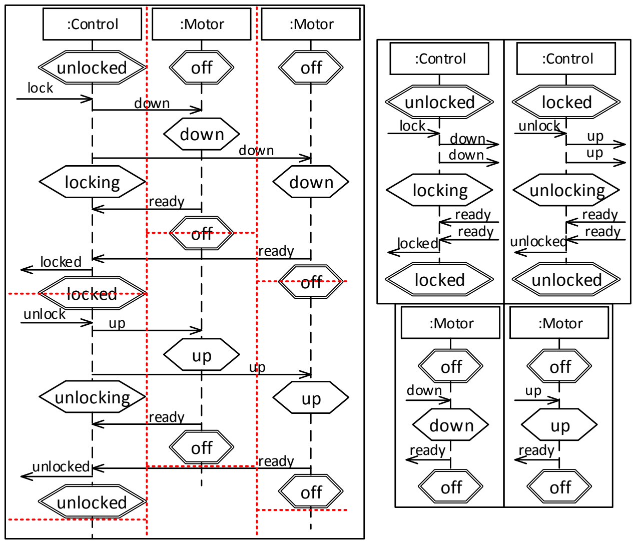


Figure 7. Partition of the extended test case

component so the duplicates can be removed and the four test cases are left that are shown in the right image of Figure 7.

C. Tool Support

To apply the techniques for describing the structure and the behaviour of the SUT as well as the test cases to test it and apply the test case decomposition a tool support will be provided. Currently the automatic test case extension based on structural and behavioural descriptions is supported. Additional graphical editors for creating the structural and behavioural descriptions and the test cases will be provided. Furthermore integration with tools for automatic test case execution and analysis like JUnit is planned.

IV. Related Work

Dias Neto et al. [11] give a survey of MBT approaches by comparing more than 400 papers. Approaches that use state based behaviour descriptions were developed amongst others from Bernard et al. [12]. They generate abstract test cases from state machines describing the behaviour of classes. Several selection criteria can be chosen. The test cases are only suitable to test classes but not components.

Another approach presented by Tretmans [13] uses labelled transition systems and the ioco-testing theory. However, the test case selection is still an open issue. The approach of Xu et al. [14] generates test cases by two different strategies: Structure-oriented and property-oriented generation. However, they also do not cover different levels of decomposition with their test cases.

The approaches of Elbaum et al [15] and Saff et al. [16] generate unit tests from system tests. But it is necessary to execute the system to get unit test. So they are not available at implementation time.

Briand et al. [17] investigated the impact of changing models on the generated test cases. They divided the test cases into three categories: Obsolete, retestable and reusable. However, they cannot update the test cases after changes, for example decompositions, of a model.

V. Conclusion and Future Work

We have presented an approach that enables an automatic test case decomposition by using the decomposition of the SUT. The test cases are extended with the systems or components internal communication and partitioned into several test cases that can be used to test the new components defined by the systems decomposition.

Future work includes consideration of the sequencing of existing and new messages during the test case extension as well as variable initial states by allowing a history for the behavioural description and the handling of indeterministic state charts. Another important aspect is tracing and impact analysis of changes of the initial test. Furthermore, the tool support will be improved by providing customized graphical editors for describing the systems structure and behaviour and the test cases and the integration of functional testing tools. After this, we plan to evaluate our approach at several existing system to compare it with existing test case generation approaches.

References

[1] M. Utting and B. Legeard, Practical Model-Based Testing: A Tools Approach. Morgan Kaufmann, Jul. 2010.

[2] A. Pretschner and J. Philipps, “10 methodological issues in model-based testing,” in Model-Based Testing of Reactive Systems, ser. Lecture Notes in Computer Science, M. Broy, B. Jonsson, J.-P. Katoen, M. Leucker, and A. Pretschner, Eds. Springer Berlin Heidelberg, Jan. 2005, no. 3472, pp. 281–291.

[3] G. J. Myers, T. Badgett, and C. Sandler, The art of software testing. Hoboken, N.J.: John Wiley & Sons, 2004.

[4] S. Sprenkle, S. Sampath, E. Gibson, L. Pollock, and A. Souter, “An empirical comparison of test suite reduction techniques for user-session-based testing of web applications,” in Software Maintenance, 2005. ICSM’05. Proceedings of the 21st IEEE International Conference on. IEEE, 2005, pp. 587–596.

[5] M. J. Harrold, R. Gupta, and M. L. Soffa, “A methodology for controlling the size of a test suite,” ACM Transactions on Software Engineering and Methodology (TOSEM), vol. 2, no. 3, 1993, pp. 270–285.

[6] D. Hao, L. Zhang, X. Wu, H. Mei, and G. Rothermel, “On-demand Test Suite Reduction,” in Proceedings of the 34th International Conference on Software Engineering, ser. ICSE ’12. Piscataway, NJ, USA: IEEE Press, 2012, pp. 738–748.

[7] A. Pretschner et al., “One evaluation of model-based testing and its automation,” in Proceedings of the 27th international conference on Software engineering, ser. ICSE ’05. New York, NY, USA: ACM, 2005, pp. 392–401.

[8] A. Marques, F. Ramalho, and W. L. Andrade, “Comparing model-based testing with traditional testing strategies: An empirical study,” in Software Testing, Verification and Validation Workshops (ICSTW), 2014 IEEE Seventh International Conference on. IEEE, 2014, pp. 264–273.

[9] I. Krger, R. Grosu, P. Scholz, and M. Broy, “From MSCs to statecharts,” in Distributed and Parallel Embedded Systems. Springer, 1999, pp. 61–71.

[10] C. Rupp, S. Queins, and B. Zengler, UML 2 glasklar: Praxiswissen fur die UML-Modellierung. Munchen; Wien: Hanser, 2007.

[11] A. C. Dias Neto, R. Subramanyan, M. Vieira, and G. H. Travassos, “A survey on model-based testing approaches: a systematic review,” in Proceedings of the 1st ACM international workshop on Empirical assessment of software engineering languages and technologies: held in conjunction with the 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE) 2007, ser. WEASELTech ’07. New York, NY, USA: ACM, 2007, pp. 31–36.

[12] E. Bernard et al., “Model-based testing from UML models.” in GI Jahrestagung (2), 2006, pp. 223–230.

[13] J. Tretmans, “Model based testing with labelled transition systems,” in Formal methods and testing. Springer, 2008, pp. 1–38.

[14] D. Xu, O. El-Ariss, W. Xu, and L. Wang, “Testing aspect-oriented programs with finite state machines,” Software Testing, Verification and Reliability, vol. 22, no. 4, 2012, pp. 267–293.

[15] S. Elbaum, H. N. Chin, M. B. Dwyer, and J. Dokulil, “Carving Differential Unit Test Cases from System Test Cases,” in Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering, ser. SIGSOFT ’06/FSE-14. New York, NY, USA: ACM, 2006, pp. 253–264.

[16] D. Saff, S. Artzi, J. H. Perkins, and M. D. Ernst, “Automatic Test Factoring for Java,” Tech. Rep. MIT-CSAIL-TR-2005-042, Jun. 2005.

[17] L. Briand, Y. Labiche, and G. Soccar, “Automating impact analysis and regression test selection based on UML designs,” in International Conference on Software Maintenance, 2002. Proceedings, 2002, pp. 252–261.