Requirements Engineering@Brazil

ER@BR2013 – 16 July 2013

Rio de Janeiro, Brazil

http://www.cin.ufpe.br/~erbr13

Jaelson Castro

Fernanda Alencar

Márcia Lucena

Gilberto Cysneiros Filho (eds.)

a CEUR Workshop Proceedings, ISSN 1613-0073

Index

Preface

The Requirements Engineering @ Brazil 2013 (**ER@BR 2013**) event is dedicated to the discussion of concepts, methods, techniques, tools, and applications associated with Requirements Engineering. It will focus on the Brazilian Requirements Engineering community addressing concerns related to three general viewpoints: research, practice and government.

It will be organized with short presentations, posters and panels, aiming to stimulate the dialogue between scientists, educators, professionals and students. The intention is to exchange experiences, present solutions as well as to survey issues and challenges in the area of Requirements Engineering.

The event is co-located with the 21st International Conference on Requirements Engineering (RE), benefiting from the common interests shared by the event and the conference. We have tried to keep the format small and informal so as to maximize interaction. We are holding a discussion-oriented event; as such the main criterion for paper acceptance in **ER@BR 2013** is relevance and potential for raising discussion. Presentation times for each regular paper, ranging from 10 to 20 minutes, have been split equally into presentation and discussion. Some papers have been presented as posters.

Concerning the review process, each of the 39 submitted papers went through a thorough review process with 3 reviews from a programme committee, providing useful feedback for authors. Revised versions of the 38 accepted papers are included in these proceedings. We thank authors and reviewers for their valuable contributions. The program was completed with a keynote given by Prof. John Mylopoulos entitled "Requirements Models at Design and Runtime" and keynote given by Prof. Anthony Finkelstein entitled "The Next 10 Years: the shape of software to come and what if means for Software Engineering". A panel on "Practices and Challenges of the Brazilian Industry in the Requirements Engineering Area was also included.

Last but not least, we want to deeply thank the organizers of the RE 2013 conference for their great support.

We look forward to lively conversations and debates with old and new friends at the workshop, in the wonderful surroundings of Rio de Janeiro, Brazil!

Jaelson Castro, Universidade Federal de Pernambuco, Brazil
Fernanda Alencar, Universidade Federal Pernambuco, Brazil
Márcia Lucena, Universidade Federal do Rio Grande do Norte, Brazil
Gilberto Cysneiros Filho, Universidade Federal Rural de Pernambuco, Brazil

Requirements Models at Design and Runtime

Prof. John Mylopoulos

University of Trento – Italy

We review the history of requirements models and conclude that a goal-oriented perspective offers a suitable abstraction for requirements analysis. We then sketch some of the desirable features (... "requirements") of design-time and runtime requirements models and draw conclusions about their similarities and differences. We also stake positions on the nature of modelling languages in general, and requirements modelling languages in particular.

The Next 10 Years: the shape of software to come and what if means for software engineering

Anthony Finkelstein

University College London, London, United Kingdom
a.finkelstein@ucl.ac.uk

Software engineering provides the method, tools and processes to support the development of complex software systems. As a discipline it must necessarily respond and adapt to the types of system that people want to build and the business context in which software development takes place. We have already seen major changes away from monolithic, custom-built systems to much more highly componentised distributed systems incorporating software packages, glue code and scripting. These changes have been paralleled by changes in the software business with outsourced development and community sourced middleware.

In this talk I examine the next set of transformations in the software business, some already evident, such as 'apps' and 'cloud' infrastructure, as well as some more speculative developments. I consider their implications for software engineering research and practice. We will pay particular attention to developments in technologies related to data management and interoperability.

Práticas e Desafios da Indústria Brasileira na Área de Engenharia de Requisitos

Detaques do Painel da Indústria do Re@Brasil

Maria Lancastre[1], Letícia Duboc[2]

[1]Escola Politécnica da Universidade de Pernambuco, Brasil

[2]Universidade do Estado do Rio de Janeiro, Brasil

([1]mlpm@ecomp.poli.br, [2]leticia@ime.uerj.br)

Abstract. This article presents a brief description of the panel: "Practices and Challenges of the Brazilian Industry in the Requirements Engineering Area", part of the RE@Brazil program. It summarizes a session that promotes discussion and knowledge exchange between professionals from academy and industry. Five guest panelists were carefully chosen to provide a wide range of perspectives on the area of Requirements Engineering. Among the panelists are four experienced industry professionals, and a moderator – a full professor - leader in the field of Requirements Engineering in Brazil.

Keywords. Requirements Engineering Practice, Industry, Challenges, Success

1 Introdução

Este artigo apresenta uma breve descrição do painel: "**Práticas e Desafios da Indústria Brasileira na área de Engenharia de Requisitos**" que faz parte do programa do RE@Brasil. Ele resume uma sessão que promove a discussão e troca de conhecimento entre diferentes profissionais da acadêmia e da indústria. Cinco painelistas foram cuidadosamente escolhidos para proporcionar uma ampla gama de perspectivas sobre área de Engenharia de Requisitos. Entre eles estão quatro profissionais experientes da indústria, e um moderador - professor pesquisador - líder da área de Engenharia de Requisitos a nível de Brasil.

O painel promove uma oportunidade para que a comunidade de requisitos, que participa do RE@Brasil, presencie a transmissão da diferentes experiências e participe da discussão relacionada as práticas chave aplicadas atualmente na indústria. Dessa forma o painel propocia a geração de uma síntese representativa do estado da área e seus desafios. Como consequência, tem-se um grande estímulo para aumentar a sinergia entre membros da comunidade brasileira na área, essencial para a geração de contribuições relevantes e projetos inovadores.

2 Participantes do Painel

O painel contempla duas empresas privadas (Globo.com e T&M) e duas empresas públicas (SERPRO e DATAPREV), além de um Professor Pesquisador líder na área – Prof. Dr. Júlio Leite – moderador do painel. Um resumo das empresas e panelistas é descrito na Figura 1. Detalhes sobre cada empresa participante, assim como a biografia de cada panelista, são apresentados nas subseções a seguir.

Empresa	**Representante**	**Função**	**Participação no Painel**
SERPRO	Helena C. Bastos	Gerente de Projeto	Painelista
GLOBO.COM	Igor Macaubas	Gestor de Programas, Agile Coach, e Gestor Técnico da área de Webmedia	Painelista
T&M	Martin Tornquist	Diretor da T&M	Panielista
DATAPREV	Karolyne Oliveira	Analista de TI	Painelista
PUC-RIO	Júlio Leite	Professor Pesquisador	Moderador

Figura 1 Empresas participantes do Painel

2.1 Serviço Federal de Processamento de Dados (SERPRO)

O **SERPRO** é uma empresa pública, vinculada ao Ministério da Fazenda, que presta serviços em Tecnologia da Informação e Comunicações para o setor público. A empresa é considerada uma das maiores organizações do setor na América Latina. O SERPRO desenvolve programas e serviços para um maior controle e transparência sobre a receita e os gastos públicos; entre eles os sistemas de declaração do Imposto de Renda, da Carteira Nacional de Habilitação e do Passaporte Brasileiro. A empresa investe no desenvolvimento de soluções tecnológicas em Software Livre e desenvolve projetos e programas que contemplem as questões sociais de acessibilidade e inclusão digital.

A representação do SERPRO no painel é feita por **Helena Cristina Bastos,** Gerente de Desenvolvimento de Sistemas que atende a Receita Federal desde 1988. Helena é especialista em Gestão da Tecnologia da Informação e mestre em Ciência da Computação, ambos pela UFPE.

2.2 Globo.com

A **Globo.com** é o braço de internet das Organizações Globo, o maior conglomerado de mídia da América Latina. Baseada no Rio de Janeiro, a Globo.com desenha, desenvolve e mantém os maiores portais verticais da internet Brasileira, todos líderes em suas categorias - O portal G1, líder em notícias, Globoesporte.com, em esportes, Globo.tv em vídeos, e Ego em notícias de celebridades. A empresa fornece ainda consultoria estratégica e

suporte tecnológico a todas as demais empresas das Organizações Globo. É reconhecida pela excelência em lidar com volumes altíssimos de tráfego, transmissões de vídeo ao vivo de larga escala, e é responsável pela maior audiência de vídeos online do país.

A representação da Globo.com no painel é feita por **Igor Macaubas**, que possui mais de 12 anos de experiência na área de Gestão de Projetos. Igor atua como Gestor de Programas e Agile Coach na Globo.com, também acumula a Gestão Técnica da área de Webmedia, responsável pela plataforma de vídeos da Globo.com. Há 4 anos na Globo.com, já liderou diversos projetos de grande destaque, como a migração de plataforma do portal G1, GloboNews e Ego, e mais recentemente nos projetos de distribuição de vídeos que irão suportar a copa do mundo de 2014 e olimpíadas de 2016.

2.3 T&M Testes de Software

A **T&M Testes de Software** tem 30 anos de mercado, e é líder no fornecimento de serviços de testes com ênfase em inovação. AT&M está alinhada com os principais modelos mundiais de maturidade em testes como o TMMI (*Testing Maturity Model Integrated*) e o TPI (*Test Process Improvement*). Seu foco é nas causas que limitam o desempenho com instrumental TOC (*Theory Of Constraints*), 6σ (*Six Sigma*), *Test Based Requirements*, *Metrics and Risk Based Testing & Automation Testing*. Seus processos de testes utilizam os mais conceituados métodos de automação baseado em *Data Driven Test, Action Based Test e Model Based Test*, produzindo testes mais abrangentes, mais baratos e melhores do que a automação tradicional.

A representação da T&M no painel é feita por **Martin Tornquist.** Martin é mestre em Ciência da Computação pela UFRGS, onde foi professor no Instituto de Informática e no Curso de Pós-graduação em Ciência da Computação por mais de 20 anos. Fundou e é diretor do grupo T&M. Martin é criador da base de conhecimento ATC (Analista de Testes Certificado) do IBQTS (Instituto Brasileiro de Testes e Qualidade de Sofware). Recentemente, ele traduziu para o português o livro "Fundamentos da Engenharia de Requisitos - um Guia de Estudo para a Certificação Profissional como Engenheiro de Requisitos pelo modelo CPRE-FL do IREB (International Requirements Engineering Board)".

2.4 DATAPREV

A **DATAPREV** é uma empresa pública vinculada ao Ministério da Previdência Social. A empresa desenvolveu sofisticados sistemas capazes de armazenar, processar e atualizar, em tempo real, as informações de milhões de contribuintes brasileiros. A empresa presta serviços para o Instituto Nacional do Seguro Social (INSS), a Receita Federal do Brasil e os Ministérios da Previdência Social, do Trabalho e Emprego e do Desenvolvimento Social e Combate à Fome. A DATAPREV desenvolve e mantém sistemas de informação em diversas plataformas tecnológicas, presta serviços de consultoria em todas as áreas da Tecnologia da Informação e Comunicações (TIC), e oferece serviços de datacenter e telecomunicações.

A representação da DATAPREV no painel é feita por **Karolyne Oliveira**. Karolyne trabalha na empresa há mais de cinco anos, onde já atuou no desenvolvimento de sistemas, análise de negócio e qualidade de software. Trabalhou como programadora na empresa NEUS e foi professora no IESP-PB. Karolyne está finalizando o seu doutorado na UFPE, onde é integrante do grupo de Engenharia de Requisitos (LER). Realizou estágio de doutorado-sanduíche, durante um ano, na Universidade Politécnica de Valência.

2.5 Julio Leite, PUC-Rio

Julio Leite um dos fundadores da área de Engenharia de Requisitos e da Sociedade Brasileira de Computação (SBC). Anteriormente, ele ocupou os cargos de professor visitante da Universidade de São Paulo (ICMC) e da Universität Kaiserslautern, de cientista visitante do Fraunhofer IESE, de professor convidado da ECI da Universidad de Buenos Aires, de visiting scholar na University of Toronto e de cientista visitante do Projeto Lucretius - Università degli Studi di Trento. Julio é Cientista do Nosso Estado (1999, 2007, e 2009) e tem participação ativa na área de Engenharia de Requisitos: é membro do Working Group 2.9 (Software Requirements Engineering) da IFIP (International Federation for Information Processing), é co-editor do livro Perspectives on Software Requirements, co-fundador das séries WER (Workshop em Engenharia de Requisitos) e FEES (Fórum de Educação em Engenharia de Software), e autor do "Livro Vivo: Engenharia de Requisitos". Este ano Julio é o organizador geral da 21ª Conferência Internacional de Requisitos (RE´13), realizada no Rio de Janeiro.

Introducing Variability in a Client-Oriented Requirements Engineering Process

Graciela D. S. Hadad[1], Jorge H. Doorn[1,2]

[1]DIIT, Universidad Nacional de La Matanza, Argentina
[2] Fac. Ciencias Exactas, Universidad Nacional del Centro de la Provincia de Buenos Aires

ghadad@ing.unlam.edu.ar,jdoorn@exa.unicen.edu.ar

Abstract. It is a good practice to consider the adaptation of any process to particular situations. This applies to Requirements Engineering where the requirements production process becomes adaptable to particular situations. Indicators depicting these situations should be created in order to guide the choice of the requirements process that best suits each specific case. Almost all literature offers a fixed requirements process independent of the context where it will be carried out. Top-down and bottom-up approaches are the most widespread ones, though middle-out approaches or combinations of them sometimes provide more accurate solutions. Our present research project "Adaptability and Completeness in Requirements Process" is centered on improving a well-developed requirements process based on natural language models (a glossary, scenarios and requirement specifications). The idea is to establish the factors that may influence a software project and the adaptations that may be introduced in the requirements process.

Keywords. Requirements process, process variability, situational factors, natural language model.

1 Introduction

Since problem domain knowledge is mostly expressed in natural language, the use of a Requirements Engineering (RE) approach based on natural language representations increases the probability of success of any software project. Natural language models, such as glossaries, use cases and scenarios, promote stakeholders communication and aid in validating requirements. Thus, an RE strategy using natural language models is considered to be client-oriented.

Throughout many research projects since 1995, our group has participated in the definition of a client-oriented RE strategy based on scenarios [1], which has been refined and tested in several organizations, achieving high-quality results. Several real-world applications were developed during this period. Two examples are: i) an integrated complex Student Administration system designed for a university from 2009 to 2010 (109 lexicon terms, 300 current scenarios, 409 future scenarios and 363 high-level requirements), and ii) an ATM Acquisition and Management system corre-

sponding to a full software development put into service in a company on 2003 (55 lexicon symbols, 20 current scenarios, 38 future scenarios and 167 requirements). During the implementation of the strategy in these organizations, sometimes ad-hoc adjustments were required to achieve the objectives of the project in compliance with deadlines and other constraints.

We believe that planning adaptations will improve the overall RE process. Therefore, the different situations that may attempt against a successful requirements process should be identified. Some activities of the RE process are performed in the same manner regardless of situational factors, while other activities are altered, removed or replaced. That is, the process can be assembled like a flexible puzzle using some pieces depending on the situational factors identified at a start point.

Situational Method Engineering, as a sub-area of Method Engineering, can help on this matter since it is advocated to build methods tailored to specific situations for the development of software.

The adaptation of the process is based on indicators describing the situation. Part of the task is to compose such indicators based on observable factors, such as the degree of business processes reengineering, the prior knowledge about the application domain, the domain complexity and the project size, among others. Those situational factors should be taken into account before executing a requirements production process. In practice, these factors are usually not taken into account at the beginning of the project, and some are regarded only during the course of the project. We believe it could be easier to get better requirements solutions following a process that addresses the particular factors surrounding the project, although literature frequently offers unique ways to produce requirements. In summary, our research project is dealing with variability in an RE process driven by the adaptation of the process to situational factors.

2 Objectives of the Research

Our research project aims to improve a well-developed client-oriented RE strategy by establishing a process adaptable to different situational factors. This requires defining which the models to be produced, the process variation points and the activities and techniques to be applied at each process phase. The process will be defined as a set of modular components, where each component will define an activity, the inputs and outputs, and the techniques to be used. It is very important to understand the alternatives adopted to implement an instance of the requirements process.

The specific objectives of the research are the following:

- Identify the different situational factors influencing the RE strategy.
- Define the variation points of the strategy according to the situational factors.
- Define all the components of the process related to the basic strategy and to the alternative instances of the strategy. This includes the identification of commonality and variability of the requirements process.
- Develop a heuristic to guide the configuration of the RE process based on predefined situations.

Therefore, the main idea is to provide an RE process, driven by narrative scenarios, which is configured according to specific project and domain characteristics. This type of research provides a rational base to define more agile requirements processes when it is suitable. At the current stage, these objectives have been partially accomplished since fourteen factors have been already identified and five variation points were defined (see next section), though they still require confirmation.

3 Scientific Contributions

The client-oriented RE strategy, depicted in Figure 1(a), involves all the requirements engineering activities: elicitation, modeling, analysis and management, and consists basically in:

- Understanding the vocabulary used in the application domain, supported by the Language Extended Lexicon model [2];
- Understanding the application domain, supported by a set of current scenarios that represent the situations observed in the domain [3];
- Defining the software system context, by producing a set of future scenarios that represent situations envisioned in a future application domain where the software system will be operating [4]; and
- Making explicit the requirements, by producing a Software Requirements Specification (SRS) where the requirements are clearly individualized from the set of future scenarios [5].

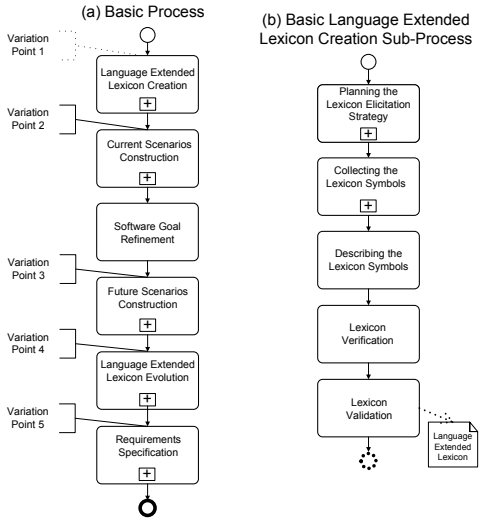


Fig. 1. Basic client-oriented RE strategy using Business Process Modeling Notation (BPMN)

Figure 1(a) presents the basic RE process independent of situational factors, while Figure 1(b) depicts one of the sub-processes, also showing its basic structure. Though both figures present a sequential flow, there are recycles due to verification and

validation activities, and the continuous improvement in understanding the problem. Furthermore, the RE strategy may be applied under an iterative and incremental process, or a spiral one [4]. At the first step of the research, a list of factors was produced, and variation points were identified in the basic process, as shown in Figure 1(a). Two types of situational factors were considered: those related to the specific application domain and those related to the specific software project. The former involves situations in the client context, while the latter takes into account the developer context. The initial factors identified are:

Context Factors	**Project Factors**
- Complexity of the Problem	- Familiarity with the Domain
- Level of Business Innovation	- Size of the Project
- New Business	- Level of Developers Rotation
- Domain Volatility	- Quality Criterion
- Target Customer	- Required Artifacts Reuse
- Level of Users Rotation	- Support for Traceability
- Level of Conflict in the Domain	- Contractual Obligation to produce an SRS

Each factor was assigned a set of possible values, such as: Problem Complexity = {low, medium, high}; New Business = {yes, no}; Target Customer = {market-driven, tailor-made}.

Besides, each factor was studied to establish its influence at the variation points, considering that a situational factor may affect the process at one or more variation points. Variation Point 1 depends on the business novelty, the degree of software customization for the market or a specific client, the familiarity of the requirements engineer team with the domain, the demand of artifacts reuse, and the level of required quality. Variation Point 2 strongly depends on the requirements engineers' knowledge about the application domain, though it is also affected by the complexity of the problem, the size of the project and the volatility of the domain, among some other factors. Variation Point 3 is almost influenced by every identified factor, though the expected level of business innovation is here the main influential factor. Variation Point 4 depends mainly on the degree of changes in the terminology used in future scenarios in relation with the vocabulary used in the application domain. Variation Point 5 is subjected to a contractual obligation to produce an SRS, and also influenced by the project's size and the need of traceability anchored on individual requirements.

As stated above, the configuration process depends on the combination of several situational factors at each variation point, and this combination may have three possible effects on the process: i) the process is performed or not; ii) the process is performed partially; or iii) the activities of the process may be done applying different techniques. The pre-defined combination of factors affecting each point will determine which of these three decisions should be taken. For example, Figure 2(a) shows a situation where the entire process is skipped (not creating the lexicon), and two complementary situations that branch off into different sub-processes (planning the lexicon elicitation) and different techniques (choosing the lexicon validation technique).

Figure 2(b) shows how the instance of a process should be constructed according to specific situational factors by skipping the whole process, by selecting the

corresponding components of Planning Lexicon Elicitation Strategy and Lexicon Validation, and by selecting a specific verification technique within the Lexicon Verification component depending on the quality criterion factor.

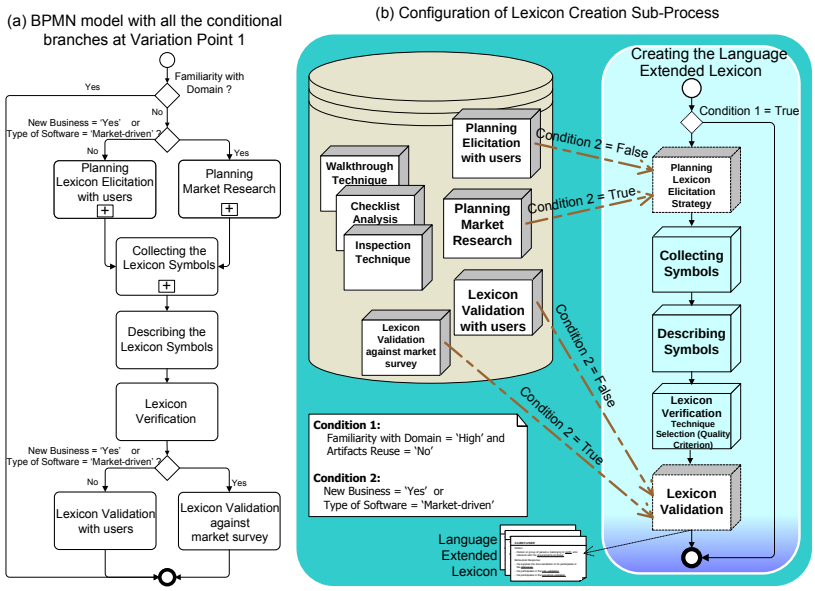


Fig. 2. Lexicon Creation Sub-Process including variants

Regarding Variation Point 3, there may be five different approaches to managing the construction of future scenarios: i) Construct Future Scenarios adapting the existing current scenarios: procedure-oriented approach; ii) Construct Future Scenarios disregarding current business processes, but following software goals supported by existing high-level current scenarios; iii) Construct Future Scenarios by a combination of the two previous approaches: hybrid approach; iv) Construct Future Scenarios using the Language Extended Lexicon derivation technique; and v) Construct Future Scenarios following a top-down approach building high-level abstract scenarios according to software goals. It should be remarked that though five ways of the future scenarios construction process were identified, many sub-processes are common to several of them. At least one real-world experience following every one of these five variants was carried out. Another issue to state is that the different branches derived at Variation Point N-1 and at Variation Point N may not be independent since not every combination is feasible.

4 Conclusions

The basic client-oriented requirements engineering strategy was built based on the experience gained after years of its implementation in the academic and professional practice. Experience has taught us that complex problems have distinctive features that must be taken into account to carry out a successful requirements process.

Requirements engineers must tailor the requirements production process by selecting the techniques that are more suitable for the specific situation. Our proposal aims to help requirements engineers to tailor a client-oriented RE process when the knowledge of certain circumstances surrounding the software project is available in the beginning. The adaptation of the requirements process in the cases we have participated has contributed to the acceptance of the process in host organizations. This should be confirmed by replicating cases with different approaches.

5 Ongoing and Future Work

The variation points were precisely defined since we have acquired enough knowledge about the basic RE process after putting it into practice in more than 200 real cases. These practices have allowed us to recognize the different possible variants of the process and the modular components required. The list of situational factors is quite exhaustive, although it is likely to be extended. Some factors not mentioned here are being evaluated. Several instances of the RE process have been applied in real cases, although they have not been formally defined prior to their application.

Most of the process components have been tested in the field, and in degree and post-degree courses, during the last 10 years. Some remaining components have been defined theoretically based on our experience and some have already been applied in case studies. Additional tests of the different process variants will be carried out to check the effectiveness of the proposed branches. This will allow identifying further commonality and variability. The main idea is to test and confront results against the formulated process variants. The presence of many factors makes nearly impossible to validate every single combination of them, considering the different possible values associated to each factor. This should not impede the exploratory research of the most likely combinations.

In next steps of the research, missing process components will be defined and a heuristic to build a particular process for a specific situation will be developed.

References

1. Leite, J.C.S.P., Doorn, J.H., Kaplan, G.N., Hadad, G.D.S., Ridao, M.N.: Defining System Context using Scenarios. In: Perspectives on Software Requirements, Kluwer Academic Publishers, USA, ISBN: 1-4020-7625-8, chapter 8, pp.169-199 (2004)
2. Hadad, G.D.S., Doorn, J.H., Kaplan, G.N.: Creating Software System Context Glossaries. In: Encyclopedia of Information Science and Technology, IGI Global, Mehdi Khosrow-Pour (ed), Information Science Reference, 2ºed., Vol.II, pp.789-794, EEUU (2008)
3. Leite, J.C.S.P., Hadad, G.D.S., Doorn, J.H., Kaplan, G.N.: A Scenario Construction Process. In: Requirements Engineering Journal, Vol.5, N° 1, pp. 38-61 (2000)
4. Hadad, G.D.S.: Uso de Escenarios en la Derivación de Software. Tesis Doctoral, Facultad de Ciencias Exactas, Universidad Nacional de La Plata, Argentina (2008)
5. Hadad, G.D.S., Doorn, J.H., Kaplan, G.N.: Explicitar Requisitos del Software usando Escenarios. In: 12th Workshop on Requirements Engineering (WER'09), pp. 63-74 (2009)

Towards Requirements Engineering Process for Embedded Systems

Luiz Eduardo Galvão Martins[1], Jaime Cazuhiro Ossada[2], Anderson Belgamo[3]

[1]Universidade Federal de São Paulo (UNIFESP), São José dos Campos, Brazil
legmartins@unifesp.br
[2]Faculdade de Tecnologia de Indaiatuba (FATEC), Indaiatuba, Brazil
jaime.ossada@fatec.sp.gov.br
[3]Universidade Metodista de Piracicaba (UNIMEP), Piracicaba, Brazil
anbelgamo@unimep.br

Abstract. This paper presents an overview about a Brazilian research towards requirements engineering process for embedded systems. The scientific contributions reported throughout the paper are concerned to templates, guidelines and tools developed during the last four years. These artifacts can help to narrow the existing gap between hardware and software embedded system teams. We brifely describe two requirements specification templates, named TERASE and CAMA, and one requirements elicitation guide, named GERSE which is supported by a software tool called Zaki.

Keywords: Embedded Systems, Requirements Process, Requirements Template and Guidance

1. Introduction

In the last two decades the industry of consumer goods has witnessed a huge increasing of Embedded Systems (ES) applications. This kind of system has been widely used, such as in the areas of entertainment products, medical devices, automotive systems, avionics, industry process control, telecommunication devices and others. Currently it is very rare to find any electronic device that is not controlled by ES.

ES is a computing system specially designed to monitor or control a physical device. This computing system usually receives input data from sensors, it processes these data using microcontrollers or microprocessors and sends the results to user interfaces and/or actuators [3]. Software developed to ES, named firmware, is highly coupled to the hardware designed to the controlled devices.

This paper reports research efforts focused to improve the ES development particularly addressing the issues related to requirements elicitation and specification of embedded systems.

This paper is organized as follows: in the section two the objectives of the research are commented; in the section three the main scientific contributions are presented; section four concludes the paper; and finally in section five ongoing and future works are pointed out

2. Objectives of the Research

The general objective of the research presented in this paper is to propose a requirements engineering process for embedded systems which will improve the development of such systems and their general quality specially diminishing the existing gap between hardware and software ES teams. In order to achieve this objective the following goals should be developed:

- To identify the main flow of embedded systems development process and to map the relevant process aspects that influence the ES requirements definition.
- To build an ES conceptual model to manage stakeholders, system environment, high level requirements, embedded software requirements, hardware requirements, and communication interface requirements.
- To build guidelines and templates to support ES requirements elicitation, modeling, specification and validation.
- To build integrated software environment to support the use of proposed guidelines and templates.

3. Scientific Contributions

The research reported in this paper has started in 2009. The first stage of the research was to identify how ES practitioners in Brazil were approaching requirements during the ES development process. The goal of this field research was to know the state of practice of requirements engineering for ES in Brazil. Professionals that have worked in several industrial segments developing ES were invited (53 ES practitioners answered a questionnaire), the most of them were professionals working in industries in São Paulo state, and the main segments covered in this research were: automotive systems, industrial automation, home appliance, domotics, medical devices, telecommunication and entertainment [5].

Some results from this field research are showed as follows: (i) the most part of the ES practitioners have education on engineering courses (67%) and just 33% have education on computer science courses; (ii) 54.9% revealed that they did not use any organized methodology to elicit requirements; (iii) 41.2% of the ES practitioners that used some requirements elicitation methodology stated that the adopted requirement procedures were not stable inside the organization; (iv) the most cited requirement elicitation techniques were: existing documentation analysis (26%), interview (19%), e-mail exchanging (18%), market analysis (17%), questionnaire (15%) and JAD – Joint Application Development (2%).

It was possible to identify the lack of templates and guidelines that addressed ES practitioner's necessities concerned to requirements elicitation and specification based on the field research results. In order to address these necessities we have developed the following artifacts:

- TERASE: template for environmental requirements specification of embedded systems (TERASE is a Portuguese acronym to "Template para Especificação de Requisitos de Ambiente em Sistemas Embarcados");
- CAMA: template for requirements specification of communication interface among ES physical components based on CAN protocol (acronym is formed by the words CAN, Martins and Almudi);
- GERSE: requirements elicitation guide for ES (GERSE is a Portuguese acronym to "Guia de Elicitação de Requisitos para Sistemas Embarcados");
- ZAKI: software tool to support GERSE activities.

In the next sections some details about each proposed artifact are introduced and briefly commented.

3.1 TERASE

The goal of TERASE template is contribute to improve the ES requirements specification particularly focused to the environmental requirements of the ES [4]. Environmental requirements are classified as non-functional requirements and they should be specified according to physical features where the ES will be deployed.

Software development teams usually have to build software according to hardware specifications defined by hardware engineers, software teams need detailed information about environmental variables and physical devices that will capture and control the system. TERASE works as facilitator to improve the communication between hardware and software teams. A complete requirements specification to describe the ES physical environment must include:

- Environmental variables
- Input devices (sensors)
- Output devices (actuators and users interface devices)
- Microcontrollers

For each one of these elements it was proposed a requirement specification card. The cards help the hardware team to record the necessary information to the software team supporting the ES requirements elicitation and specification process.

3.2 CAMA

In the construction of embedded systems, the software starts being developed when the hardware is already in a very advanced stage of development. The hardware design tends to be dominant due to having a major cycle of development, being more

stable and requiring logistical dependence on external partners, such as suppliers and outsourced developers. There is not, so far, an appropriate methodology to help developers to specify the requirements to automotive embedded systems, causing a large gap between designers from hardware and software areas, especially in the early phases of structuring the design [8]. One of the possibilities that CAMA Template offers is the integration between developers through a resource that provides easy communication channels between the hardware designers and software engineers.

CAMA Template is used to specify the relevant and special aspects of the automotive embedded communication systems network that use CAN protocol in exchanging information [6]. The template is structured in Figure 1. The ES features managed by CAMA are organized using specification cards.

An automotive embedded system was chosen for the study case to have its requirements specified through the proposed template. The study was based on a finished specification from a major international car manufacturer with a plant in Brazil, with a large insertion in automotive Brazilian market.

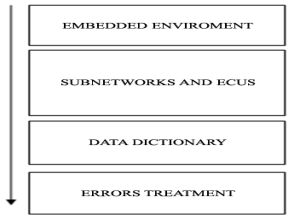


Fig. 1. – CAMA Template's General Structure

3.3 GERSE

The main goal of the proposed guide is to help ES engineers during the requirements elicitation process. GERSE leads ES engineers during the elicitation process offering a set of activities that addresses the ES main features [7]. Using GERSE, ES engineers can manage the requirements elicitation process in an organized way. The proposed guide helps the requirements definition allowing its complete specification for products based on embedded technology.

GERSE is divided into two phases, named pre-phase and main phase, which are organized in seven categories. These categories are organized in 46 activities, which are responsible to generate the artifacts that will compose the ES requirements. Each activity produces at least one artifact that can be both a document describing a specific feature of the product or a diagram modeling any specific feature. The pre-phase activities will help the ES engineers to make the transition from the high level requirements to technical requirements. Figure 2 shows a GERSE overview presenting the categories proposed to each phase.

GERSE documentation were sent to four ES engineers to evaluate the proposed guide, the evaluation was performed through survey. The ES engineers expertise was in automotive systems, medical devices and entertainment areas. All ES engineers

evaluated GERSE as a useful guide for ES requirements elicitation stating that such guide is easy to use and contributes to increase the ES development quality.

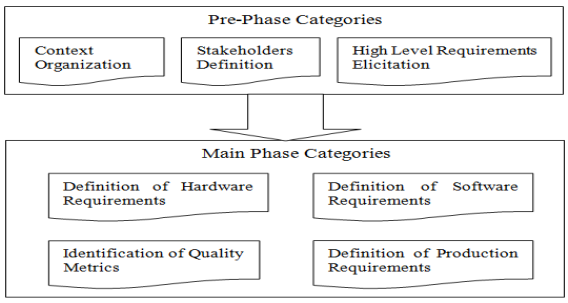


Fig. 2. Phases and categories supported by GERSE.

3.4 ZAKI

The adoption of any software process can be facilitated by the use of computer support. In this sense, a tool called Zaki was developed to support GERSE activities and the requirements elicitation process for embedded systems.

Zaki tool is divided into two modules, according to GERSE phases - pre-phase and main phase - supporting activities like requirements elicitation, analysis and management for embedded systems. Zaki tool was developed using .NET platform - C# language- and the SQL Server database.

During the pre-phase, Zaki tool supports functionalities related to manage information about project guidelines and main product features, development organizational impact and target audience. During the main phase, Zaki tool is divided into three modules: Definition of Hardware Requirements, Definition of Software Requirements, and Identification of Quality Metrics.

4. Conclusions

The communication problem between hardware and software teams is a great challenge to be overcome in the context of ES development [1][2]. Software engineers need to receive a precise and complete requirements specification covering all aspects of hardware and environment where ES will be executed. A significant part of such specification is built by hardware engineers. Failures and mistakes in the communication process between hardware and software teams have strong impact in the final cost and schedule as well as in the general quality of the system.

In this paper we presented some research results that addressed the communication problem during ES development. These results consisted of two requirements specification templates – called TERASE and CAMA - and one requirements elicita-

tion guide – called GERSE – both designed to improve the communication among ES stakeholders. GERSE is supported by a software tool called Zaki.

5. Ongoing and Future Work

Currently this research group is working to improve Zaki tool to support all GERSE activities as well as adjusting some GERSE activities to make them easier to be used by requirements engineering teams in the ES context. GERSE is also being tested in the context of critical embedded systems specifically for an embedded system to control a medical device (insulin infusion pump). The main future works are listed as follows:

- To test GERSE in larger ES project;
- To integrate TERASE, CAMA and GERSE in a same software environment;
- To test and adapt the proposed artifacts in the context of robotics systems;
- To improve the templates and guidelines proposed to support requirements specifications for adaptive embedded systems;
- To start research efforts focused to requirements gathering in the context of cyber-physical systems.

References

1. Graaf, B.; Lormans, M. and Toetenel, H.: Embedded Software Engineering: The State of the Practice. IEEE Software (2003)
2. Liggesmeyer, P. and Trapp, M.: Trends in Embedded Software Engineering. IEEE Software, pp. 19 – 25 (2009)
3. Broy, M.: Requirements Engineering for Embedded Systems. In: Workshop on Formal Design of Safety Critical Embedded Systems (FemSys). Munich, Germany (1998)
4. Martins, L. E. G. ; Souza Jr., R. ; Oliveira Jr., H. P. ; Peixoto, C. S. A.: TERASE: Template para Especificação de Requisitos de Ambiente em Sistemas Embarcados. In: 13th Workshop on Requirements Engineering (WER). pp. 50-61, Cuenca (2010)
5. Ossada, J. C.; Martins, L. E. G.: Um Estudo de Campo sobre o Estado da Prática da Elicitação de Requisitos em Sistemas Embarcados. In: 13th Workshop on Requirements Engineering (WER). pp. 30-41, Cuenca (2010)
6. Almudi Neto, D; Martins, L. E. G.: A Requirements Specification Template of a Communication Network Based on CAN Protocol to Automotive Embedded Systems. Journal of Computer Science and Technology, v. 10, pp. 143-149, La Plata (2010)
7. Ossada, J. C.; Martins, L. E. G.; Belgamo, A.; Ranieri, B. S.: GERSE: Guia de Elicitação de Requisitos para Sistemas Embarcados. In: 15th Workshop on Requirements Engineering (WER). pp. 57-70, Buenos Aires (2012)
8. Post, A.; Menzel, I.; Hoenicke, J. and Podelski, A.: Automotive Behavioral Requirements Expressed in a Specification Pattern System: a Case Study at BOSCH, Requirements Engineering Journal, vol. 17, pp. 19-33. Springer-Verlag (2012)

Transformação de um Modelo de Empresa em Requisitos de Software

Fábio Levy Siqueira[1] and Paulo Sérgio Muniz Silva[2]

[1] Programa de Educação Continuada da Poli-USP, São Paulo, Brazil
[2] Escola Politécnica da Universidade de São Paulo, São Paulo, Brazil
fabio@levysiqueira.com.br, paulo.muniz@poli.usp.br

Abstract. The transformation from stakeholder requirements into system and software requirements is traditionally executed manually, through a refinement process. This work presents ongoing research to provide automatic transformation from stakeholder requirements into software requirements in the context of enterprise systems. The proposed transformation uses an enterprise model as a stakeholder requirements model. This paper describes the obtained and intended scientific contributions, highlighting the tool we developed. In addition, it presents current and future works.

Resumo. A transformação de requisitos das partes envolvidas em requisitos de sistema e de software é tradicionalmente realizada manualmente, através de um processo de refinamento. Este trabalho apresenta a pesquisa realizada para automatizar essa atividade, permitindo a transformação automática de requisitos das partes envolvidas em requisitos de software no contexto de sistemas empresariais. Para isso, é usado um modelo da empresa como modelo de requisitos das partes envolvidas. O artigo apresenta as contribuições científicas pretendidas e obtidas até o momento, destacando a ferramenta criada. Também são apresentados os trabalhos em andamento e futuros.

Palavras-chave: requisitos, modelo de empresa, transformação, caso de uso.

1 Introdução

Os requisitos das partes envolvidas são obtidos a partir da análise das necessidades, metas e objetivos das partes envolvidas. Eles representam a "interação pretendida que o sistema terá com o seu ambiente operacional e que será a referência contra a qual cada serviço operacional resultante será validado" [1, p.19]. Uma das atribuições do engenheiro de requisitos é transformar os requisitos das partes envolvidas em requisitos de sistema, que representam uma especificação técnica para o sistema, do ponto de vista do fornecedor [1]. Essa transformação deve considerar diversas informações, como restrições, atributos de qualidade e fatos assumidos, cabendo ao engenheiro de requisitos analisar alternativas ao ponderar sobre requisitos conflitantes, prioridades e riscos.

Tradicionalmente essa transformação é feita através de um refinamento manual. Buscando apoiar o trabalho do engenheiro de requisitos, este trabalho apresenta a proposta de realizar uma transformação automática ou semiautomática de um modelo de requisitos das partes envolvidas em requisitos de software, considerando sistemas intensivos de software. Essa transformação considera sistemas empresariais, usando um modelo de empresa como representação dos requisitos das partes envolvidas. Nesse contexto, o software é apenas mais um elemento da empresa. Dessa forma, um modelo empresarial permite representar o software ao descrever seus efeitos sobre os outros elementos do ambiente empresarial - mas sem descrever detalhes de sua implementação. Além disso, esse modelo possui conhecimento do domínio que pode ser usado para a transformação desses requisitos em requisitos de software.

Este artigo está organizado da seguinte forma: na Seção 2 é apresentado o objetivo da pesquisa e o escopo. Em seguida, na Seção 3 são apresentadas as contribuições da pesquisa, evidenciando a ferramenta criada. Por fim são apresentadas as conclusões, na Seção 4, e os trabalhos futuros e em andamento na Seção 5.

2 Objetivo da Pesquisa

O objetivo da pesquisa é permitir a transformação de um modelo de requisitos das partes envolvidas, usando um modelo de empresa, em requisitos de software. Essa transformação deve ser realizada de forma semiautomática (ou mesmo automática), com a mínima intervenção de um engenheiro de requisitos. Além de facilitar as atividades de Engenharia de Requisitos, essa transformação busca evitar erros na especificação dos requisitos – em especial o desenvolvimento de um sistema incompatível com o ambiente de negócio.

Apesar de a proposta de transformação ser conceitual, podendo ser aplicada a diferentes escopos e diferentes modelos de empresa e de requisitos de software, para que ela seja aplicada é necessário instanciá-la. Dessa forma, o escopo foi limitado a sistemas de automação de processos de negócio, uma vez que as características desse tipo de sistema permitem simplificar a transformação, seja ao facilitar a criação das regras de transformação, ou mesmo ao evidenciar a presença dos requisitos em um modelo da empresa. Por outro lado, decidiu-se usar um modelo textual de caso de uso como modelo dos requisitos de software. A escolha dessa representação foi devido à popularidade da técnica caso de uso.

3 Contribuições Científicas

A principal contribuição do estudo da transformação de um modelo de empresa em requisitos de software é ajudar no entendimento da relação entre os diferentes níveis de requisitos. Os requisitos das partes envolvidas são tradicionalmente refinados manualmente em requisitos de software, porém essa transformação nem sempre é clara e depende do conhecimento e das habilidades do engenheiro de requisitos.

Uma vez que um modelo de empresa é usado como modelo dos requisitos das partes envolvidas, esse estudo também contribui com o entendimento da relação entre

o contexto empresarial e os requisitos. As vantagens e desvantagens do uso de um modelo de empresa para esse fim são discutidas em [5]. Essa possibilidade é também analisada empiricamente em [6], em que se realizou um quasi-experimento com alunos de pós-graduação para avaliar a qualidade e o tempo despendido para criar casos de uso a partir de uma representação textual dos requisitos das partes envolvidas ou de um modelo de empresa. O resultado desse experimento indica que a qualidade média do caso de uso criado a partir do modelo de empresa foi igual ou maior à qualidade média do caso de uso criado usando a representação textual. Além disso, o tempo médio despendido para criar o caso de uso empregando um modelo de empresa foi menor ou igual ao tempo médio para criar o caso de uso considerando a representação textual.

Do ponto de vista prático, o estudo dessa transformação contribui ao automatizar a geração de modelos de requisitos. O uso de conceitos de Engenharia Dirigida por Modelos para tratar da transformação de requisitos das partes envolvida em requisitos de software é discutido em [4]. Nesse trabalho são apresentadas regras propostas por outros trabalhos que podem ser usadas para esse fim. Essas regras são implementadas usando a linguagem de transformação *Operational QVT*, permitindo que elas possam ser usadas automaticamente. Existe também um outro conjunto de regras que foram criadas a partir da análise de um exemplo, as quais são apresentadas em [2]. A possibilidade de usar essa transformação foi analisada através de um outro quasi-experimento realizado com alunos de pós-graduação, descrito em [2]. O resultado desse experimento indica que não houve diferenças de qualidade do caso de uso gerado através da transformação, em comparação aos casos de uso gerados manualmente (diretamente ou considerando também um modelo de empresa).

Além das regras de transformação, uma outra contribuição dessa pesquisa é a definição dos metamodelos empregados. O metamodelo é um modelo que define a linguagem de modelagem a ser usada por um outro modelo. Definiu-se um metamodelo para o caso de uso e outro para a representação da empresa. O metamodelo de caso de uso foi baseado no resultado do *survey* dos elementos mais comuns de representações textuais de caso de uso, o qual é apresentado em [3]. Por outro lado, o metamodelo de empresa foi baseado em trabalhos da área de Organização e Métodos. Esse metamodelo é apresentado em [5], sendo discutido o emprego dele em um exemplo.

3.1 Ferramenta

Para permitir a transformação de um modelo de empresa em casos de uso, criou-se a ferramenta EMUCase (Enterprise Model to Use Case)[1], apresentada na Figura 1. Essa ferramenta representa uma versão inicial da transformação, considerando 22 regras de transformação, algumas definidas através da análise de trabalhos relacionados e outras definidas considerando um exemplo. A ferramenta permite a criação de modelos de empresa contendo as cinco visões definidas em [5]: processual, de estrutura organizacional, motivacional, de documentos e do ambiente físico. Um especialista (engenheiro de requisitos ou especialista do domínio) deve criar um modelo atual (*as-*

[1] A ferramenta está disponível em http://www.levysiqueira.com.br/projects/emucase.

is) e um modelo idealizado da empresa com o software (*to-be*). Cada um desses modelos pode conter diversos diagramas para cada uma dessas visões. Através de um assistente (*wizard*), a ferramenta verifica a consistência dos modelos de empresa criados e permite a transformação automática em um modelo de caso de uso em XML. A transformação permite que o modelo de empresa seja escrito tanto em português como em inglês, gerando os casos de uso na língua apropriada.

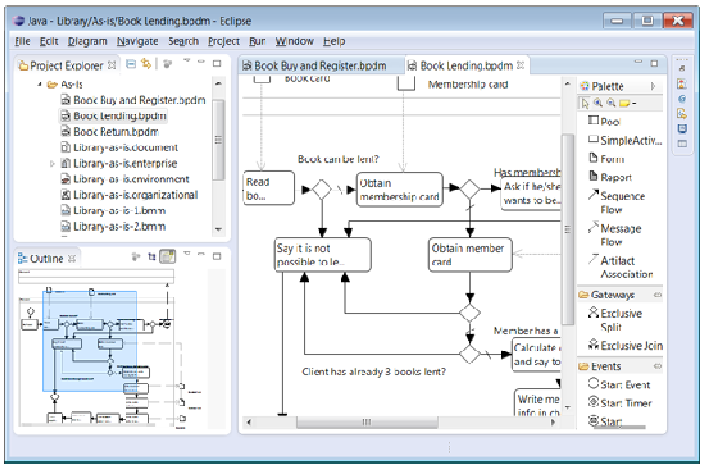


Fig. 1. A ferramenta EMUCase.

A ferramenta foi criada como um plug-in para o IDE Eclipse, seguindo os conceitos de Engenharia Dirigida por Modelos. Os metamodelos de empresa e de caso de uso foram implementados no Espaço Técnico (TS) do MOF, usando o *Eclipse Modeling Framework* (EMF) e o *Graphical Modeling Framework* (GMF). Para cada um dos metamodelos foi definida a sintaxe abstrata, a sintaxe concreta e a semântica. A transformação é feita através de uma transformação de metamodelos, ou seja, as regras definidas mapeiam elementos do metamodelo de empresa para elementos do metamodelo de caso de uso. Essas regras foram descritas usando o padrão *Operational QVT*.

4 Conclusões

Este trabalho apresentou a proposta de transformação de um modelo de empresa, usado como um modelo dos requisitos das partes envolvidas, em um modelo de requisitos de software. Foram discutidas as principais contribuições desse estudo, incluindo alguns dos resultados já obtidos. Também foi apresentada brevemente a ferramenta criada para apoiar essa transformação, a qual está disponível para o uso.

5 Trabalhos Futuros e em Andamento

Existem alguns trabalhos em andamento para melhorar a ferramenta e a transformação. Em relação à ferramenta, pretende-se permitir a intervenção de um engenheiro de requisitos para a tomada de algumas decisões que atualmente são executadas automaticamente. Além disso, planeja-se melhorar a representação textual do caso de uso, permitindo, por exemplo, o uso de diferentes metamodelos e o apoio na redação de casos de uso. Em relação à transformação, pretende-se definir mais regras de transformação, que seriam baseadas na análise de outros sistemas. Também se pretende replicar o experimento já executado para avaliar a transformação com um maior número de sujeitos. Também se pretende aplicar essa proposta em um projeto real.

Um outro trabalho futuro é usar essa infraestrutura com o objetivo de rastrear como mudanças no ambiente empresarial afetam os requisitos de software. Como a transformação produz um mapeamento entre os modelos de empresa e o de requisitos de software, pretende-se usar esse mapeamento para analisar quais partes do modelo de requisitos sofrem impactos com uma determinada alteração no contexto empresarial (como, por exemplo, a mudança de um processo de negócio). Essa análise pode ser realizada automaticamente, facilitando o trabalho de gerência de requisitos.

Por fim, um trabalho futuro é analisar a relação entre os modelos de empresa *as-is* e *to-be*. As metas consideradas e as decisões tomadas para gerar o modelo *to-be* poderiam ser usadas como base para a definição dos requisitos de software.

Referências

1. ISO: Systems and software engineering - Life cycle processes - Requirements engineering, ISO/IEC/IEEE 29148 (2011)
2. Siqueira, F.L.: Transformação de um Modelo de Empresa em um Modelo de Casos de Uso Seguindo os Conceitos de Engenharia Dirigida Por Modelos. Tese de Doutorado, Departamento de Computação e Sistemas Digitais, Escola Politécnica da Universidade de São Paulo, 267p (2011)
3. Siqueira, F.L. and Muniz Silva, P.S.: An Essential Textual Use Case Meta-model Based on an Analysis of Existing Proposals. In: Workshop on Requirements Engineering (WER), pp. 419-430 (2011)
4. Siqueira, F.L. and Muniz Silva, P.S.: Transforming an Enterprise Model into a Use Case Model Using Existing Heuristics. In: Model Driven Engineering Workshop (MoDRE), pp.21- 30 (2011)
5. Siqueira, F.L. and Muniz Silva, P.S.: Using an Enterprise Model as a Requirements Model in Process Automation Systems: A Proposal. In: International Conference on Information Systems and Technology Management (CONTECSI), pp.3064-3088 (2011)
6. Siqueira, F.L. and Muniz Silva, P.S.: Analyzing the Use of an Enterprise Model as a Stakeholder Requirements Model: An Experiment. In: Workshop on Requirements Engineering (WER) (2013)

Desafios de monitoração de requisitos não funcionais: avaliação em Transparência de Software

André Luiz de Castro Leal[1,2], Henrique Prado Sousa[1], Julio Cesar Sampaio do Prado Leite[1]

[1] Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)
Rio de Janeiro, RJ, Brasil

[2] Departamento de Matemática
Universidade Federal Rural do Rio de Janeiro (UFRRJ)
BR-465, Km 7, Seropédica, Rio de Janeiro - CEP 23890-000
andrecastr@gmail.com, hsousa@inf.puc-rio.br, www.inf.puc-rio.br/~julio

Resumo. Com a demanda da sociedade pela disponibilização da informação aberta, a Transparência passa a ser reconhecida como um requisito de potencial aplicação a software. Como requisito não funcional, a Transparência desafia engenheiros de software com suas características peculiares e ainda pouco pesquisadas. A partir dessas demandas, surge o questionamento referente às formas de avaliação da Transparência em software, tanto em sua arquitetura como em seu ambiente de execução e entrega da informação. O presente trabalho fomenta a discussão sob o olhar da avaliação de Transparência de software.

Keywords: Transparência de software, requisito não funcional, monitoração, sistemas multi-agentes.

1 Introdução

Uma sociedade madura, fortemente envolvida em suas rotinas com o uso de sistemas computacionais, almeja por acesso a todo e qualquer tipo de informação. Nesse sentido, a informação aberta é um tema central que evidencia um requisito não funcional, a Transparência [1].

Uma recente demonstração do movimento em torno informação explícita é a criação da Lei de Acesso, Lei nº 12.527, de 18 de novembro de 2011 [8]. A lei estabelece obrigações para os órgãos e entidades do poder público quanto à gestão da informação; define os tipos de informação que podem ser solicitadas; estabelece obrigações de divulgação espontânea de informações pelos órgãos da Administração Pública e medidas que devem ser adotadas para assegurar o acesso às informações.

É sabido que sistemas de informação estão implantados e dão suporte atualmente em quase todo domínio onde a informação é consumida. Diante do desafio da Transparência e da importância do software no suporte ao uso da informação, Engenheiros de Software passam a ter o desafio de lidar com dados abertos, usabilidade orientada ao cidadão, acessibilidade, apresentação de informação,

proveniência da informação e confiança na informação, que são características inseridas no contexto da Transparência a fim de prover o chamado Sistemas de Software Transparentes [6]. E esse desafio não ocorre apenas no uso do sistema para prover Transparência, mas também em sua própria arquitetura que pode ser transparente.

O Grupo de Pesquisa em Engenharia de Requisitos da PUC-Rio (Grupo ER - PUC-Rio) [1] desenvolve pesquisa sobre o tema Transparência a partir da definição de conceitos, estruturas, símbolos, termos, catálogos, sua aplicação em software [6] e modelos de negócio [7], criados de forma colaborativa, com o objetivo de sistematizar, a partir de critérios técnicos, ações, modelos, sistemas e definições sobre o tema. Dentro dessa realidade, há um recorte que particularmente nos interessa: a avaliação semi-automática ou automática da Transparência [2]. Ou seja, como conceber e implementar sistemas que possam tratar a monitoração de atributos pré-definidos em catálogo de Transparência de Software [1].

O artigo apresentará a perspectiva de pesquisa sobre o tema e será dividido em seções como segue: na seção dois serão apresentadas os objetivos da pesquisa; na seção 3, as contribuições científicas; a seção 4 apresentará as conclusões; e por fim, na seção 5 serão apresentadas as perspectivas de trabalhos futuros e o estado atual da pesquisa.

2 Objetivos da Pesquisa

O principal objetivo da pesquisa é focar a questão da avaliação de Transparência no contexto de software, que é uma questão importante a ser tratada uma vez que a avaliação de requisitos não funcionais não é algo que aceite valores exatos. Tornar software transparente requer incialmente avaliações para se entender quais pontos são passíveis de suportar tal qualidade [6].

O escopo inicial restringe-se em avaliar software em execução. Ou seja, avaliar traços da execução de um software a fim de compará-los com catálogo pré-definido para extrair pontos de conformidade e não conformidade entre o software e atributos pré-definidos de Transparência.

Inicialmente, o interesse é desenvolver e evoluir um projeto baseado em políticas de monitoração utilizando a arquitetura de sistemas multi-agentes (SMA). Para isso é feito um aprofundamento dos conceitos relacionados com SMA a fim de compreender como agentes de software atuando como monitores seriam capazes de efetuar coletas de dados confiáveis e em tempo real sobre o processo de software, para posteriormente divulgar os resultados acerca das ações do usuário e do software monitorado.

Serão criados modelos de avaliação da Transparência para estabelecer políticas de monitoração. Tais políticas serão sugeridas com base no *framework* i* [3]. Outra forma de representação de intencionalidades dos agentes será abordada a partir do modelo do Painel de Intencionalidades proposto por [4]. O objetivo do uso do Painel de Intencionalidade é motivado por seus benefícios de representação exclusiva de metas concretas e flexíveis, e suas inter-relações representadas através de elos de dependência, correlação, contribuição e equivalência no mesmo diagrama, considerando os níveis de relevância. A importância de criação de modelos é tornar

transparente também a arquitetura do SMA. Esse ponto toca diretamente no ponto da Transparência da Informação.

Os estudos tem nos revelado que outras características podem ser acrescentadas às políticas de monitoração para potencializar os atributos de Transparência. Confiança e Proveniência possuem características que parecem estar intimamente relacionadas nesse contexto. Em [5] são descritas relações de potencialização de influência bilateral entre Confiança e Transparência. Já em [9, 10] encontra-se registros da relação entre Confiança e Proveniência.

Confiança está relacionado na questão de acreditar ou se ter confiança em alguém ou alguma coisa, além de ser um fator psicológico que é reforçado a partir de questões de informação. O expectador necessita de informação ao longo de algum processo para que possa se sentir seguro de que o ocorrido é confiável. Daí nossa percepção de que a Transparência de software é capaz de influenciar em questões de Confiança e vice-versa.

Por outro lado, há características de Proveniência que estão intimamente ligadas às questões de Transparência e Confiança. A análise de taxonomias de Proveniência sugere que pontos como "Qualidade do Dado" (*data quality*), que permeia questões de "rastro ou linhagem do dado" que permitem estimar sua qualidade e "confiança" (*reliability*). Além de outras características de "trilhas de auditoria" (*audit trail*), atribuição ou "responsabilidade" pelo dado (*liability*) e informação que estão intimamente ligados aos conceitos de Transparência de Processo e Informação, presentes em [1].

Proveniência, nos casos citados, tem sido tratada na literatura científica como um processo para manter rastros de dados (origem, transição e destino) com o objetivo de permitir posteriormente a verificação de como o dado foi produzido, bem como a qualidade do processo que o produziu.

Portanto, umas das principais preocupações com a pesquisa é estabelecer políticas de avaliação da Transparência baseadas em critérios técnicos, a partir da evidência de sua relação com outras características capazes de potencializar o nível de confiança do usuário no processo de monitoração. Como a arquitetura do software também é um elemento passível de Transparência, tais políticas necessitam estabelecer ou sugerir formas de auditoria dos próprios agentes de monitoração, garantindo a transparência também para as ações dos mesmos.

Outro ponto importante na pesquisa é a condição de estabelecer os domínios do conceito de rastreabilidade, utilizado nesse artigo como Rastreabilidade, termo já classificado e definido na Engenharia de Software (ES) e relacionado à Engenharia de Requisitos (ER), e sua fronteira ou relação com as questões de Proveniência, termo amplamente usado na área de Banco de Dados.

Um ponto chave para o estudo destas abordagens está na construção de modelos baseados no *NFR framework,* proposto por [13]. Uma primeira impressão é que Rastreabilidade, tal como propõe a literatura da ES, trata-se da rastreabilidade de artefatos em um processo de software; do outro lado, a Proveniência trata de questões de persistência de rastros de execução e de sua recuperação.

No caso da Proveniência, há uma condição particular de rastros em diferentes camadas: na coleta de rastros na composição/concepção de determinado *workflow*, definida como prospectiva; e na coleta de rastros na execução de um *workflow*,

definida como retrospectiva. Apesar de algumas semelhanças há fortes indicações de que as duas abordagens trabalham em níveis de abstração diferentes.

A intenção é procurar entender como processos de Rastreabilidade podem dar suporte sistematizado na construção de processos de avaliação da Transparência e como que estruturas de Proveniência podem dar suporte às questões de estruturação, semântica, captura, persistência, e recuperação de rastros de um software avaliado.

3 Contribuições Científicas

Uma das principais expectativas com relação à contribuição da pesquisa é a definição de uma estratégia de monitoração com o objetivo de entender se atributos de Transparência estão sendo observados em um software em execução.

Com isso, pretende-se definir modelos explícitos de políticas de avaliação Transparência de software baseados em *framework* i*. Tal contribuição pretende satisfazer às necessidades de suporte metodológico, sistematizado e baseado em critérios técnicos para futuras implementações de sistemas que irão atuar sob aspectos dos requisitos não funcional de Transparência. Além disso, nesse primeiro momento tratar a construção e evolução de um SMA que automatize o processo de monitoração e validação.

Uma das primeiras contribuições elaboradas pela pesquisa foi detalhada em [2], que propôs a primeira versão de um SMA de coleta de dados e comparação com catálogo de Transparência [1]. Sua sistematização é realizada a partir da construção de 4 agentes: MONITOR, CANONIZADOR, CONSOLIDADOR e ANALISADOR. Em suma, o agente MONITOR trata de monitorar traços de execução de determinado software e gera um novo traço para dentro do SMA. A partir disso, o agente CANONIZADOR marca os traços de execução com delimitadores condizentes com os atributos do catálogo de Transparência [1]. O CONSOLIDADOR limpa os traços não marcados e gera um novo traço de execução para que o ANALISADOR possa comparar com o catálogo de Transparência. Ou seja, a partir dos traços de execução e padrões estabelecidos no catálogo é verificado se o software avaliado está em conformidade com atributos de Transparência.

Os modelos representados na ocasião foram baseados no *framework* i* com o objetivo de representar e facilitar a análise de execução dos agentes com o uso da representação da intencionalidade distribuída [2], comum aos SMA. Feitas as correções, evolução do SMA e melhorias no modelo intencional, o trabalho foi submetido e aceito [14] pela Revista de Informática Teórica e Aplicada.

4 Conclusões

Pelo andamento das pesquisas bibliográficas, escrita de artigos preliminares, de uma primeira versão do SMA e das discussões em grupo, entendemos que o caminho que menos onera a compreensão de propostas de monitoração de requisitos não funcionais perpassa por criação de modelos. O uso de modelos baseados no *framework* i*

permite-nos estabelecer tanto objetivos, ou metas flexíveis para a estratégia de monitoração, como também permite com que *rationales*, ou operacionalizações, da monitoração sejam evidenciadas e melhor compreendidas.

Por outro lado, envolver na pesquisa conceitos de Transparência, Confiança, Rastreabilidade e Proveniência requer uma representação de suas características e também de suas relações. Entendemos que o uso do NFR *framework* [13] possa ser uma ferramenta adequada para tais representações e os catálogos de cada uma dessas abordagens serão construídos para que fiquem evidenciadas características individuais, comuns e suas relações.

5 Trabalhos Futuros

Os integrantes da pesquisa, juntamente com o Grupo ER - PUC-Rio discutem atualmente a respeito de conceitos e inter-relacionamentos entre Rastreabilidade e Proveniência para que modelos descritos em *NFR framework* sejam estabelecidos e as fronteiras e interseções entre as duas abordagens sejam delineadas.

Alguns trabalhos futuros começam a surgir em um *brainstorm* a partir da análise das possibilidades com a evolução da pesquisa. Uma primeira visão de trabalho que pode ser promissor é o desenvolvimento de estratégias para sistemas de recomendação. A partir do momento em que a execução da monitoração é realizada há o registro da monitoração, como também, o registro de conformidades e não conformidades do *software* monitorado em relação aos padrões de qualidade de Transparência. Modelos intencionais para sistemas de recomendação poderão ser criados e ser úteis para definir sistemas que possam, a partir da base de conhecimento criada e das não conformidades, sugerir alternativas para softwares monitorados e que não estejam de acordo com os padrões pré-estabelecidos de Transparência.

Outra possibilidade de trabalho futuro é avaliar se a proposta da pesquisa é aderente a outros requisitos não funcionais. Entendemos que a pesquisa indica que outros requisitos não funcionais poderão ser operacionalizados e validados a partir da proposta da política que está sendo criada. Apesar disso, uma questão que já nos inquieta é se uma estratégia de monitoração tem que ser obrigatoriamente diferente para requisitos não funcionais distintos.

Como também tratamos na pesquisa sobre questões de persistência de traços de execução e de monitoração seria importante que meta-modelos do registro das informações fossem sugeridos de forma padronizada com o objetivo de reuso.

Referências bibliográficas

1. Grupo Transparência de Software, PUC-Rio. http://www.er.les.inf.puc-rio.br/~wiki/index.php/Transpar%C3%AAncia_de_Software.
2. Leal, A.L. de C., Sousa, H.P., Leite, J.C.S.P,: Lucena, C.J.P. de.: Modelagem intencional de políticas e implementação de agentes de monitoração de transparência em sistemas de software. ftp://ftp.inf.puc-rio.br/pub/docs/.../11_20_castro.pdf (2011).
3. Leite, J.C.S.P, Capelli, C.: Exploring i* Characteristics that Support Software Transparency. In.: Proc. of the 3rd International i* Workshop pp. 51-54. Recife-PE (2008)

4. Oliveira, A.P.A., Leite, J.C.S.P.. Cysneiros, L.M., Cappelli, C.: Eliciting Multi-Agent Systems Intentionality: from Language Extended Lexicon to i* Models. In: Jornadas Chilenas de Computação 2007 - UNAP, 2007, Iquique.: Proceedings of the XXVI International Conference of the Chilean Computer Science Society. Iquique: Los Alamitos: IEEE Computer Society Press, v. 16. p. 40-49 (2007)
5. Cysneiros, L.M., Werneck, V.M.B.: An Initial Analysis on How Software Transparency and Trust Influence each other. In: 12th Workshop on Requirements Engineering, Proceedings of, v. 1. pp. 27-32. Valparaiso. Proceedings of 12th Workshop on Requirements Engineering, Universidad Tecnica Federico Santa Maria (2009)
6. Leite, J. C. S. P.. Sistemas de Software Transparentes. Palestra Convidada. In.: Simpósio Brasileiro de Engenharia de Software. http://www-di.inf.puc-rio.br/~julio/Slct-pub/transp-sbes.pdf. Florianópolis, SC (2006)
7. Leal, A.L.C., Sousa, H.P., Leite, J.C.S.P., Braga, J.L.: Transparência aplicada a modelo de negócios. In: XIV WER - Workshop em Engenharia de Requisitos, v. XIV. pp. 321-332. Proceedings XIV CibSE, Rio de Janeiro (2011)
8. Lei de Acesso. Lei nº 12.527, de 18 de novembro de 2011, http://www.acessoainformacao.gov.br/acessoainformacaogov/acesso-informacao-brasil/legislacao-integra-completa.asp
9. Clifford L. A.: When documents deceive, Trust and provenance as new factors for information retrieval in a tangled web. In.: Journal of the American Society for Information Science and Technology, pp. 12-17 (2001)
10. Dai, C., Lin, D., Bertino, E., Kantarcioglu, M.: Trust evaluation of data provenance. Technical Report CERIAS TR, Purdue University (2008)
11. Dividino, R., Schenk, S., Sizov, S., Staab, S.: Provenance, trust, explanations–and all that other meta knowledge. Künstliche Intelligenz (2009)
13. Chung, L., Nixon, B., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. Kluwer Academic Publishers, Massachusetts (2000)
14. Leal, A. L. C., Sousa, H. P., Leite, J. C. S. P., Lucena, C. J. P.. Aplicação de Modelos Intencionais em Sistemas Multi-agentes para Estabelecer Políticas de Monitoração de Transparência de Software. Revista de Informática Teórica e Aplicada: RITA, v. 20, p. 111-138, 2013.

Ubiquitous, Pervasive and Mobile Computing: A Reusable-Models-based Non-Functional Catalogue

Milene Serrano[1] and Maurício Serrano[1]

[1] Universidade de Brasília (UnB/FGA), Curso de Engenharia de Software, Brasil.

{mileneserrano, serrano}@unb.br

Abstract. This paper reports on experiences in the development and use of a non-functional catalogue based on Goal-Oriented Requirements Engineering. This catalogue is specific for ubiquitous, pervasive and mobile computing domain, and it represents a common baseline focused on the concerns of this domain, their interdependencies and operationalizations. Moreover, our reusable models-based catalogue has been thoroughly evaluated by using different ubiquitous contexts. Therefore, it is in constant evolution. According to our evaluation process, the reuse of existing and shared models demonstrates, among others benefits, that: (i) it offers a suitable and reusable body of knowledge to deal with ubiquitous, pervasive and mobile quality criteria; and (ii) it reduces elicitation time and team efforts while working with non-functional issues. We also present an overview about the catalogue's development process, emphasizing the last obtained refinements.

Keywords: Non-Functional Requirements; Goal-Oriented Requirements Engineering; Reusable-Models-based Catalogue; Ubiquitous, Pervasive and Mobile Computing; Systematic Development.

1 Introduction

In the Ubiquitous, Pervasive and Mobile Computing domain, we must consider different issues, such as: devices heterogeneity, content adaptation, context awareness, mobility, ever-changing environments, and content server distribution. Various authors (e.g. [1][2]) agree with the fact that these issues are real concerns in that domain. Therefore, these concerns are intrinsic and commonly found when we are, for example, developing ubiquitous, pervasive and mobile systems and applications.

Although there is a consensus about the importance of those issues, unfortunately, there is a lack of support - for the mentioned domain - in order to guide developers from the requirements elicitation to implementation, specially while respecting non-functional requirements. Moreover, we know that the requirements engineering activities demand an adequate conceptual modeling support to be appropriately performed.

In order to contribute to this field, we have dedicated part of our work to develop a non-functional requirements catalogue, specifically for Ubiquitous, Pervasive and Mobile Computing domain. We also provide a method for using this catalogue to guide systematic developments by considering applications in this domain. Both, the catalogue as well as the method, are in constant refinement to attend new demands in this domain, in which the evolution is inherent to deal with emergent technologies.

Additionally, it is important to emphasize that the catalogue is constructed by using the NFR Framework. This framework was first proposed by [3], and it constitutes a Goal-Oriented Requirements Engineering (GORE) [4] approach for capturing non-functional requirements, by also defining their interdependencies and operationalizations. This kind of approach is interesting as it ends where the traditional approaches, i.e. Object-Oriented approaches (e.g. Rational Unified Process (RUP)), start. In other words, GORE approaches focus their contributions on activities that precede the requirements specification to facilitate the validation of architectural decisions. Therefore, the NFR Framework offers support to: (i) design alternatives for dealing with different non-functional issues; (ii) consider conflicts, tradeoffs and priorities, and (iii) evaluate the decisions impact based on non-functional isues that commonly influence the success of ubiquitous, pervasive and mobile applications.

In this paper, we briefly present our catalogue and its contributions. The paper is organized in sections. In Section 2, we mention the goals of our work. In Section 3, we describe the scientific contributions of the proposed non-functional catalogue for ubiquitous, pervasive and mobile applications development by offering an overview of the catalogue's development process and contents. In Section 4, we have the final considerations and, finally, in Section 5, we suggest directions for further work.

2 Objectives of Research

We have been investigated interesting non-functional-requirements-driven approaches, such as [5][6]. However, most of them are issue-specific-oriented and they support specific applications. Therefore, they do not provide a reusable models baseline for applications that do not match with these specific issues and application profiles.

Additionally, there is a lack of approaches that guide the development of ubiquitous, pervasive and mobile applications from the requirements to code, focused on different quality criteria, such as: user satisfaction, device heterogeneity, mobility, content adaptation, context awareness and ever-changing environments.

Considering these timely observations, our research is mainly motivated by the goals: (i) provide a guideline, based on reusable models, for ubiquitous, pervasive and mobile applications in order to orient software engineers from the requirements to code in the development of these applications, and (ii) as our non-functional catalogue is constructed by considering non-functional requirements that are commonly found in ubiquitous, pervasive and mobile applications, it provides generic support, and it also contributes to the reuse of these generic solutions in different applications, whose developers can concentrate their efforts on the specialization of these solutions to better attend specific concerns.

3 Scientific Contributions

Our non-functional requirements catalogue was developed by composing different conceptual non-functional requirements models in order to obtain a common reusable models baseline for ubiquitous, pervasive and mobile applications. In the catalogue's development process, we focused our attention on three main activities, non-functional requirements (1) elicitation; (2) decomposition, and (3) interdependencies identification. In order to perform these activities, we used ubiquitous, pervasive and mobile scenarios in different cognitive domains (e.g. e-commerce, educational and dental), consultation with experts, and our experimental research. Our objective was the elicitation of quality criteria commonly found while developing ubiquitous, pervasive and mobile applications. Moreover, we evaluated the elicited non-functional requirements with the users participation. This process also contributed to the catalogue's evolution and adequacy. Therefore, in practice, the activities described before as well as the evaluation were iteratively performed, by allowing to incrementally develop the catalogue as a knowledge baseline, specific for the domain of interest.

A briefly overview about the catalogue's development are presented through the following phases: (i) **Investigation**, in which we investigated the State-Of-The-Art to compile an adequate initial understanding of ubiquitous, pervasive and mobile concerns. At the end of this phase, we obtained a first version of the catalogue, which consisted of a top-level ubiquitous, pervasive and mobile requirements. Basically, at this stage, the catalogue included three top-level non-functional requirements (i.e. Ubiquity, Pervasiveness, and Mobility), and four sub-non-functional requirements (i.e. Content Adaptability, Context Awareness, Device Heterogeneity and Software Processes Complexity Invisibility); (ii) **Experimental Research**, in which we experimented the first version of our catalogue by using it to systematic develop ubiquitous, pervasive and mobile applications. As the result of our first experimental research, we obtained some interdependencies between the non-functional requirements as well as some operationalizations for them. We also identified the necessity of some improvements in our catalogue first version. Therefore, we evolved it by incorporating the non-functional requirement User Satisfaction as a seminal ubiquitous, pervasive and mobile issue. Moreover, we refined the User Satisfaction, decomposing it on Usability, Content/Service Accessibility, Ubiquitous, Pervasive and Mobile Profile Awareness, and others. This new catalogue version was constituted of 21 non-functional requirements; (iii) **Iterative Evolution**, in which we have performed, since 2007, several iterations in order to evolve the catalogue by following emergent technologies and others ever-changing elements in ubiquitous, pervasive and mobile contexts. Basically, during this phase, we iteratively (a) created new catalogue contents; (b) eliminated replications, redundancies and ambiguous specifications; and (c) improved the reusability condition of the catalogue. Some case studies that are used in this iterative evolution are detailed presented at [7][8][9][10][11][12]; and (iv) **Collaborative Evolution**, in which, based on the fact that we opened our catalogue to scientific community, we have incrementally refined the catalogue according to the feedback we obtained from others ubiquitous, pervasive and mobile groups' projects.

Based on this process, our catalogue's last version is composed of almost 700 interdependent non-functional requirements. They are organized according to their prioritizations in ubiquitous, pervasive and mobile applications. The prioritizations were obtained throughout the last five years. Actually, the non-functional requirements that are most commonly found in the ubiquitous, pervasive and mobile applications development received highest priority. Some of these non-functional requirements are Ubiquity, Pervasiveness, Mobility and User Satisfaction. Therefore, they are at the top-level of the catalogue's non-functional requirements hierarchy. Then, we have 17 non-functional requirements at the second level, which includes: Content Adaptability, Context Awareness, Device Heterogeneity and Transparency. Furthermore, there are almost 200 non-functional requirements at the third level, such as: Self-Regulation, Autonomy, Reactivity, and Controllability. According to our research, these non-functional requirements are applicable to a broad class of ubiquitous, pervasive and mobile applications. As the catalogue is in constant evolution, the refinements also involve refactoring in the non-functional requirements' prioritizations.

In order to improve the use of our catalogue, we developed a Web-application to deal with the huge number of non-functional requirements reusable models shared in our baseline. It facilitates the access of these reusable models and it helps in the presentation and browsing of the catalogue's contents based on an exploration tree to navigate and select a desired non-functional requirements, their meanings, and links to the models of their interdependencies. Moreover, we also developed a method to allow the use of our catalogue as a guideline to orient the systematic development of ubiquitous, pervasive and mobile applications from the requirements to code by focusing on non-functional requirements commonly found in these applications. The catalogue's use method is based on five activities, some of them decomposed on others sub-activities. They are: Explore, Collect, Model, Operationalize and Validate.

The Explore activity is divided into Consult and Extract sub-activities. It assists the developers in catalogue's exploration by improving the investigation of different ubiquitous, pervasive and mobile concerns, and in the knowledge extraction from the catalogue by improving the deduction of what knowledge - i.e. catalogue's predefined non-functional requirements - is pertinent for the application under development.

The Collect activity is composed of Pick-up and Instantiate/Evolve sub-activities. It helps in the requirements elicitation by picking them up from our baseline. The developers can decide to use the requirements as they are specified in the catalogue, or they can instantiate these requirements and evolve them to better attend the specific concerns of the application under development. When the non-functional requirements match with the ubiquitous, pervasive and mobile application's needs, the developers can basically reuse them as they are defined. Although, some adjustments can be necessary. Thus, the non-functional requirements must be instantiated and evolved.

The Model activity is based on Decompose or Determine Interdependencies. It assists in the design by guiding the non-functional requirements modeling. If the developers only picked-up the non-functional requirements from the baseline without the necessity of adjustments, the Model activity is not necessary in this case. The developers can obtain the model from the catalogue and use it as it is specified. However, if the developers modified and adapted the non-functional requirements by instantiating

and evolving them, it is necessary to decompose and/or determine their new interdependencies. We suggest a modeling based on the NFR Framework's notation, which is centered on a graph called Softgoal Interdependencies Graph (SIG). In a SIG, the non-functional requirement is a softgoal that has interdependencies with others non-functional requirements. More about the NFR Framework can be obtained in [3].

The Operationalization activity assists the developers in low abstraction level by offering predefined operationalizations' set, which can be viewed as a set of possibilities and strategies in order to facilitate and guide the implementation of different issues. However, when the catalogue's offered support does not address the developers' needs, it is also possible to establish new support using the developers' expertise.

The Validate activity is divided into Evaluate and Solve Conflicts sub-activities. It helps in the non-functional requirements evaluation using the notion of propagation rules and offspring labels - *denied*, *weakly denied*, *undecided*, *weakly satisfied*, *satisfied* and *conflict* - provided by the NFR Framework [3]. Here, it is possible to check interdependencies using specific correlation rules and also considering the stakeholders' participation. Moreover, it is also possible to identify conflicts in both, non-functional requirements interdependencies and operationalizations. In our case studies, for each application under development, we scheduled meetings with the stakeholders to evaluate the non-functional requirements, their interdependencies and operationalizations by considering specific scenarios to test/investigate the applicability, pertinence and adequacy of the requirements obtained by exploring, collecting, modeling, and operationalizing centered on the catalogue. The method also contemplates the feedback notion, allowing refinements when misconception or misunderstanding occurs from faulty judgment, deficient knowledge or lack of forethought.

4 Conclusions

In this paper, we report on our experiences with our reusable models-based catalogue centered on GORE. The main purpose of our initiative on constructing this catalogue is to provide a common baseline of ubiquitous, pervasive and mobile concerns - i.e. a framework for this domain - in order to guide the development of applications in this domain from the requirements to code. It is possible as the catalogue is centered on different non-functional requirements as well as it contains the interdependencies between these requirements and also possible ways to operationalize them.

The results obtained by evaluating our framework at both, Software Engineering Laboratories at PUC-Rio and at UnB/FGA, indicate that we are in the right direction, mainly if we focus our attention on: (i) improving the elicitation process centered on non-functional requirements of ubiquitous, pervasive and mobile applications; (ii) reducing the spent elicitation time; (iii) reducing the team efforts in dealing with common found ubiquitous, pervasive and mobile non-functional requirements, by allowing that the team concentrate the efforts on specific issues, and (iv) providing a guideline for the developers of ubiquitous, pervasive and mobile applications that orients them from the requirements to code. Thanks to the GORE nature, the proposed framework also maintains the traceability for Requirements Engineering activities.

It is relevant to consider that we are also conscious about our unsolved challenges, such as: (i) the demand of time to maintain and evolve our catalogue; (ii) the necessity of advanced mechanisms for knowledge management and version controller; and (iii) the necessity of a tool support to improve the catalogue's reuse. As presented in Section 5, we have experimented with possible strategies to deal with these challenges.

5 Ongoing and Future Work

In order to maintain our catalogue up to date, we have opened access to our baseline. Our intention is the constant refinement of it by (i) providing more precise methods; (ii) changing the priorities of the non-functional requirements specified into the catalogue, and (iii) incorporating new quality criteria as well as their interdependencies with others criteria, and operationalizations based on emergent technologies. We also envision further work on offering a tool support to facilitate the catalogue's use.

References

1. Weiser, M.: Some Computer Science Issues in Ubiquitous Computing. CACM, vol. 36(7), pp. 75-84 (1993)
2. Landay, J.A., Borriello, G.: Design Patterns for Ubiquitous Computing. IEEE, vol. 36, no. 8, pp. 93-95 (2003)
3. Chung, L., Nixon, B., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. Int. Series in Soft. Engineering, vol. 5, 476 pages (2000)
4. Mylopoulos, J.: Goal-Oriented Requirements Engineering. In: XI Conf. Iberoamericana de Soft. Eng., pp. 13-17 (2008)
5. Pashazadeh, S.: Modeling Non-Functional Requirements in Designing Middleware for Pervasive Healthcare System. In: 5th Int. Conference on Application of Information and Communication Technologies, pp. 1-5 (2011)
6. Kavakli, E., Kalloniatis, C., Loucopoulos, P., Gritzalis, S.: Incorporating Privacy Requirements into the System Design Process. In: vol. 16(2), pp. 140-158 (2006)
7. Serrano, Milene: Reuse-oriented Approach for Incremental and Systematic Development of Intentional Ubiquitous Applications. Doctoral Thesis, 228 p. (2011)
8. Serrano, Milene, Lucena, C.J.P. de.: Dynamic Content Adaptation in Mobile Applications driven by Intentional Multi-Agent Systems. Chapter in the Handbook of Research on Mobile Software Engineering, 18 p. (2011)
9. Serrano, Milene, Lucena, C.J.P. de.: Dynamic Interface Adaptation for Ubiquitous Devices driven by Agents and Ontologies. In: 1st Int. Conf. on Pervasive and Embedded Comp. and Communication Systems, 10 p. (2011)
10. Serrano, Milene; Lucena, C.J.P. de.: Intentional Mobile Agents in Ubiquitous Systems. In: 3rd Int. Conf. on Agents and Artificial Intelligence, 10 p. (2011)
11. Serrano, Milene, Lucena, C.J.P. de.: Applying FIPA Standards Ontological Support to Intentional-MAS-Oriented Ubiquitous System. In: 12th Int. Conf. on Enterprise Information Systems, pp. 114-121 (2010)
12. Serrano, Milene, Serrano, Maurício, Lucena, C.J.P. de.: Ubiquitous Software Development Driven by Agents' Intentionality. In: 11th Int. Conf. on Enterprise Information Systems, pp. 25-34 (2009)

Requisitos ao Código: Uma Proposta para o Ensino da Engenharia de Software no Ensino Médio

Maurício Serrano[1] e Milene Serrano[1]

[1] Universidade de Brasília (UnB/FGA), Curso de Engenharia de Software, Brasil.

{serrano, mileneserrano}@unb.br

Resumo. Esse artigo apresenta uma proposta educacional visando o ensino da Engenharia de Software para alunos do ensino médio, mais precisamente, ensinando como ocorre o processo de desenvolvimento de software dos requisitos ao código. Para tanto, a metodologia fundamenta-se no uso de plataformas de ensino inovadoras e robótica educacional, com as quais são explorados recursos lúdicos, temas multidisciplinares, animações 3D, robôs e sensores para tornar o aprendizado mais divertido, realista, e criativo. Os autores da proposta apoiam-se nas teorias do Construtivismo, Construcionismo e outras. Aplicada a um pequeno grupo de alunos do ensino médio, a proposta já demonstrou-se válida, dentre outros benefícios: (i) à elicitação de requisitos; (ii) à modelagem desses; (iii) ao projeto de interfaces, e (iv) à implementação.

Palavras-chave: Requisitos, Engenharia de Software, Construtivismo, Construcionismo, Robótica Educacional, Orientação à Meta.

1 Introdução

A Universidade de Brasília - Faculdade do Gama (UnB/FGA) é fruto do Plano de Desenvolvimento Institucional (PDI) da UnB. Esse plano de desenvolvimento determinou a instalação de novos campi de ensino superior, vinculados ao Programa de Apoio a Planos de Reestruturação e Expansão das Universidades Federais (REUNI). O intuito era aproximar os núcleos de ensino às comunidades mais afastadas. Assim, a UnB/FGA originou-se como um polo de tecnologia para as cidades no entorno do Gama/DF, oferecendo cinco cursos: Engenharia de Software, Engenharia Aeroespacial, Engenharia Automotiva, Engenharia Eletrônica e Engenharia de Energia.

Apesar da proposta inicial do campus da UnB/FGA, percebe-se, na prática, uma participação ainda muito pequena dos alunos das escolas, concentradas na área de abrangência do campus, nos processos de seleção da universidade. Atribui-se essa pequena participação, principalmente: (i) à noção entre os alunos do ensino público de que a UnB, bem como a UnB/FGA, são inacessíveis para os mesmos; (ii) à qualidade inferior do ensino nas escolas públicas, e (iii) à necessidade de muitos desses alunos trabalharem antes mesmo de obterem um diploma de ensino superior. Diante desse

cenário, os autores desse artigo desenvolveram uma proposta de ensino diferenciada visando alcançar algumas metas, tais como: (i) promover uma oportunidade de integração e socialização dos alunos de ensino médio junto à UnB/FGA, e (ii) introduzir a Engenharia de Software, mais especificamente o desenvolvimento de software dos requisitos ao código, usando recursos lúdicos, animações 3D, Plataformas de Ensino (e.g. Plataforma Alice [1] e Ambiente Scratch [2]), e Robótica Educacional [3].

A Plataforma Alice, por exemplo, é um software gratuito, proposto pela Carnegie Mellon University (CMU) em Pittsburgh, que corrobora com as nossas metas, dando um primeiro passo, uma vez que essa plataforma explora o uso da tecnologia 3D com animações gráficas, jogos interativos e vídeos compartilhados na Web no ensino da programação orientada a objetos. Adicionalmente, o ambiente de ensino propiciado pela plataforma possibilita aos alunos visualizarem imediatamente seus pequenos e iniciais programas em execução, o que pode melhorar a capacidade de aprendizado desses alunos em tópicos introdutórios da área de Engenharia de Software e afins.

Já a Robótica Educacional, ou Robótica Pedagógica, pressupõe a introdução de recursos tecnológico (ex. computadores e software) no intuito de auxiliar na educação. Em âmbito internacional, entidades e governos de países desenvolvidos encontram-se avançados no uso de tecnologias para aprimorar o processo de ensino e aprendizado. Entretanto, no Brasil, a Robótica Educacional caminha devagar, com ainda pequenos esforços para tornar o cidadão brasileiro competitivo, qualificado em um mundo globalizado e exigente. Um dos princípios fundamentais da Robótica Educacional é integrar vários atores sociais que possuem relações de ensino e aprendizado, visando promover a investigação de conceitos multi e interdisciplinares (ex. matemática, português, física, geometria, biologia e outros), e o desenvolvimento de projetos - do planejamento (i.e. análise e projeto) à construção (i.e. implementação) - de forma criativa, cooperativa, exploratória e experimental.

Em ambas as iniciativas de ensino, o aluno é instigado a pensar, questionar, argumentar, pesquisar e propor soluções a um dado problema, reagindo com base nos seus próprios valores, na sua própria vivência, o que possibilitará ao mesmo participar ativamente da construção das suas próprias estruturas de conhecimento.

Nesse artigo, demos um passo adiante, e apresentamos uma proposta de ensino que faz uso dessas iniciativas, mas como um alcance um pouco mais ousado, uma vez que o nosso objetivo é ensinar Engenharia de Software, mais precisamente, desenvolvimento de software dos requisitos ao código para alunos de Ensino Médio. Além disso, fundamentamos nossa proposta na Teoria do Construtivismo de Jean Piaget [4]; na Teoria do Construcionismo de Seymour Papert [5]; nas Invariantes Pedagógicas de Célestin Freinet [6]; e nos Paradigmas de Aprendizado de Leclercq & Denis [7] [8].

A submissão desse artigo para uma conferência de Engenharia de Requisitos tem como objetivo divulgar a proposta educacional, orientada por tópicos de interesse da área, junto à comunidade científica para que novas iniciativas ocorram, despertando talentos (i.e. novos Engenheiros de Software e novos Engenheiros de Requisitos).

O artigo está organizado em seções. Na Seção 2 são abordados os objetivos da pesquisa. Na Seção 3, são apresentadas as principais contribuições científicas. A Seção 4 concentra-se nas considerações finais, e finalmente, a Seção 5 descreve os trabalhos em andamento bem como as intenções dos autores para trabalhos futuros.

2 Objetivos da Pesquisa

Considerando o Estado-da-Arte nos tópicos de interesse dessa pesquisa, ressalta-se a existência de algumas propostas de ensino, mais precisamente no âmbito internacional, preocupadas em estimular os alunos de ensino médio quanto ao raciocínio lógico, visando melhorar o rendimento dos mesmos nas disciplinas exatas (principalmente, matemática e física). No âmbito nacional, poucas iniciativas nesse sentido são encontradas, tais como a do Projeto Destacom [9].

Apesar de válidas, essas iniciativas mostram-se limitadas diante dos anseios dos docentes do campus da UnB/FGA, sendo necessário não apenas estimular o raciocínio lógico entre os alunos de ensino médio, mas também ensinar noções intrínsecas às áreas de graduação oferecidas no campus. Como uma primeira iniciativa nesse sentido, pensou-se no ensino de como elicitar os requisitos de um software, modelar esses requisitos e codificar esse software com base na modelagem estabelecida. No intuito de contribuir nesse sentido, propõe-se uma metodologia de ensino diferenciada, centrada em Plataformas de Ensino Inovadoras e na Robótica Educacional, cujas metas de ensino principais são: (i) melhorar o raciocínio lógico; (ii) apresentar como um software é desenvolvido dos seus requisitos ao código, e (iii) estimular o senso crítico dos alunos de ensino médio, motivando-os a questionar, argumentar, e participar, futuramente, de forma colaborativa ao longo da sua vida acadêmica e profissional.

3 Contribuições Científicas

A proposta de ensino apresentada nesse artigo baseia-se no uso de atividades que compreendem desde a sugestão de um desafio, que no caso da Plataforma Alice e similares será o desenvolvimento de um minigame, e no caso da Robótica será o desenvolvimento de um robô, até a concretização desse desafio, ou seja, a implementação do minigame ou do robô. Para que o processo de desenvolvimento seja conduzido adequadamente, são introduzidos aos alunos de ensino médio conceitos e outras particularidades da Engenharia de Software, incluindo; (i) a necessidade de se elicitar adequadamente os requisitos para ambos, minigame e robô; (ii) a necessidade de se modelar esses requisitos usando Orientação à Meta [10], mais precisamente a ferramenta de modelagem i* [11]; (iii) a necessidade de se validar - junto aos interessados - o protótipo das interfaces, intrínseco ao desenvolvimento de minigames, usando uma visão simplificada centrada em cenários [12], e (iv) a necessidade de se implementar ambos, minigame e robô, com base, respectivamente, na validação das interfaces junto aos interessados e na modelagem orientada à meta.

Consideremos, primeiramente, as justificativas quanto às escolhas: (1) Orientação à Meta, o uso desse paradigma baseia-se no fato de um robô ser mais facilmente especificado quanto aos objetivos a serem atingidos pelo mesmo. Além disso, para atingir esses objetos, uma forma apropriada e intuitiva é através da realização de tarefas, que podem ser decompostas em subtarefas. (2) nesse contexto, a Ferramenta de Modelagem i*, orientada à meta, oferece uma notação centrada, não exclusivamente, mas principalmente, nos conceitos de meta, tarefa e dependência entre tarefas; e (3) Cená-

rios, o uso dessa notação deu-se pelo fato de permitir - usando uma descrição semi-estruturada e em linguagem natural - o detalhamento quanto aos propósitos do minigame, a especificação dos atores (personagens) do mesmo, a determinação dos recursos envolvidos no mesmo, e a apresentação dos episódios do mesmo.

Aproveitando o fato dos minigames, desenvolvidos usando a Plataforma Alice ou ambientes de ensino inovadores similares, serem baseados em animações gráficas 3D, propomos o uso desse recurso lúdico para ensinar a noção de interfaces gráficas junto aos alunos de ensino médio. Na Plataforma Alice, um minigame, simplificadamente, é composto por: (i) um ambiente (e.g. deserto, neve, gramado, dentre outros), dado por um *template* predefinido (ou refinado a partir de um *template* predefinido); (ii) personagens e objetos, ambos definidos como Objetos, usando como base a programação orientada a objetos; (iii) propriedades para cada conjunto de Objetos, ou sejam, atributos, considerando a programação orientada a objetos; e (iv) comportamentos para cada conjunto de Objetos, ou sejam, métodos, considerando a programação orientada a objetos. A proposta de ensino focada na Plataforma Alice e em ambientes similares consiste em trabalhar a noção de projeto de interfaces, validando-as com os interessados usando "protótipos". O desafio (i.e. o minigame) é lançado pelo professor, que atua como um orientador nesse processo. A ideia é dividir a turma em grupos pares, os quais representam ora os interessados (ou sejam, os clientes interessados no desenvolvimento do minigame), ora os desenvolvedores (ou sejam, os engenheiros de software que desenvolverão o minigame). No papel de interessados, os alunos procuram avaliar o projeto das interfaces, manifestando, cada qual, sua satisfação quanto as mesmas, e criticando-as, caso as mesmas não correspondam às expectativas, já estabelecidas na divulgação do desafio por parte do professor. Já no papel de desenvolvedores, os alunos projetam as interfaces, provendo aos interessados os protótipos das mesmas. Como são minigames, entende-se pelo projeto de interfaces, o uso de imagens estáticas, que juntas proveem aos interessados uma noção dinâmica de como futuramente se comportará o minigame. Em outras palavras, o projeto de interfaces confere uma visão quadro a quadro da animação 3D que representa o minigame. Finalmente, os alunos, no papel de engenheiros de software, implementam o minigame usando recursos gráficos 3D. Percebe-se, com essa breve descrição, que os alunos têm a oportunidade de: (i) entender os requisitos necessários para o desenvolvimento do minigame; (ii) modelar esses requisitos usando como base ambos, projeto de interfaces e cenários; (iii) validar o projeto de interfaces junto aos usuários antes mesmo da implementação, e (iv) implementar o projeto, obtendo uma animação 3D/minigame.

Usando Robótica Educacional, a proposta de ensino diferencia-se um pouco, visando explorar outros conceitos e fundamentos da Engenharia de Software. Quanto ao lançamento do desafio, o mesmo continua sendo realizado pelo professor. A turma pode ser dividida de diferentes maneiras, como por exemplo: meninos e meninas; duplas, grupos pares, dentre outras abordagens. Lembrando que os focos são o robô, suas metas e tarefas. Portanto, o desafio consiste em desenvolver um robô que tem objetivos bem definidos. Os alunos de ensino médio são orientados, então, a definir tarefas e subtarefas capazes de atingir esses objetivos. Dessa forma, a proposta de ensino centrada em robótica prevê o uso da Orientação à Meta na elicitação e modelagem dos requisitos. Faz-se necessária a introdução dos alunos de ensino médio

quanto ao uso da Ferramenta de Modelagem i*. Nos primeiros experimentos, propõem-se que essa introdução deve apresentar uma visão simplificada da Ferramenta de Modelagem i*, ou seja, focada, principalmente, nos elementos: meta, tarefa e subtarefas. Posteriormente, em experimentos mais avançados, que exploram a colaboração entre robôs ou mesmo a competição entre robôs, pode ser introduzida a noção de dependência, por exemplo. Uma vez modelados os requisitos, os alunos podem implementar o robô com base nessa modelagem. Percebe-se, com essa breve descrição, que os alunos têm a oportunidade de: (i) entender os requisitos necessários para o desenvolvimento do robô; (ii) modelar esses requisitos usando como base uma modelagem orientada à meta centrada, principalmente, nas noções de meta, tarefa, subtarefa e dependência, e (iv) implementar o robô com base na modelagem.

Vale ressaltar que já aplicamos, junto a um pequeno grupo de alunos do ensino médio, uma primeira versão da nossa metodologia de ensino. Essa primeira interação permitiu, dentre outras contribuições, refinar a proposta, incorporando, por exemplo, a especificação das interfaces em cenários. Os resultados obtidos demonstram que os autores encontram-se na direção correta, agregando valores em vários contextos, desde a socialização dos alunos de ensino médio junto a UnB/FGA até mesmo o aprendizado relativamente rápido de vários aspectos do desenvolvimento de software (e.g. elicitação de requisitos, modelagem de requisitos, projeto de interfaces e implementação). Além de obterem um primeiro contato com o paradigma da Orientação à Meta, a linguagem de programação orientada a objetos e o uso de cenários.

4 Conclusões

Esse artigo reportou nossas iniciativas no intuito de propor uma metodologia de ensino diferenciada, capaz de conduzir o aprendizado de alunos do ensino médio em tópicos das áreas de Engenharia de Software e Engenharia de Requisitos, tal como o desenvolvimento de "software" orientado à meta. Desenvolvimento esse ensinado aos alunos considerando desde a elicitação de requisitos, passando pela modelagem dos mesmos e, finalizando, com a implementação do software desejado.

Os alunos de ensino médio, dentre outros benefícios, têm a oportunidade de aprender, através de recursos lúdicos, temas multidisciplinares e robótica: (i) como elicitar os requisitos de um software usando Orientação à Meta; (ii) como validar uma interface junto aos interessados; (iii) como modelar os requisitos elicitados usando Orientação à Meta; (iv) como implementar o software, e (v) como implementar a interface.

De acordo com os resultados obtidos com a aplicação da primeira versão dessa metodologia de ensino junto a um grupo limitado de alunos do ensino médio, podem ser destacadas algumas contribuições: (i) alto interesse entre os alunos de ensino médio ao longo de todas as aulas ministradas (até o momento, um total de 5 aulas, com duração de 2 horas cada); (ii) satisfação dos alunos de ensino médio quanto ao fato de estarem desenvolvendo um "produto", usando conceitos e suportes que, atualmente, são apresentados apenas para graduandos; (iii) socialização entre os alunos de ensino médio bem como entre esses alunos e os professores e monitores da UnB/FGA; (iv) facilidade no aprendizado de conceitos e fundamentos inerentes à Engenharia de

Software; (v) possibilidade de construção das próprias estruturas de conhecimento; (vi) desenvolvimento do senso crítico (os alunos evoluíram ao longo das aulas, em termos, por exemplo, de participação em sala bem como no número de questionamentos entre alunos e entre alunos e professores/monitores), e (vii) outras contribuições.

5 Trabalhos Em Andamento e Futuros

Como comentado anteriormente, uma primeira versão dessa metodologia já foi aplicada a um grupo limitado de alunos do ensino médio, o que permitiu refiná-la quanto ao uso de novas técnicas, várias delas baseadas nas invariantes pedagógicas propostas em [7] [8], tais como: o uso de auto-avaliação entre os alunos de ensino médio bem como o uso de um diário de trabalho para registro das experiências adquiridas.

Atualmente, a proposta de ensino apresentada vem sendo aplicada a um grupo de 13 alunos do ensino médio, expandindo-se, em breve, para atender um grupo de 20 alunos. Portanto, os novos resultados quanto à aplicação da metodologia de ensino serão baseados em um grupo de alunos bastante relevante, totalizando 33 integrantes do ensino médio, além dos autores desse artigo, dos monitores (i.e. alunos de graduação, alguns bolsistas e outros voluntários), e dos professores do ensino médio. Esperam-se com isso obter melhorias na metodologia, visando à adequação da proposta e um levantamento mais concreto sobre as reais contribuições e limitações da mesma.

Referências

1. Plataforma Alice/Carnegie Mellon University, http://www.aliceprogramming.net
2. Ambiente Scratch/Massachusetts Institute of Technology, http://scratch.mit.edu
3. Chen, M., Lucas, G.: Education Nation: Six Leading Edges of Innovation in Our Schools. ISBN: 978-1-1181-5740-4, 320 pages, January (2012)
4. Piaget, J., Duckworth, E.: Piaget takes a teacher's look. In: Learning: The magazine for creative teaching, vol. 2, n. 2, p. 22-27 (1973)
5. Papert, S.: Mindstorms: children, computers, and powerful ideas. Books (1980)
6. Freinet, E.: O itinerário de Célestin Freinet: a livre expressão na pedagogia de Freinet. Rio de Janeiro (1979)
7. Denis, B., Leclercq, D.: The fundamental I.D.'s and their associated problems. In: 1st. Workshop of the Special Interest Group on Instructional Design of EARLI, pp. 67–83 (1994)
8. Leclercq, D., Denis, B.: Méthodes de formation et psychologie de l'apprentissage. Service de Technologie de l'Education de l'Universite´ de Liège (1998)
9. Destacom/Universidade Federal do Mato Grosso do Sul, http://destacom.ufms.br
10. Mylopoulos, J.: Goal-Oriented Requirements Engineering. In: XI Conference Iberoamericana de Software Engineering, pp. 13-17 (2008)
11. Yu, E.: Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. In: 3rd International Symposium on Requirements Engineering (RE'97), pp. 226-235, January (1997)
12. Leite, J.C.S.P., Rossi, G., Balaguer, F., Maiorana, V., Kaplan, G., Hadad, G., Oliveros, A.: Enhancing a Requirements Baseline with Scenarios. Requirements Engineering Journal, 2(4), pp. 184-198 (1997)

Abordagem para Reuso de Requisitos Tardios em Sistemas de Informação

Mauricio Manoel C. Junior, Maria Lencastre, João Araújo

Polytechnic School of Engineering, University of Pernambuco (UPE)
mauriciomanoel@gmail.com, mlpm@ecomp.poli.br, ja@di.fct.unl.pt

Abstract. The reuse of use case specifications within a specific area of knowledge, rather than building them from scratch, is a strategy that not only can increase the quality of the new specifications but also can help their construction. This paper presents a current and ongoing research related with the reuse in Requirements Engineering. This includes a survey on the industry, a method and a web-based tool whose objective is to provide support for reuse in the Requirements community.

Keywords. Requirements Engineering, Use Cases, Requirements Reuse, Requirements Visualization, Traceability.

1 Introdução

Durante as fases do desenvolvimento de *software*, pode-se observar que alguns artefatos gerados (especificação, modelos de análise, arquitetura, projeto, ou parte do código) são similares, ou até mesmo iguais a muitos confeccionados em outros projetos. Esses mesmos artefatos podem ser reusados para construção de novos *softwares*, diminuindo o trabalho realizado durante o desenvolvimento de um novo produto [1].

A Engenharia de Software tem investido esforços para facilitar a construção de produtos a partir do uso de experiência e documentação previamente desenvolvida, testada e utilizada. Assim, como resultado da busca por produzir softwares com alta qualidade, incluindo a confiabilidade, surgiu o desenvolvimento baseado em reuso [2].

O conceito de reuso consiste em utilizar artefatos de software existentes para o desenvolvimento de um novo sistema. O reuso pode ser aplicado a qualquer etapa do ciclo de vida do software, podendo-se fazer reuso de diversos artefatos, desde documentos de requisitos, especificações de sistemas, modelos, projetos arquiteturais, até ao reuso do próprio código fonte [2]. Para ser efetivo, o reuso de artefatos de software exige a criação e manutenção dos mesmos, além de ferramentas que facilitem todas as atividades envolvidas no processo [3]. Porém, ainda existe uma grande carência nessa área, sendo relevante o investimento na definição e criação de métodos e ferramentas que tornem o reuso uma técnica mais conveniente, e até mesmo mais estimulante [4].

Na ER é possível reutilizar o conhecimento, experiência e artefatos de projetos já existentes [5], porém apesar dos diferentes esforços realizados e dos resultados já obtidos, ainda existe carência nessa área [6]. Este trabalho é motivado por essa realidade, sendo seu foco voltado para reuso em ER, com ênfase na fase de especificação de requisitos baseada nos casos de uso, que se enquadra na etapa dos requisitos tardios (RT) [7]..O RT foca nos requisitos funcionais e não-funcionais de um sistema, com suas funções e qualidades relevantes ao *software* [7]. Neste trabalho, assumiu-se que para cada caso de uso existe associado um objetivo principal, que representa o que o usuário pretende alcançar [8].

Um caso de uso é um meio de especificar e capturar requisitos de um sistema, definindo seu comportamento de acordo com as necessidades dos usuários ou outras entidades que interagem com o sistema, representados como atores [9]. A especificação de casos de uso é um processo extremamente repetitivo [10]. Um exemplo de repetição são os diferentes sistemas que incluem a funcionalidade *reserva* de um objeto, entre eles: reserva de carro, reserva de bicicleta, reserva de equipamentos eletrônicos, reserva de equipamentos para construção civil. Estes sistemas incluem casos de uso comuns como: autenticar cliente ou operador, cadastrar cliente, cadastrar reserva, cadastrar locação e pagamento da reserva. Assim, artefatos relacionados a esta funcionalidade podem ser reusados facilmente entre esses projetos. Porém, geralmente, isso não ocorre por falta de formas de sistematização das técnicas de reuso, ausência de ferramentas que o auxiliem, e ausência de repositórios para compartilhamento desses artefatos. Em virtude desses fatores, frequentemente esta tarefa torna-se manual, sendo necessário, para cada projeto, a mesma especificação de artefatos já especificados.

Diversos trabalhos na literatura focam o uso de especificação de casos de uso para reuso, utilizando diferentes técnicas e tipos de abstrações. Entre eles podem-se destacar os trabalhos de: Modelagem de Variabilidade de Cenário como Mecanismos Transversais (MSVCM) [11]; Fragmentos de Requisitos [12]; Especificações de Casos de Uso para LPS Baseada em Fragmentos [13]; e, Abordagem de Caso de Uso Orientada por Padrões [3]. Observando as abordagens existentes, não se encontra uma estratégia integrada que contemple as diferentes etapas. Faltam propostas que contemplem desde a etapa de proposta e abstração de formas de reuso em casos de uso, considerando templates, tags (textos repetitivos) em um mesmo caso de uso e/ou, variabilidade relacionada a um caso de uso, até ao reuso de fragmentos de casos de uso disponíveis através de um catálogo.

Nesse contexto, pode-se caracterizar a questão tratada neste trabalho: "*Como dar suporte a especificações de casos de uso, tomando como base o conceito de reuso (reaproveitando o que já foi pensado, escrito, e muitas vezes testado), permitindo assim redução do esforço na escrita de elementos repetitivos em um mesmo projeto e entre projetos diferentes*?".

O restante deste artigo está organizado da seguinte forma. Seção 2 discute os objetivos da pesquisa, e os conceitos e abordagens aplicados neste trabalho. Seção 3 apresenta as constribuiçoes cientificas deste trabalho. Seção 4 apresenta as conclusões deste trabalho e as observações feitas na pesquisa.Por fim, a seção 5 apresenta trabalhos futuros .

2 Objetivos da Pesquisa

O objetivo desta pesquisa é auxiliar na especificação de casos de uso, tornando-a uma tarefa menos repetitiva, e dando suporte para que ela seja baseada em conhecimento e especificação prévia. Foram tomadas como inspiração as técnicas de reuso propostas por Dias [14], Bonifácio [11] e Issa e Alali [3], que vêm sendo promissoras no sentido de redução de esforço. Assim, este trabalho busca facilitar as atividades vivenciadas por gerentes de projetos e analistas, tanto na criação dos casos de uso, quanto na manutenção dos mesmos.

Como solução foi proposto um método [15,16], chamado M-4REuse, e foi desenvolvida uma ferramenta de suporte ao reuso na ER, o 4REuse [17]. Ambas visam apoiar na especificação de casos de uso a partir do reuso, e ao mesmo tempo, auxiliar e estimular a adoção desta prática, contribuindo na redução das dificuldades, comumente encontradas.

Para descrever a solução proposta, alguns objetivos específicos foram considerados:

1. Mapeamento de como é feito o levantamento de requisitos e sua aplicação com reuso em empresas do Recife;
2. Proposta de uma abordagem de especificação de casos de uso tomando como base o conceito de reuso, que seja facilmente, integrável às abordagens de fragmentos de caso de uso [14], modelagem de variabilidade [11] e catálogo de padrões. [3];
3. Proposta e disponibilização na *web* de uma ferramenta protótipo, que auxilia na especificação de novos projetos, considerando a definição e reuso de um catálogo de fragmentos de casos de uso;
4. Apresentação a especialistas do mercado a ferramenta *4REuse*;
5. Verificação e análise da ferramenta através de diversos estudos realizados com alunos da POLI-UPE, e;
6. Sugestões de melhorias nos processos de especificação de casos de uso e na construção de um catálogo de artefatos reusáveis.

3 Contribuições Científicas

Como principais contribuições associadas a este trabalho pode-se assinalar:

1. A proposta de um método de especificação de requisitos e ferramenta baseado em reuso, que visam minimizar o problema de esforço na especificação de casos de uso;
2. A aplicação do uso da ferramenta no meio acadêmico, de uma forma planejada, de modo a permitir uma análise mais concreta e evoluções da mesma;
3. Obtenção de indícios, através do uso no mercado e do experimento de *software* realizado, que a ferramenta 4REuse reduz o esforço da especificação de casos de uso;
4. A integração entre ferramentas, permitindo a geração de casos de teste a partir de casos de uso e rastreabilidade;
5. Elaboração do catálogo de fragmentos de casos de uso;

6. Publicação de artigo científico na 16th IASTED *International Conference on Software Engineering and Applications* - SEA 2012.
7. Ferramenta 4REuse foi aplicada em um experimento simplificado realizada no meio acadêmico, visando avaliar a usabilidade da ferramenta, considerando os atribuídos de Intuitividade, Operacionalidade, Eficiência de Uso, Aprendizagem, Atratividade e Satisfação. O resultado mostrou que 67,33% concordaram com a ferramenta auxilia na especificação de casos de uso [18].

4 Conclusões

É de amplo conhecimento na comunidade de Engenharia de Software que o processo de Engenharia de Requisitos é repleto de dificuldades, normalmente ligadas a requisitos especificados de forma incompleta, ao pouco conhecimento sobre o domínio no qual ele se aplica, ou relacionadas ao pouco conhecimento sobre as reais necessidades dos usuários. Essas dificuldades tornam-se uma barreira para o sucesso de um projeto, isto porque o sucesso de um sistema de *software* é decorrente do cumprimento das finalidades para o qual o sistema é desenvolvido. Diversos autores ressaltam também a importância da Engenharia de Requisitos e seus impactos, principalmente, em relação aos custos e tempo de desenvolvimento dos projetos. Neste cenário, o reuso de artefatos de requisitos é visto como uma alternativa para ajudar nas atividades de Engenharia de Requisitos, a fim de diminuir o impacto causado pelas dificuldades destacadas. A adoção de uma abordagem de reuso tem potencial para proporcionar redução no tempo de construção, pois considera que não é necessário sempre fazer do zero, e não é necessário fazer novamente; além disso, o reuso pode aumentar a qualidade do produto desenvolvido, pois só se reusa o que é bom e, consequentemente, pode gerar redução do custo no software.

Neste trabalho, técnicas de reuso foram aplicadas para auxiliar na especificação de casos de uso, uma vez que esta tarefa é algo repetitivo e custoso para muitas empresas. Com essa motivação, este trabalho propôs uma abordagem de reuso de requisitos (um método e uma ferramenta) para especificação de casos de uso, baseado em técnicas de reuso. A principal finalidade foi auxiliar a atividade de especificação de casos de uso.

Para a ferramenta proposta, foi realizada uma avaliação da usabilidade na ferramenta 4REuse, que incluiu a realização de um estudo empírico, levando em consideração os atributos relativos à intuitividade, operacionalidade, eficiência de uso, aprendizagem, atratividade e satisfação do usuário.

Analisando-se os resultados de forma geral, observa-se que 67,33% das respostas da pesquisa foram atribuídas às alternativas que indicam a concordância dos participantes em relação às questões afirmativas apresentadas da usabilidade da ferramenta. Por outro lado, 13,11% das respostas foram atribuídas às alternativas que caracterizam discordância dos participantes. É importante ressaltar que 19,56% das respostas da pesquisa foram destinadas à alternativa “neutro”, o que indica o índice de neutralidade dos participantes em relação às questões apresentadas.

5 Trabalho Futuro e em Andamento

Como recomendações e sugestões de trabalhos para evoluir a presente pesquisa pode se destacar:

- Comparar a ferramenta seguindo o mesmo protocolo do experimento de *software* apresentado neste trabalho, objetivando aumentar a quantidade de amostras. Aumentando as amostras poderá se avaliar estatisticamente se a ferramenta criada possui usabilidade aceitável se comparado a outras usadas na indústria;
- Avaliar o método proposto, para que seja possível verificar se o método *M-4REuse* auxilia efetivamente na construção para reuso dentro do catálogo;
- Avaliar o desenvolvimento com reuso e o uso da ferramenta em uma empresa que trabalhe com metodologias ágeis, medindo o desempenho de especificação de requisitos de *software* e se haverá impactos positivos para quem utiliza *Test DrivenDevelopment* - TDD [19] em seu desenvolvimento.
- Classificar os fragmentos de casos de uso, de acordo com domínios, para que seja possível melhorar a busca e mostrar ao usuário a capacidade de reuso do catálogo;
- Procurar por outras formas de melhoria na construção do catalogo de fragmentos de casos de uso e na especificação de casos de uso com reuso, analisando na literatura trabalhos relacionados à construção de catalogo de requisitos e a especificação de casos de uso para reuso.
- Desenvolver novas funcionalidades na ferramenta 4REuse para fornecer o uso de métricas de suporte a estimativas. Por exemplo, relacionando-se cada requisito do repositório com seus dados históricos, permitindo-se o cálculo do esforço de programação, prazos e custos, auxiliando com informações para estimativas de tamanho, produtividade e alocação de pessoas.
- Desenvolver funcionalidades na ferramenta 4REuse relacionadas à moderação na inclusão de requisitos no repositório, para verificar o conteúdo dos requisitos que serão compartilhados para reuso.
- Desenvolver funcionalidades na ferramenta 4REuse ligadas à prototipação de telas a partir da estrutura de dados e regras de negócio.
- Evoluir a ferramenta para que ela continue sendo colaborativa, criando novos módulos que facilitem a comunicação e validação da mesma pelos analistas, desenvolvedores e usuários;
- Realizar novos experimentos visando verificar a eficiência e a eficácia da abordagem proposta em empresas de desenvolvimento de *software*.

6 Referências

[1] ROSSI, A. C. **Representação do componente de software na FARCSoft: ferramenta de apoio à reutilização de componentes de software**. Dissertação (Mestrado em Sistemas Digitais) - Escola Politécnica da Universidade de São Paulo, Departamento de Engenharia de Computação e Sistemas Digitais - São Paulo. 2004.

[2] KRUEGER, C. W. Software Reuse. **ACM Computing Surveys**, v. 24, n. 2, p. 131-183, doi:10.1145/130844.130856, 1992.
[3] ISSA, A. A. e ALALI, A. I. Automated requirements engineering: Use case patterns-driven approach. **Software, IET**, v. 5, n. 3, p. 287-303, doi:10.1049/iet-sen.2010.0014, 2011.
[4] THAKARE, A. B. T. e M., V. A Study of Sofware Reuse and Models. **IJCA Proceedings on National Conference on Innovative Paradigms in Engineering and Technology (NCIPET 2012)**, n. 15, p. 14-17, 2012.
[5] FRAKES, W. B. e KANG, K. Software Reuse Research: Status and Future. **IEEE Trans. Softw. Eng.**, v. 31, n. 7, p. 529-536, doi:10.1109/TSE.2005.85, 2005.
[6] CHENG, B. H. C. e ATLEE, J. M. Research Directions in Requirements Engineering. In: FUTURE OF SOFTWARE ENGINEERING (FOSE '07). **Anais**... [S.l.]: IEEE. Disponível em: <http://dl.acm.org/citation.cfm?id=1253532.1254725>. Acesso em: 25 nov. 2012, 2007.
[7] CASTRO, J.;; KOLP, M. e MYLOPOULOS, J. Towards requirements-driven information systems engineering: the Tropos project. **Inf. Syst.**, v. 27, n. 6, p. 365-389, doi:10.1016/S0306-4379(02)00012-1, 2002.
[8] SANT, V. F. A. e CASTRO, J. F. B. Developing Use Cases from Organizational Modeling. **IV Workshop de Engenharia de Requisitos**, 2010.
[9] OMG. **Unified Modeling Language Specification**. Disponível em: <http://www.omg.org/spec/UML/>. Acesso em: 1 dez. 2012.
[10] GARCÍA, J. D. et al. Specifying use case behavior with interaction models. **JOURNAL OF OBJECT TECHNOLOGY**, v. 2, 2003.
[11] BONIFACIO, R. **Modeling Software Product Line Variability in Use Case Scenarios An Approach Based on Crosscutting Mechanisms**. Tese (Doutorado) - Universidade Federal de Pernambuco, Cin, Centro de Informática - Pernambuco. 2010.
[12] ARAÚJO, A. R. **Um Método para a Criação de Fragmentos de Requisitos para Reuso em Sistema de Informação**. Dissertação (Mestrado em Engenharia da Computação) - Universidade de Pernambuco, Escola Politécnica - Pernambuco. 2011.
[13] ARAÚJO, D. O. **Elaboração de especificações de casos de uso para linhas de produto de software baseada em fragmentos**. Dissertação (Mestrado em Informática) - Universidade Federal do Rio de Janeiro, Instituto de Matemática, Núcleo de Computação Eletrônica - Rio de Janeiro. 2010.
[14] DIAS, F. G. **Elaboração de Requisitos de Software: uma Abordagem baseada em Fragmentos de Casos de Uso**. Dissertação (Mestrado em Informática) - Universidade Federal do Rio de Janeiro, UFRJ - Rio de Janeiro. 2008.
[15] MANOEL, M. C. J. et al. Reusing Use Cases Specification in Information Systems. **16th IASTED International Conference on Software Engineering and Applications.**, doi:10.2316/P.2012.790-025, 2012.

[16] MANOEL, M. C. J. **4REuse: Um Método e uma Ferramenta para o Apoio à Especificação de Requisitos baseado em Reuso**. Dissertação (Mestrado em Engenharia da Computação) - Universidade de Pernambuco, Escola Politécnica - Pernambuco. 2012.
[17] **4REuse Tools**. Disponível em: <http://www.4reuse.info/>. Acesso em: 3 jun. 2013.
[18] MANOEL, M. C. J. **4REuse: um Método e uma Ferramenta para o Apoio à Especificação de Requisitos baseado em Reuso**. Disponível em: <http://goo.gl/NJqZj>. Acesso em: 1 jun. 2013.
[19] BECK. **Test Driven Development: By Example**. 1. ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002.

Um processo para construção de software mais transparente

Eduardo Almentero[1], and Julio Cesar Sampaio do Prado Leite[1]

[1]Pontifícia Universidade Católica do Rio de Janeiro, PUC - Rio, Brasil

{ealmentero, julio}@inf.puc-rio.br

Resumo. À medida que o software está cada vez mais presente em nossas vidas, se tornando parte de nosso cotidiano, a sociedade também exigirá sua transparência. Cabe, portanto à engenharia de software, como disciplina, propor métodos e ferramentas para apoiar esta nova demanda. Um software é transparente quando manipula informações transparentes e informa sobre si mesmo, como funciona, o que faz e porque o faz. Devido à complexidade deste novo requisito, ele foi entendido como um conjunto de qualidades que contribuem individualmente para transparência. Neste trabalho, procuramos abordar o requisito de transparência sob a ótica da simbiose entre modelos de requisitos e artefatos de arquitetura e código, com o objetivo de abranger um maior conjunto das qualidades exigidas para tornar o software transparente.

Palavras-chave: transparência de software, rastreabilidade, requisitos, cenários, arquitetura, documentação.

1 Introdução

À medida que o software está cada vez mais presente em nossas vidas, se tornando parte de nosso cotidiano, a sociedade também exigirá sua transparência, principalmente daqueles que utilizamos com mais freqüência e que tem grande importância para nós, como por exemplo, os sistemas bancários. Atualmente, sabemos que a comunicação feita através do software é obscura, porém é essencial que o software em si seja totalmente transparente, para que todos possam ter o conhecimento sobre exatamente o que ele faz e como faz.

Um software é transparente quando manipula informações transparentes (transparência da informação) e informa sobre si mesmo, como funciona, o que faz e porque o faz (transparência do processo) [1]. Segundo Leite [2], a necessidade de transparência terá um grande impacto no processo de desenvolvimento de software. Portanto, é necessário que pesquisas sejam realizadas no sentido de atender esta demanda para aqueles que usam o software ou são de alguma forma afetados por ele.

O grupo de engenharia de requisitos da PUC-Rio (Grupo ER) [7] é pioneiro na pesquisa de transparência de software. Nesse sentido, o grupo entendeu que o conceito de transparência era muito complexo e abstrato, mas que poderia ser mais bem entendido como um conjunto de qualidades (metas flexíveis) que contribuem individualmente para a transparência. A fim de definir que qualidades deveriam ser consideradas e de que forma contribuiriam para transparência, realizamos um extensivo "survey" e validações guiadas por questionário. O NFR framework foi utilizado para modelar a rede de metas flexíveis criada para definir a transparência. Esta rede é composta por 27 nós folha e quatro grupos [3]. Cada nó é uma característica ou meta flexível necessária para alcançarmos a transparência.

Em particular, no trabalho [4] evoluído posteriormente em [5] requisitos modelados através de cenários foram associados ao código fonte, de forma que este se tornasse mais fácil de ser entendido, aumentando assim sua transparência. Cenário é a descrição de situações comuns ao cotidiano [6], ou seja, é uma técnica de descrição que auxilia a compreensão de uma situação específica de um software, priorizando seu comportamento. Os cenários devem abordar aspectos de usabilidade, permitir um maior detalhamento do conhecimento do problema, a unificação de critérios, a obtenção do compromisso de usuários, a organização de detalhes e o treinamento de pessoal. Escolhemos uma abordagem baseada em cenários porque quando utilizados na engenharia de requisitos, os cenários ajudam tanto na compreensão do sistema que será desenvolvido quanto na validação do conhecimento do engenheiro pelo cliente. Isto se dá pelo fato de que os cenários são descritos em linguagem natural e modelam situações do sistema, fazendo com que clientes se sintam mais à vontade e entendam melhor o processo de descobrimento dos requisitos.

Como a transparência impacta o processo de desenvolvimento de software como um todo, sabemos que é essencial não limitá-la apenas ao artefato de código fonte. Desta forma, utilizamos a experiência adquirida na definição da transparência de software, com objetivo de viabilizar a transparência do processo e dos artefatos produzidos durante o desenvolvimento do software. **Nossa principal estratégia é a associação de modelos de requisitos aos artefatos**, de forma que seja mantida a rastreabilidade entre eles e entre os artefatos produzidos em cada fase, durante o ciclo de desenvolvimento do software. Desta forma, procuramos atender às diferentes qualidades requeridas para se obter um software transparente.

Este artigo está organizado da seguinte forma: na seção 2 são apresentados os objetivos delineados para a pesquisa. Na seção 3 descrevemos as contribuições alcançadas com o trabalho realizado. As conclusões obtidas durante a realização da pesquisa são discutidas na seção 4. E por fim, na seção 5, os trabalhados que estão sendo realizados atualmente e os futuros são detalhados.

2 Objetivos da Pesquisa

O presente trabalho de pesquisa tem como objetivo principal investigar e definir uma estratégia para alcançar a transparência de um software. Tal objetivo implica na adequação do processo de desenvolvimento, de forma que em cada fase a transparência seja abordada com o uso de modelos de requisitos associados aos artefatos produzidos.

A primeira etapa investigada foi a fase de definição da arquitetura do software. Esta arquitetura será composta de cenários organizados em grupos. Tais grupos serão identificados a partir de uma análise do léxico ampliado da linguagem (LAL) [9], que é uma técnica baseada na descrição de termos através de noção e impacto. Esta técnica foi criada para ajudar no entendimento da linguagem específica do domínio, contribuindo desta forma para o entendimento do domínio como um todo. Esta análise se baseia na idéia que termos muito utilizados dentro do domínio representam conceitos importantes, e podem ser aproveitados para organizar as situações do sistema. A inclusão dos cenários no grupo se baseia na comparação dos elementos que compõem sua descrição com a definição do termo que representa o grupo. A Fig. 1exibe o processo de divisão dos cenários em grupos.

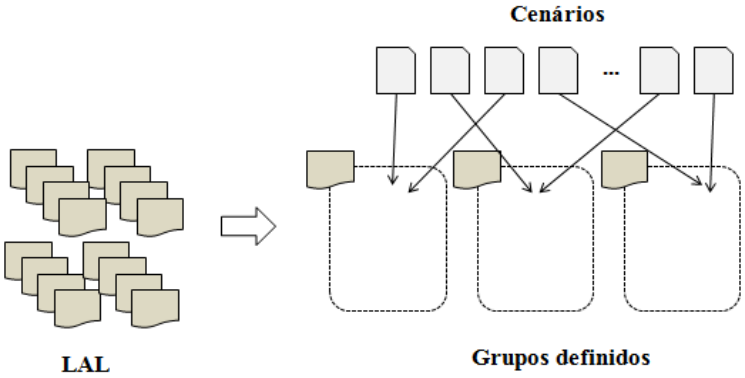


Fig. 1. Processo de organização dos cenários em grupos

Com base nesta organização inicial, propomos a divisão dos cenários em camadas, de acordo com o framework MVC [10]. Esta divisão visa à separação da interface do comportamento do sistema, permitindo um melhor entendimento de ambos, além de estabelecer os cenários de controle, que serão responsáveis por administrar o fluxo de informações do sistema e aplicar as regras de negócio. Após a divisão, os cenários de cada camada são detalhados de acordo com uma série de heurísticas já estabelecidas e são posteriormente operacionalizados, de forma que os cenários fiquem associados ao código, formando um só documento.

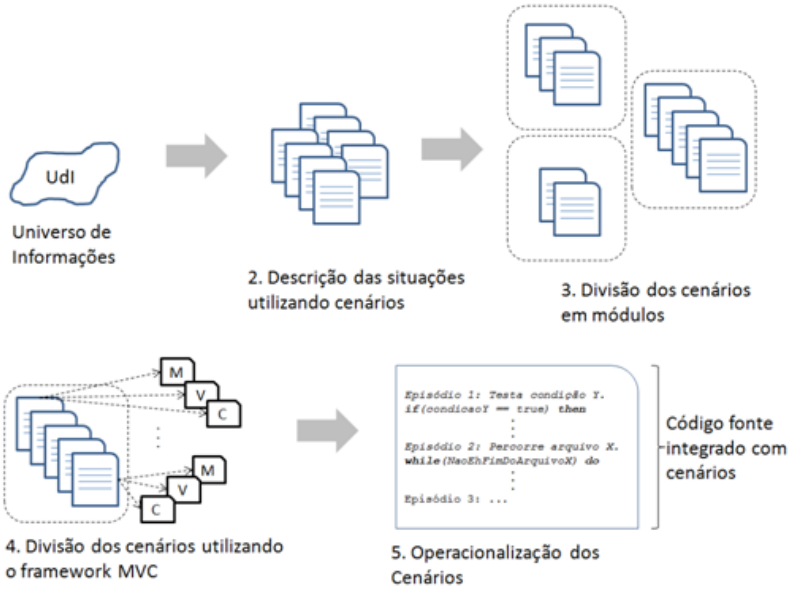


Fig. 2. Processo de desenvolvimento com viés de transparência

Este processo permite a rastreabilidade do código para os cenários que o documentam e destes para os cenários da arquitetura, que por sua vez podem ser imediatamente relacionados aos cenários iniciais, isto é, a especificação de requisitos do sistema, como mostra a Fig. 2. Diante de tal estrutura, toda alteração, tanto da dos cenários utilizados como documentação quanto do código, pode ser propagada facilmente, ajudando a evitar a erosão da documentação.

3 Contribuições Científicas

Acreditamos que a grande contribuição deste trabalho é a definição de um processo com passos bem definidos, que pode ser facilmente transformado em um processo semiautomático, facilitando a construção de software mais transparente. O processo proposto engloba a definição da arquitetura de forma guiada a partir do LAL e propõe sua descrição através de cenários. A abordagem de cenários escolhida permite a descrição de requisitos não funcionais. Desta forma conseguimos tratar outro problema interessante, que é a descrição destes requisitos no documento de arquitetura. O detalhamento destes cenários evolui durante o processo de desenvolvimento, proporcionando uma divisão conceitual da arquitetura com base no framework MVC, até chegar à efetiva implementação do software. Como todo o processo é essencialmente baseado na evolução de cenários, e estes artefatos, quando evoluídos, mantêm naturalmente um elo com os anteriores, temos a possibilidade de rastrear o impacto de

cada decisão até o código fonte e também seguir o sentido oposto. Tendo associados a estes cenários o código e a arquitetura, dispomos do rastro da arquitetura para o código e vice versa.

Outra importante contribuição foi a evolução da plataforma C&L [8] e construção de uma linha de produtos para bibliotecas digitais, ambas guiadas por qualidades de transparência. Esta evolução do C&L iniciou sua transformação em uma plataforma de desenvolvimento de software transparente. Os esforços deste trabalho visam à evolução desta ferramenta para atender cada vez mais as necessidades de desenvolvedores e outros interessados no desenvolvimento de software transparente. Os trabalhos realizados pelo grupo neste sentido incentivam outros que também contribuem para o enriquecimento da plataforma.

4 Conclusões

Neste trabalho abordamos uma estratégia para possibilitar o desenvolvimento de software mais transparente. Embora mais experimentos sejam necessários, durante a aplicação deste método em sistemas simples, como o de uma biblioteca digital, tivemos indícios de que o software produzido realmente atende uma parcela significativa das qualidades necessárias para a transparência, nos permitindo dizer que o tornamos mais transparente. Além disto, percebemos que a manutenção da documentação do software se tornou mais simples, devido à sua proximidade do código. E por fim, outro benefício foi a possibilidade de especificar o impacto de requisitos não funcionais, devido ao uso de cenários, já na descrição da arquitetura do sistema.

5 Trabalhos em Andamento e Futuros

Atualmente trabalhamos na criação de um experimento para a aplicação da estratégia proposta na prática. O desenho deste experimento visa identificar a complexidade de aplicação da solução e de que forma ela contribui para a manutenção do software e de sua documentação. Desejamos também verificar de que forma um software desenvolvido de acordo com tal proposta atende a alguns requisitos de qualidade essenciais para que a transparência seja alcançada, como entendimento e leiturabilidade, por exemplo.

Como trabalho futuro, pretendemos evoluir a ferramenta C&L de forma que ela dê suporte à aplicação do método, automatizando as etapas com regras bem definidas, como a identificação de grupos a partir do LAL e organização de cenários, e facilitando as etapas mais complexas, como a divisão dos cenários em camadas e seu detalhamento. Desejamos também evoluir a ferramenta para que ajude na evolução dos cenários associados ao código fonte, de forma que seja mais fácil evitar a erosão da documentação devido à evolução dos requisitos do software. A disponibilização de uma ferramenta para facilitar a utilização das abordagens propostas certamente irá atrair mais atenção ao uso de técnicas para construção de software mais transparente.

6 Referências

1. Leite, J. C. P., Cappelli, C.; Exploring i* Chracteristics that Support Software Transparency. In Proceedings, Vol. 322, 2008, PP.51-54.
2. http://amazinnggg.blogspot.com/2006/07/software-transparency.html. Acessado em 10/03/2013.
3. Cappelli, C.; An Approach for Business Processes Transparency Using Aspects. Tese de Doutorado. Departamento de Informática, PUC-Rio, 2009.
4. Silva, Lyrene Fernandes da, Sayão, Miriam, Leite, Julio Cesar S. P., Breitman, Karin Koogan; Enriquecendo o Código com Cenários; XVII Brazilian Symposium on Software Engineering (SBES2003), Manaus-AM, 2003.
5. Almentero, E. ; Re-engenharia do Software C&L para Plataforma Lua-Kepler utilizando princípios de transparência. Dissertação de Mestrado. Departamento de Informática, PUC-Rio, 2009.
6. Leite, J.C.S.P, Rossi, G., Balaguer, F., Maiorana, V., Enhancing a requirements baseline with scenarios; In: Third IEEE International Symposium on Requirements Engineering - RE97, Proceedings. IEEE Computer Society Press, January, 1997, pp 44-53.
7. URL: http://www-di.inf.puc-rio.br/~julio/Slct-pub/transp-sbes.pdf
   Grupo de Pesquisas em Engenharia de Requisitos da Puc-Rio. Disponível em: http://transparencia.inf.puc-rio.br/wiki/index.php/Integrantes. Acessado em 10/03/2013.
8. C&L – Cenários e Léxicos – Disponível em: http://pes.inf.puc-rio.br/cel. Acesso em: 12/03/2013.
9. Leite, JCS do, and Ana PM Franco. "A strategy for conceptual model acquisition." Requirements Engineering, 1993., Proceedings of IEEE International Symposium on. IEEE, 1993.
10. Krasner, Glenn E., and Stephen T. Pope. "A description of the model-view-controller user interface paradigm in the smalltalk-80 system." Journal of object oriented programming 1.3 (1988): 26-49.

Usando Modelos de Requisitos em Tempo de Execução: Potencial e Desafios

Vítor E. Silva Souza and Renata Guizzardi

Núcleo de Estudos em Modelagem Conceitual e Ontologias (Nemo)
Departamento de Informática, Universidade Federal do Espírito Santo (Ufes)
Vitória, ES, Brasil
{vitorsouza,rguizzardi}@inf.ufes.br

Resumo Pesquisadores têm cada vez mais direcionado sua atenção ao uso de modelos em tempo de execução (*runtime*), provendo ferramentas e *frameworks* que auxiliam os desenvolvedores na tarefa de construir software alinhado a seus requisitos/arquitetura. Em particular, algumas pesquisas em Engenharia de Requisitos concentraram-se em desenvolver sistemas de software que possuam a capacidade de ler seus próprios modelos de requisitos e tomar decisões a partir de uma análise do mesmo. É uma tendência comum, por exemplo, na área de sistemas adaptativos, para a qual contribuímos recentemente. Neste artigo, propomos um novo projeto de pesquisa sobre o uso de modelos de requisitos em tempo de execução, discutindo seus potenciais benefícios e os desafios envolvidos.

Palavras-chave: requisitos, objetivos, modelos, tempo de execução, sistemas colaborativos de gestão de conhecimento

1 Introdução

Na área de modelagem conceitual, um assunto que vem atraindo a atenção de pesquisadores é o uso de modelos em tempo de execução. Esta tendência pode ser observada em artigos que estabelecem agendas de pesquisa e na criação de *workshops* específicos sobre o tema.[1] No contexto da Engenharia de Requisitos, por exemplo, a ideia é fazer com que o sistema de software leia seu próprio modelo de requisitos e tome decisões em tempo de execução baseado nele.

Um dos autores deste artigo propôs recentemente uma abordagem baseada em requisitos para o desenvolvimento de sistemas adaptativos [4]. A proposta, batizada de *Zanshin*, inclui também um *framework* que utiliza modelos em tempo de execução como explicado acima: em tempo de execução, indicadores específicos do sistema são monitorados e, caso algum destes apresente problemas, o sistema é instruído sobre o que deve ser feito para adaptar-se àquela situação. Em outras palavras, o *framework* utiliza os modelos em *runtime* para implementar um *feedback loop* que provê habilidades de auto-adaptação ao sistema.

Para representar requisitos, utilizamos modelos que incluem objetivos (*hard* e *soft*), tarefas, suposições de domínio (o que se assume ser verdadeiro) e restrições

[1] Veja, por exemplo, http://www.comp.lancs.ac.uk/~bencomo/WorkshopMRT.html.

de qualidade (critérios precisos para satisfação de objetivos *soft*). A abordagem *Zanshin* complementa estes modelos com requisitos do *feedback loop*, a saber: "Requisitos de Percepção" (*Awareness Requirements* ou *AwReqs*, indicam o que monitorar) [6] e "Requisitos de Evolução" (*Evolution Requirements* ou *EvoReqs*, especificam o que fazer para adaptar-se) [5].

Nosso objetivo é focar em domínios de aplicação que possam se beneficiar do uso de modelos em tempo de execução. Dentre eles, destacamos a área de Gestão Colaborativa do Conhecimento (GCC). Sistemas desenvolvidos para dar suporte a GCC geralmente são complexos, pois devem permitir que usuários de diferentes perfis e níveis de proficiência interajam, ao mesmo tempo gerenciando os itens de conhecimento pessoais de cada um destes usuários e/ou comunidades de usuários. Portanto, a capacidade de ajustar os requisitos em tempo de execução daria mais flexibilidade a sistemas deste tipo, tornando-os mais capazes de lidar com a dinâmica do ambiente de compartilhamento de conhecimento colaborativo.

Este projeto tem como meta o desenvolvimento de uma metodologia (possivelmente baseada em $i^\star$), seguindo a linha de Souza [4] para desenvolver sistemas que utilizem modelos de requisitos em tempo de execução. Como campo de experimentação para as futuras propostas deste projeto, analisaremos trabalhos anteriores recentes na área de sistemas de GCC, procurando entender que tipo de suporte metodológico devemos desenvolver para novas versões do sistema.

Especificamente, com relação à metodologia existente, nosso objetivo é sanar algumas das limitações existentes na abordagem *Zanshin*; além de estudar de forma mais aprofundada a semântica dos diferentes elementos sintáticos utilizados em nossos modelos de objetivos nas diferentes fases do processo de software, com possível aplicação não só para sistemas adaptativos mas, em geral, para modelar sistemas que utilizam requisitos em tempo de execução.

Neste artigo, descrevemos nossa agenda de pesquisa neste tópico para o futuro próximo, apresentando potenciais benefícios do uso de modelos de requisitos em *runtime* e discutindo os desafios associados. A Seção 2 apresenta os objetivos da pesquisa. Em seguida, a Seção 3 discute as possíveis contribuições e desafios envolvidos nesta pesquisa. Finalmente, a Seção 4 apresenta as conclusões.

2 Objetivos de Pesquisa

Podemos definir o seguinte objetivo de pesquisa para nossa proposta: *Definir uma linguagem bem-fundamentada para a criação de modelos de requisitos para sistemas de software que possam ser utilizados em tempo de execução pelo próprio sistema, dando suporte à algum tipo de processo de tomada de decisão* **e** *propor um processo sistemático para a criação destes modelos em diferentes fases do processo de software.* A partir do objetivo acima, derivamos questões de pesquisa mais específicas:

Q1: *Qual é o estado-da-arte do uso de modelos de requisitos em runtime?* Estamos interessados em conhecer os tipos de modelos utilizados para representar requisitos (sintaxe e semântica), em que passos do processo eles

são utilizados e que tipo de análise e tomada de decisão é feita em tempo de execução com base nestes modelos. Algumas destas propostas podem ser utilizadas como base para as propostas seguintes desta pesquisa.

Q2: *Qual o significado por trás dos elementos de um modelo de requisitos (orientado a objetivos)?*
Queremos utilizar análise ontológica para definir um meta-modelo que possa ser utilizado como base para construção de diferentes modelos de requisitos, ou seja, em diferentes pontos do processo (análise inicial dos requisitos, especificação detalhada, projeto, etc.) até seu uso em tempo de execução. Nosso foco é nas abordagens orientadas a objetivos.

Q3: *Que tipo de suporte metodológico pode ser oferecido para engenheiros de requisitos e desenvolvedores de software em geral usando estes modelos?*
Um dos objetivos desta pesquisa é também a proposta de alguma forma de suporte metodológico, o que pode variar de simples recomendações até um completo processo sistemático para a criação dos diferentes modelos.

Q4: *Que tipo de ferramenta poderia ser oferecida para dar suporte a engenheiros de requisitos e desenvolvedores de software em geral?*
Além de suporte metodológico, ferramentas que provejam suporte para criação, derivação, verificação, etc. de modelos (ferramentas CASE) são importantes para a adoção da proposta na prática. *Frameworks* que utilizam os modelos em tempo de execução para auxiliar sistemas-base na tomada de decisão baseada em modelos são também necessários para experimentar a proposta (vide **Q5**).
Possíveis domínios para o desenvolvimento de tais *frameworks* são sistemas adaptativos [4], interoperabilidade na Web Semântica [3] ou Sistemas Colaborativos de Gestão do Conhecimento [3].

Q5: *A proposta é factível e útil na prática?*
Por fim, porém não menos importante, todas as ideias propostas devem ser validadas por meio de experimentos e estudos de caso que consistam na elicitação de requisitos e no desenvolvimento de sistemas (ou simulação dos mesmos), executando em conjunto com os *frameworks* desenvolvidos para análise dos modelos em tempo de execução.
Experimentos já desenvolvidos anteriormente podem auxiliar nesta tarefa: o clássico exemplo do sistema de agendamento de reuniões [5], o sistema de despacho de ambulâncias [6,5], o caixa eletrônico [8], o sistema de informação acadêmico [3], etc.

3 Contribuições e Desafios

Dadas as questões apresentadas na Seção 2, nosso plano de trabalho começa com a revisão bibliográfica (fase na qual nos encontramos atualmente) sobre o uso

de modelos em tempo de execução (**Q1**), seguida pela análise ontológica destes modelos e proposta de um meta-modelo para modelos de requisitos (**Q2**). Em seguida, atividades relacionadas à elaboração da metodologia (**Q3**), desenvolvimento de ferramentas (**Q4**) e experimentos (**Q5**) serão conduzidas de forma iterativa, em paralelo. desta maneira, esperamos utilizar partes da metodologia que forem sendo definidas incrementalmente para descobrir, por meio dos experimentos, que tipo de ajuste se faz necessário.

O restante desta seção descreve, em quatro partes, as contribuições e desafios relacionadas às questões de pesquisa **Q2** a **Q5**.

3.1 Uma Linguagem Bem-Fundamentada para Uso de Requisitos em Tempo de Execução (Q2)

De acordo com Franch et al. [2], o uso do *framework* $i^\star$ na prática é afetado pela falta de uma semântica uniforme que descreva os conceitos da linguagem. Isto pode ser explicado por sua popularidade entre diferentes grupos de pesquisa e a diversidade de sua aplicação, o que acaba gerando dialetos distintos. Já há algum tempo, nosso grupo de pesquisa (Nemo) está envolvido em pesquisas que buscam um modelo semântico comum para os conceitos do núcleo da linguagem, utilizando ontologias de fundamentação para justificar as escolhas feitas.

Recentemente chegamos à conclusão que a linguagem utilizada em [4] também sofre de problemas similares. Isso abre, portanto, uma oportunidade de aproveitar o trabalho feito no Nemo para propor correções nesta linguagem, de modo que se torne uma linguagem de modelagem bem-fundamentada, possivelmente com impacto positivo em sua usabilidade, adoção na prática, etc.

Esperamos também investigar trabalhos relacionadas, como as ontologias existentes para engenharia de requisitos (ex.: a que foi utilizada em *Zanshin*), o que pode ser útil para a proposta de um meta-modelo bem-fundamentado desta linguagem. O desafio aqui é a proposta de um meta-modelo que seja universalmente aceito por diferentes pesquisadores e profissionais da área.

3.2 Desenvolvimento da Metodologia (Q3)

Desenvolver uma metodologia que dê suporte ao uso de requisitos em tempo de execução poderá auxiliar profissionais a construir modelos de boa qualidade. Automatizar certas partes do processo também os libera de tarefas repetitivas. Um dos desafios é dar suporte ao usuário sem, no entanto, restringi-lo em excesso.

Revisar a literatura de áreas relacionadas também pode ser útil nesta questão para auxiliar no desenvolvimento metodológico. Por exemplo, Engenharia Orientada a Modelos (*Model-Driven Engineering*, ou MDE) pode auxiliar no desenvolvimento de ferramentas que efetuem transformação de modelos (como foi feito, por exemplo, em [7]). Naturalmente, esta tarefa está também relacionada ao desenvolvimento de ferramentas (**Q4**).

Esperamos, também, utilizar os dois sistemas de GCC (apresentados na próxima subseção) como base de testes sobre a qual poderemos validar nossas

hipóteses e intuições, possivelmente promovendo novas ideias para a metodologia em desenvolvimento. A esse respeito, apesar de focarem na mesma área, tais sistemas são bem diferentes em estrutura e funcionalidade, tendo cada um deles seus desafios em particular. A próxima subseção discute estes desafios de forma mais aprofundada.

3.3 Desenvolvimento de Ferramentas (Q4)

O desenvolvimento de ferramentas pode ser dividido em duas partes: (a) desenvolver uma ferramenta CASE que auxilie desenvolvedores na tarefa de seguir a metodologia proposta de forma a criar modelos corretos e úteis; e (b) desenvolver um *framework* que use estes modelos em tempo de execução para efetuar algum tipo de tomada de decisão. Como mencionamos anteriormente, sistemas adaptativos [4] ou Sistemas GCC são domínios nos quais temos alguma experiência, o que nos motiva a construir ferramentas para estes domínios.

Sistemas GCC auxiliam usuários na obtenção e no compartilhamento de seus itens de conhecimento pessoais, enquanto provêem também funcionalidades de interação social. Suas capacidades são alinhadas com a Gestão Construtivista de Conhecimento [1], que defende que mais atenção e cuidado deve ser dada à fonte do conhecimento em práticas e sistemas GC. Tanto Biblioref [3] quanto Trama[2] são exemplos de sistema GCC.

Biblioref é um sistema GCC que dá suporte ao compartilhamento de documentos. Neste sistema, classificação e compartilhamento de conhecimento é baseada no desenvolvimento de taxonomias individuais de cada usuário, que são mapeadas em uma taxonomia de referência. Os principais benefícios deste sistema são: (a) promover colaboração entre usuários através do acesso mútuo aos seus documentos; (b) dar autonomia ao usuário na organização do conhecimento; e (c) prover mecanismos que relacionam os diferentes esquemas de classificação dos usuários, permitindo aos mesmos localizar potenciais colaboradores a partir de inferências em cima destas relações.

Trama foi desenvolvido para dar suporte à colaboração entre os membros do Laboratório de Tecnologias de Apoio a Redes de Colaboração. Ele é baseado em um sistema de gestão de conteúdo e provê ferramentas para armazenamento de artefatos de conhecimento (ex.: relatórios, atas de reunião, artigos de pesquisa, etc.), além de ferramentas de colaboração como fóruns, bate-papo e calendário.

Na área de sistemas adaptativos, iremos focar na melhoria do *framework Zanshin*, uma vez que modelos bem-fundamentados sejam desenvolvidos utilizando a abordagem homônima (vide Seção 3.1).

3.4 Validação por Meio de Experimentos (Q5)

Um aspecto importante de qualquer proposta é a validação. Métodos de validação abrangem desde descrições simples de cenários para motivação da pesquisa até estudos de caso completos conduzidos em empresas parceiras, pesquisas

[2] `http://labtar.ufes.br/index.php?option=com_content&view=article&id=104`.

com profissionais da área, etc. Conduzir experimentos em áreas que envolvem muita participação humana, como a Engenharia de Requisitos e a GCC, é em geral um grande desafio.

Como contribuições esperamos entregar resultados empíricos que falem sobre a utilidade, factibilidade e escalabilidade de nossas propostas.

4 Conclusões

Neste artigo, apresentamos os objetivos de pesquisa e rascunhamos um plano de trabalho para o desenvolvimento de uma linguagem bem-fundamentada para modelos de requisitos que possam ser utilizados em *runtime*, além de um processo para sua criação na prática. Esta pesquisa foca na Engenharia de Requisitos Orientada a Objetivos, com particular interesse em $i^\star$ e trabalhos relacionados.

Pretendemos seguir esta agenda de pesquisa nos próximos anos e esperamos contribuir com resultados às comunidades de pesquisa em Engenharia de Requisitos, Modelagem Conceitual e Gestão do Conhecimento.

Agradecimentos

Este trabalho foi parcialmente apoiado pela FAPES (`http://www.fapes.es.gov.br`) por meio da bolsa PRONEX #52272362.

Referências

1. Guizzardi, R.S.S.: Agent-oriented Constructivist Knowledge Management. Phd thesis, University of Twente, The Netherlands (2006)
2. Guizzardi, R.S.S., Franch, X., Guizzardi, G.: Applying a foundational ontology to analyze means-end links in the i* framework. In: Proc. of the 6th International Conference on Research Challenges in Information Science. pp. 1–11. IEEE (2012)
3. Manola, R., Guizzardi, R.S.S., Gomes, R.L.: Biblioref: a semantic bibliographic reference management system. In: Companion Proc. of the 14th Brazilian Symposium on Multimedia and the Web. pp. 149–151. ACM (2008)
4. Souza, V.E.S.: Requirements-based Software System Adaptation. Phd thesis, University of Trento, Italy (2012)
5. Souza, V.E.S., Lapouchnian, A., Angelopoulos, K., Mylopoulos, J.: Requirements-driven software evolution. Computer Science - Research and Development pp. 1–19 (2012)
6. Souza, V.E.S., Lapouchnian, A., Robinson, W.N., Mylopoulos, J.: Awareness Requirements. In: Lemos, R., Giese, H., Müller, H.A., Shaw, M. (eds.) Software Engineering for Self-Adaptive Systems II, Lecture Notes in Computer Science, vol. 7475, pp. 133–161. Springer (2013)
7. Souza, V.E.S., Mazón, J.N., Garrigós, I., Trujillo, J., Mylopoulos, J.: Monitoring Strategic Goals in Data Warehouses with Awareness Requirements. In: Proc. of the 2012 ACM Symposium on Applied Computing. pp. 1075–1082. ACM (2012)
8. Tallabaci, G., Souza, V.E.S.: Engineering Adaptation with Zanshin: an Experience Report. In: Proc. of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (to appear) (2013)

Integrando Aspectos de Sustentabilidade à Engenharia de Sistemas

Camilla Bomfim[1], Wesley Nunes[1], Leticia Duboc[1],
Carina Alves[2], Xavier Franch[3] e Renata Gizzardi[4]

[1] IME, Universidade do Estado do Rio de Janeiro, Brasil
{camillajbomfim, aleysenun }@gmail.com, leticia@ime.uerj.br,
[2] CIN Universidade Federal de Pernambuco, Pernambuco, Brasil
cfa@cin.ufpe.br
[3] ESSI, Universitat Politecnica de Catalunya, España
franch@essi.upc.edu
[4] Centro de Informática Universidade Federal do Espírito SantoVitória, Brasil
rguizzardi@inf.ufes.br

Resumo. Sustentabilidade é uma das principais forças motoras de nossa sociedade. Entre as muitas iniciativas para alcançar esta meta estão as de TI, que se preocupam principalmente com o consumo responsável de recursos durante o desenvolvimento e a operação de sistemas de software. Software, no entanto, é parte de um contexto mais amplo, sistemas sócio-técnicos, cujo projeto tem uma grande influência no consumo de recursos. Uma maior compreensão do papel da sustentabilidade e a incorporação deste aspecto à engenharia de sistemas sócio-técnicos pode representar uma importante contribuição para vencer este desafio. Este artigo descreve a proposta de uma pesquisa para compreender o estado-da-prática e desenvolver métodos e técnicas que deem suporte aos diferentes aspectos de sustentabilidade durante a engenharia de sistemas.

Palavras-chave: sustentabilidade, engenharia de sistemas, sistemas sócio-técnicos

1 Introdução

Sustentabilidade pode ser definida como "a habilidade de atender às necessidades presentes sem comprometer a habilidade de futuras gerações de satisfazer suas próprias necessidades" [1]. Esta habilidade inclui a satisfação de quatro dimensões: ambiental, social, econômica e humana [1][2]. Sustentabilidade, portanto, tem se tornado uma força propulsora da sociedade. Preocupações sobre o impacto das atividades humanas vêm aumentando e vários esforços foram iniciados globalmente, principalmente para reduzir o consumo e aumentar a eficiência energética.

Com o passar dos anos, sistemas de TI se tornaram onipresentes em nossa sociedade, melhorando nossas vidas e trabalho, oferecendo conveniência e outros benefícios. Em consonância com este cenário, tais sistemas têm sido identificados como um dos ativos na busca por sustentabilidade [3].

TI normalmente se preocupa com sustentabilidade a partir de duas perspectivas:

- Estes sistemas são grandes consumidores de recursos, não somente quando eles estão operando, mas também considerando todo o ciclo de vida [3]. É

estimado que a energia consumida por sistemas de TI (contando *datacenters*, redes e outros dispositivos baseados em computadores) alcançará cerca de 15% do consumo total até 2020 [4].

- Sistemas de TI estão no coração de todas as áreas críticas à sustentabilidade. Portanto, eles têm um papel importante em chamar a atenção e controlar o uso eficiente de recursos em uma variedade de áreas, tais como *Smart Cities*, *domotics* e exploração de recursos naturais como o petróleo.

A situação atual de TI em relação à sustentabilidade pode ser definida como "criando consciência", mas ainda está longe de ser satisfatória. Grandes firmas, como Google, Microsoft, Apple, entre outras, têm sido criticadas por não priorizar projetos de *datacenters* com uso eficiente de energia sobre a aquisição de fontes baratas deste recurso [5]. O relacionamento entre software e sustentabilidade é ainda uma preocupação menor, muitas vezes nem sendo considerada por usuários e desenvolvedores. Engenharia de software sustentável não é um termo usado amplamente em discussões de tecnologia e responsabilidade ambiental [6].

No entanto, o tratamento de sustentabilidade a partir das perspectivas citadas é uma forma limitada de lidar com a questão. Sistemas de software são parte de um contexto mais amplo, sistemas sócio-técnicos, incluindo organizações, humanos e outros fatores. Os processos de negócio definidos por estas organizações, assim como sistemas de TI, determinam a forma como recursos são consumidos. Existem casos que ilustram o impacto que a reformulação de sistemas sócio-técnicos podem ter em sustentabilidade. Por exemplo, FedEx mudou em 2008 o seu negócio de entrega de documentos a longa distância, oferecendo aos clientes a opção de enviar eletronicamente o documento a ser entregue. Portanto, FedEX foi capaz de enviar a cópia do documento original para o escritório mais próximo do destino, imprimindo e enviando o documento localmente. Este caso ilustra como lidar com sustentabilidade pode requerer a participação de diferentes partes interessadas (*stakeholders*) para adaptar os processos da organização.

O tratamento de sustentabilidade ao nível da engenharia de sistemas engloba não somente as mudanças tecnológicas, mas também o coração de uma organização, seus processos de negócio, e mudanças neste nível são mais difíceis de implementar. No entanto, uma pesquisa recente mostrou que a mentalidade dos CEOs está mudando: dos 750 CEOs questionados em 2010, a maioria respondeu que sustentabilidade deveria ser completamente incorporada à estratégias e operações (96%), implementada por diretorias (93%) e integradas às cadeias de suprimento (88%), um aumento de 25-35% de uma pesquisa parecida em 2007 [7].

Dado o impacto que o projeto de sistemas sócio-técnicos podem ter na sustentabilidade da sociedade, torna-se necessário incorporá-la lista dos principais requisitos não-funcionais (RNF), atribuindo-lhe um tratamento equivalente ao dispensado à outros RNF importantes, como segurança ou desempenho. Como observado por Amsel [6], é preciso encontrar formas de definir e alcançar este RNF. Esta visão é compartilhada por outros pesquisadores, ainda que de forma restrita a software, que sugerem a adaptação de técnicas existentes da engenharia de software, a criação de toolkits, a construção de uma ontologia, e a incorporação deste tipo de meta durante a engenharia de requisitos [8][9].

2 Objetivos da Pesquisa

Apesar ser uma das maiores preocupações do mundo atual, sustentabilidade ainda não é satisfatoriamente compreendida ou tratada durante projetos de desenvolvimento de sistemas [9][10]. Em particular, existem alguns desafios para incorporar aspectos de sustentabilidade à engenharia de sistemas, dentre eles estão:

i. O estado-da-prática sobre o tratamento de sustentabilidade durante a engenharia de sistemas ainda não é conhecido;
ii. Faltam definições, políticas, métricas e padrões para sustentabilidade e formas de calcular o impacto de alternativas de satisfação destas metas no sistema resultante [9];
iii. Existem muitas alternativas a considerar em sistemas complexos, com respeito a sustentabilidade [9];
iv. Uma revisão de literatura sistemática recente [10] mostrou que existem poucas pesquisas sobre o tratamento de sustentabilidade durante a engenharia de software, que está intimamente relacionada à engenharia de sistemas [11]. Soluções existentes foram desenvolvidas para domínios específicos, não podendo ser facilmente utilizadas/adaptadas a outros domínios.

Nossa pesquisa pretende avançar o estado-da-arte e o estado-da-pratica, respondendo às seguintes questões:

RQ1: Qual é o estado-da-prática em relação ao tratamento de sustentabilidade durante a engenharia de sistemas?
RQ2: Que aspectos de sustentabilidade são relevantes para o projeto de sistemas sócio-técnicos?
RQ3: Como integrar aspectos de sustentabilidade ao processo de engenharia de sistemas?

Para responder às questões acima são vislumbrados os seguintes objetivos:

i. Maior compreensão/consciência por parte da indústria e comunidade acadêmica dos conceitos de sustentabilidade na engenharia de sistemas, assim como dos benefícios e impedimentos para incorporá-los ao projeto de sistemas sócio-técnicos;
ii. Integração de aspectos de sustentabilidade à engenharia de sistemas através do desenvolvimento de métodos, técnicas e ferramentas que deem suporte aos diferentes aspectos de sustentabilidade em sistemas sócio-técnicos (ambiental, social, econômica e humana).

3 Contribuições Científicas

Esta pesquisa ainda se encontra em suas fases iniciais. Vislumbramos a engenharia de requisitos orientada a metas e ontologia como abordagens promissoras para a incorporação de sustentabilidade na engenharia de sistemas.

Abordagens de engenharia de requisitos que usam metas/agentes [12, 13, 14] facilitam o entendimento e descrição de problemas associados com estruturas de negócios, processos e seus sistemas de suporte. Em particular, o relacionamento entre agentes é uma parte importante da visão de mundo necessária para modelar sistemas sócio-técnicos. Esta abordagem pode ser especialmente útil para representar metas

referentes às quatro dimensões de sustentabilidade [1][2] e ponderar sobre o impacto destas metas na satisfação das outras metas do sistema.

Ontologias vêm sendo reconhecidas como ferramentas conceituais de grande importância em ciência da computação desde o final da década de sessenta, principalmente em áreas como modelagem de dados (modelagem conceitual), inteligência artificial [15][16] e engenharia de software. Neste trabalho, pretende-se aplicar a fundamentação teórica desenvolvida na engenharia de uma ontologia no domínio da *Sustentabilidade.* Em uma pesquisa recente, Mahaux, Heymans and Saval [8] estendeu uma taxonomia para sustentabilidade, inicialmente proposta por Cabot et al. [9]. Ambos trabalhos usam o framework i* para representar a taxonomia e para facilitar a exploração, o entendimento e a comparação de medidas de sustentabilidade. Estas taxonomias só levam em consideração os aspectos ambientais de sustentabilidade, e ambos os autores reconhecem a necessidade de uma ontologia genérica no tema. Estes trabalhos compõem nosso ponto de partida. Com o auxílio de UFO[21], pretende-se analisar os aspectos de sustentabilidade cuidadosamente para que os conceitos pertinentes ao domínio sejam conhecidos, relacionados e, com o auxílio da ontologia, possam finalmente ficar semanticamente claros.

Portanto, como contribuição, espera-se:

i. Descrição do estado-da-prática em relação ao tratamento de sustentabilidade das empresas que participaram das entrevistas e questionário online, com foco nos benefícios e impedimentos percebidos e nas reais ações tomadas pelas mesmas;
ii. A identificação de métodos, técnicas e ferramentas do estado-da-arte que podem ajudar a vencer os desafios identificados no item anterior;
iii. Realização e divulgação de estudos de caso que investiguem a utilização da engenharia de requisitos orientada a metas/agentes para modelar metas sustentabilidade durante o desenvolvimento de sistemas e ponderar sobre o impacto das técnicas no estado-da-arte (operacionalizações) para a satisfação destas metas;
iv. Descrição precisa dos principais aspectos de sustentabilidade em sistemas sócio-técnicos, através de uma ontologia.

4 Conclusões

Garantir sustentabilidade é um dos maiores desafios da sociedade atual, levando ao surgimento de vários esforços globais para reduzir o impacto ambiental das atividades humanas. Dentre estes esforços encontram-se iniciativas de TI, preocupadas principalmente com o consumo de recursos durante o desenvolvimento e operação de sistemas de software [3][4]. Sistemas sócio-técnicos, que incluem os processos de negócios de organizações, também determinam o consumo de recursos e podem ser projetados de forma a satisfazer metas de sustentabilidade. É imprescindível incorporar o tratamento de sustentabilidade à engenharia destes sistemas. Para tal, é preciso definir sustentabilidade como um RNF de forma precisa e desenvolver formas de satisfazê-lo [6][8][9].

Este artigo descreve a proposta de uma pesquisa que visa investigar sustentabilidade no contexto de sistemas sócio-técnicos e desenvolver métodos e técnicas que deem suporte aos diferentes aspectos de sustentabilidade durante a engenharia de sistemas. A seguir, serão listadas as atividades já realizadas neste início da pesquisa, assim como os seguintes passos.

4 Trabalhos Futuros e em Andamento

Para alcançar os objetivos descritos na Seção 2, será realizada uma pesquisa de natureza empírica e qualitativa, começando por uma investigação do estado-da-prática e a identificação de lacunas em relação ao estado-da-arte.

Dada a natureza exploratória de RQ1, decidiu-se realizar um estudo baseado em entrevistas semi-estruturadas [17]. Este método é apropriado, pois permite o projeto cuidadoso do guia de entrevista com antecedência e a coleta de dados diretamente de especialistas. Um guia de entrevista consiste em um documento que lista as principais perguntas que devem ser feitas aos entrevistados para responder às questões de pesquisa. Tomando como base este documento, o entrevistador guia as perguntas, mas com flexibilidade para indagar mais quando julgar necessário, como por exemplo, para esclarecer respostas ou terminologia. Esta característica é extremamente importante para a pesquisa proposta, dada a falta de uma terminologia homogênea para sustentabilidade. Um guia foi elaborado para estas entrevistas, investigando questões como: (i) quais dimensões/aspectos de sustentabilidade são considerados durante o projeto de sistemas sócio-técnicos, (ii) quais são os benefícios e as limitações percebidos por projetistas de sistemas em relação a satisfação de metas de sustentabilidade, (iii) como as organizações avaliam a satisfação destas metas de sustentabilidade, e (iv) se as organizações conhecem/utilizam técnicas ou padrões de sustentabilidade específicos, como a avaliação do ciclo de vida (LCA) [18].

Adicionalmente, para validar o entendimento do guia, foi feita uma entrevista piloto com uma organização e será criado um glossário.

Amostragem e Coleta de dados: Selecionaremos organizações com a maior variedade possível de características dentro da nossa rede de colaboradores na indústria. Visamos organizações que desenvolvem sistemas sócio-técnicos para os quais sustentabilidade é parte inerente dos processos de negócio. Organizações potenciais receberão uma carta-convite explicando o estudo e perguntando por seu interesse em participar do mesmo. Espera-se entrevistar entre 10 e 20 organizações. As entrevistas estão planejadas para durar uma hora e serão gravadas para análise subsequente.

Análise dos Dados: As entrevistas serão transcritas e analisadas. Serão utilizadas as técnicas de análise de contexto [19] e análise temática [20] como base para avaliar os dados coletados. Para a análise, o seguinte procedimento será seguido: (i) leitura das transcrições e agrupamento das explicações que se referem a mesma pergunta; (ii) organização das respostas para cada pergunta em planilhas, para melhor avaliar e codificar as evidências relacionadas a cada pergunta, e (iii) todas as respostas serão analisadas e será realizada a codificação dos dados obtidos.

Em paralelo às entrevistas será feita uma revisão da literatura para a subsequente identificação de lacunas entre o estado-da-arte e o estado-da-prática. Estas atividades serão importantes para responder à questão de pesquisa RQ2. Uma vez terminada esta primeira fase, o conhecimento adquirido será utilizado para abordar a questão de pesquisa RQ3, utilizando abordagens de engenharia de requisitos orientada a metas e ontologia.

Agradecimentos

Os autores agradecem a Birgit Penzenstadler, Claudia Alaya e Dolors Costal pelas valiosas discussões sobre este trabalho.

Referências

1. UN World Commission on Environment and Development: Report of the World Commission on Environment and Development: Our Common Future. In: United Nations Conference on Environment and Development. (1987)
2. Goodland R.: Encyclopedia of Global Environmental Change, ch. Sustainability: Human, Social, Economic and Environ- mental. John Wiley and Sons. (2002)
3. Tomlison, B.: Greening through IT: Information Technology for Environmental Sustainability. The MIT Press. (2010)
4. Pernici B. (coord.): What IS can do for Environmental Sustainability. In: Panel at CAiSE'11. London, United Kingdom (2011)
5. Greenpeace: How Green is your Cloud. Available at http://www.greenpeace.org/international/Global/international/publications/climate/2012/iCoal/HowCleanisYourCloud.pdf. (2012)
6. Amsel, N.; Ibrahim, Z.; Malik, A.; Tomlinson, B.: Toward Sustainable Software Engineering. In International Conference on Software Engineering ICSE NIER Track. Honolulu, Hawaii (2011)
7. UN Global Compact; Accenture: A New Era of Sustainability UN Global Compact–Accenture CEO Study. Available at: http://www.accenture.com/us-en/Pages/insight-new-era-sustainability-summary.aspx (2010)
8. Mahaux, M.; Heymans, P.; Saval, G.: Discovering Sustainability Requirements: An Experience Report. In International Working Conference on Requirements Engineering: Foundation for Software Quality. REFSQ'11, 19-33. Essen, Germany (2011)
9. Cabot, J.; Easterbrook, S.M.; Horkoff, J., Lessard, L.; Liaskos, S.; Mazón, J.N.: Integrating sustainability in decision-making processes: A modelling strategy. In International Conference on Software Engineering ICSE Companion. pp. 207–210. IEEE. Vancouver, Canada, (2009)
10. Penzenstadler, B.; Bauer, V.; Calero C.; Franch X.: Sustainability in Software Engineering: A Systematic Literature Review for Building up a Knowledge Base. In Workshop on Enterprise-Aligned Software Engineering EASE'12. North Carolina, US (2012)
11. Pyster, A., D. Olwell, N. Hutchison, S. Enck, J. Anthony, D. Henry, and A. Squires (eds).: *Guide to the Systems Engineering Body of Knowledge* (SEBoK) version 1.0. Hoboken, NJ: The Trustees of the Stevens Institute of Tecnology. Available at: http://www.sebokwiki.org (2012)
12. Chung L.; Nixon B. A.; Yu E.; Mylopoulos J.: Non-Functional Requirements in Software Engineering. Kluwer Academic Publishers. (2000)
13. Lamsweerde A.: Requirements Engineering: From System Goals to UML Models to Software Specifications. Wiley (2009)
14. Yu E.; Giorgini P; Maiden N.; Mylopoulos J.: Social Modeling for Requirements Engineering. The MIT Press (2011)
15. Mealy, G.H.: Another Look at Data. In Fall Joint Computer Conference: 525–534, (AFIPS Conference Proceedings, Volume 31), Washington, DC. US (1967)
16. Hayes P.: The Naive Physics Manifesto, In Ritchie, D. (Ed.) Expert Systems in Microeletronics age. Edinburgh University Press, 242-270. (1978)
17. Robson, C.: Real World Research: A Resource for Social Scientists and Practitioner-researchers. Second Edition. Blackwell Publishers Inc. (2002)
18. ISO 14040: Environmental management - Life cycle assessment - Principles and framework, International Organisation for Standardisation (ISO), Geneva (2006)
19. Krippendorff, A.: Content Analysis. Sage Publications, London. (1980)
20. Cruzes D. S, Dybå T.: Recommended Steps for Thematic Synthesis in Software Engineering. ESEM 2011: 275-284 (2011)
21. Guizzardi, G.: Ontological Foundations for Structural Conceptual Models. PhD thesis, University of Twente, The Netherlands (2005)

Apoio Semântico à Engenharia de Requisitos

Joselaine Valaski, Wilian Stancke, Sheila Reinehr, e Andreia Malucelli

Pontifícia Universidade Católica do Paraná, Curitiba, Brasil
joselaine.valaski@pucpr.br, stancke@ieee.org,
sheila.reinehr@pucpr.br, malu@ppgia.pucpr.br

Resumo. A atividade de elicitação de requisitos requer habilidade para discutir os problemas e por meio deles projetar as soluções computacionais. A deficiência desta atividade impacta profundamente nas etapas posteriores do desenvolvimento do software. Desta maneira entende-se que é necessário utilizar um vocabulário comum e formalizado para aprimorar as comunicações, a interoperabilidade e o reuso de artefatos. As ontologias são um tipo de formalismo que podem ser aplicados para contribuir com a melhoria destas atividades. O grupo GPES da PUCPR está desenvolvendo pesquisas com o objetivo de identificar o potencial das ontologias na área de Engenharia de Requisitos. O objetivo é desenvolver um ambiente semântico baseado em ontologias para apoiar a atividade de elicitação de requisitos com objetivo de melhorar a comunicação entre humanos e máquinas. O uso das ontologias deve ser estendido à atividades posteriores do desenvolvimento de software.

1 Introdução

O Grupo de Pesquisa em Engenharia de Software (GPES) da PUCPR foi formalizado em 2009 e concentra suas pesquisas nas seguintes áreas: Qualidade de Software, Melhoria de Processo de Software, Métricas de Software, Gerência de Projetos e Portfólios de Projetos, Reuso de Software, Aprendizagem Organizacional, Ontologias aplicadas à Engenharia de Software e Agentes de Software aplicadas à Engenharia de Software. Na área de Engenharia de Requisitos, no momento, o maior interesse do grupo está relacionado a atividade de elicitação de requisitos e o uso da Unified Modeling Language (UML) executável.

A elicitação dos requisitos tem como o principal objetivo identificar as reais necessidades dos usuários que podem ser ou não resolvidas por uma solução computacional. A elicitação dos requisitos requer habilidade para discutir os problemas e por meio deles projetar as soluções computacionais. A deficiência desta atividade impacta profundamente nas etapas posteriores do desenvolvimento do software. Sendo assim, a comunicação torna-se imprescindível para o sucesso da atividade de elicitação de requisitos. O uso de um vocabulário comum é um das principais práticas a serem realizadas para se atingir comunicação de qualidade. Também é importante estabelecer uma linguagem formal para garantir não somente o melhor entendimento entre humanos mas também o processamento automático por computadores. Além disso, a padronização de linguagens para a representação dos artefatos facilita a

interoperabilidade entre os mesmos. O uso de formalismos reduz a ambiguidade e a replicação dos artefatos.

Outra questão importante a ser ressaltada, é que ao longo dos anos, muitas linguagens, modelos, métodos, ferramentas, frameworks e paradigmas foram propostos para apoiar as atividade da Engenharia de Requisitos. A falta de entendimento e consenso no uso destes termos, também é um fator que pode comprometer a comunicação e a qualidade das atividades da Engenharia de Requisitos. Desta maneira, considera-se importante uma proposta ontológica para a formalização dos conceitos que envolvem a área.

Considerando os problemas de comunicação e formalização existentes na área de Engenharia de Requisitos, entende-se que as ontologias podem contribuir com a melhoria da comunicação manual e automática nas atividades que envolvem a Engenharia de Requisitos. Ontologia é um tipo de formalismo utilizado para a representação do conhecimento e tem sido utilizada com os seguintes objetivos: prover uma compreensão comum compartilhada de uma estrutura de informação entre pessoas ou agentes de software, possibilitar o reuso de domínios de conhecimento, fazer suposições explícitas de um domínio e separar o domínio de conhecimento do domínio operacional [1]. Uma das propostas classificam as ontologias em *lightweight* e *heavyweight*. As ontologias *lightweight* incluem na sua estrutura apenas conceitos, relações e propriedades e as ontologias *heavyweight* adicionam axiomas às ontologias *lightweight* [2]. Esta distinção destaca a diferença de estrutura de conhecimento que cada uma delas pode representar. Enquanto as ontologias mais simples definem uma estrutura de conhecimento, as ontologias mais complexas adicionam o poder de raciocínio à esta estrutura.

Os artefatos gerados na fase de elicitação podem e devem ser reutilizados em fases posteriores ao desenvolvimento do software, como por exemplo, a especificação de requisitos. O uso da UML facilita o trabalho dos engenheiros de requisitos fornecendo uma linguagem de notação padronizada, entretanto, entre a análise de requisitos e a implementação do software existe uma lacuna, pois um profissional será responsável por transformar estes requisitos em outros diagramas intermediários (como por exemplo, diagramas de classe e sequência) e que posteriormente serão transformados em códigos-fonte pelos desenvolvedores. A UML executável [3] preenche a lacuna existente entre os modelos de *design* e os modelos de implementação, permitindo aos engenheiros descrever o software já na fase de requisitos da maneira como ele será implementado em código-fonte. Entende-se que as ontologias podem ter um papel importante na integração entre a atividade de elicitação de requisitos e as atividades mais avançadas do desenvolvimento do software.

Neste contexto, o GPES-PUCPR tem quatro objetivos principais: (i) identificar como as ontologias estão sendo aplicadas na área de Engenharia de Requisitos; (ii) estabelecer uma taxonomia para a área de Engenharia de Requisitos; (iii) desenvolver um ambiente semântico para o apoio na elicitação de requisitos; e (iv) identificar quais abordagens da UML executável estão sendo desenvolvidas pelo meio científico e quais as abordagens utilizadas na indústria. Nas próximas seções são apresentados em maiores detalhes os objetivos da pesquisa, as contribuições científicas, as conclusões e os trabalhos futuros.

2 Objetivos da pesquisa

Nesta seção são apresentados os principais objetivos das pesquisas que estão sendo desenvolvidas pelo GPES-PUCPR na área de Engenharia de Requisitos.

2.1 Identificar como as ontologias estão sendo aplicadas na área de Engenharia de Requisitos

Está sendo realizada uma revisão sistemática nas bases ScienceDirect, IEEE, ACM Digital e em congressos na área de Engenharia de Requisitos com o objetivo de identificar como as ontologias estão sendo aplicadas na área de Engenharia de Requisitos. O interesse é tanto em ontologias *lightweight* quanto *heavyweight*, aplicadas a qualquer etapa do ciclo de desenvolvimento envolvido na área de Engenharia de Requisitos.

Inicialmente mais de 1.800 artigos científicos foram retornados e após a aplicação de critérios de exclusão aproximadamente 120 artigos estão sendo analisados. Como resultado final pretende-se obter uma visão geral do tipo de formalismo utilizado para representar as ontologias, em atividade da Engenharia de Requisitos elas estão sendo aplicadas e qual é o principal papel das ontologias na área.

Com este resultado será possível apontar de que maneira as ontologias podem contribuir com a melhoria da comunicação na área. Trabalho semelhante foi realizado pelos pesquisadores com o objetivo de identificar a aplicação de ontologias na aprendizagem organizacional [4].

2.2 Estabelecer uma taxonomia para a área de Engenharia de Requisitos

Em pesquisa recente realizada pelos pesquisadores [5], referente ao levantamento de temas mais publicados no WER (Workshop em Engenharia de Requisitos), houve dificuldade em estabelecer uma classificação dos temas discutidos na área de Engenharia de Requisitos. Também foi possível observar a diversidade de termos utilizados para denominar os artefatos, alguns dos mais comuns encontrados são apresentados na Fig. 1.

Com os resultados da revisão sistemática do objetivo anterior, pretende-se apontar e discutir taxonomias para a área de Engenharia de Requisitos. Algumas propostas já foram realizadas neste sentido, no entanto, a revisão sistemática terá o objetivo de discutir e ampliar o que já foi proposto.

Entende-se que é essencial a existência de uma ontologia (*lightweight* ou *heavyweight*) para o domínio da Engenharia de Requisitos, mesmo que esta seja inicialmente uma taxonomia dos principais termos. Uma ontologia do domínio pode facilitar o entendimento da área e melhorar a comunicação entre os próprios pesquisadores.

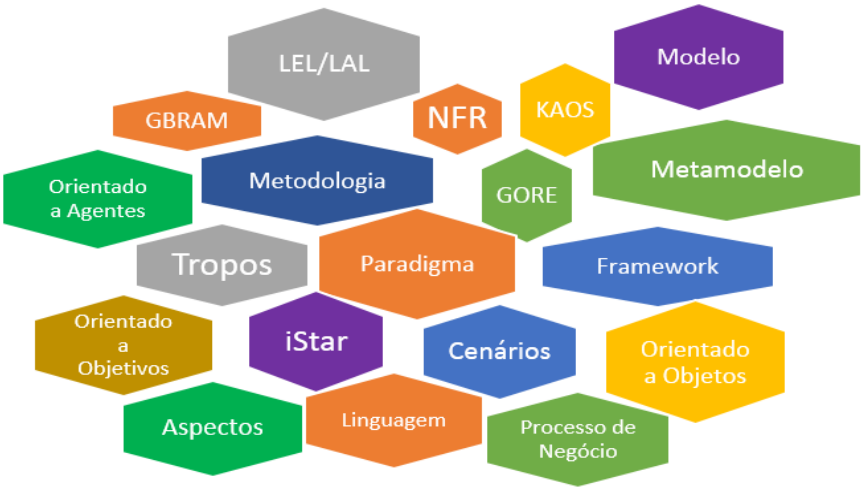


Fig. 1. Alguns termos comuns utilizados na área de Engenharia e Requisitos

2.3 Desenvolver um ambiente semântico para o apoio na elicitação de requisitos

Motivados pela necessidade de melhorar as comunicações na atividade de elicitação de requisitos, pretende-se desenvolver um ambiente baseado em ontologias para aprimorar a atividade de elicitação de requisitos. Com os resultados do primeiro objetivo descrito nesta seção, pretende-se reutilizar ontologias já propostas e/ou propor novas ontologias.

As ontologias serão integradas no ambiente de maneira a permitir maior reutilização de artefatos produzidos nas fases posteriores do desenvolvimento do software, independentemente das técnicas, modelos ou paradigmas utilizados. As ontologias também terão o papel de melhorar a comunicação entre usuários e analistas de sistemas, permitindo a elicitação colaborativa dos requisitos. O apoio computacional na modelagem conceitual também é um dos objetivos do ambiente proposto, pois o especialista de domínio não possui conhecimentos suficiente para utilizar os formalismos recentemente propostos.

Um ambiente para compartilhamento de materiais de aprendizagem relacionados a Engenharia de Software apoiado por ontologias foi desenvolvimento em trabalhos anteriores pelos pesquisadores [6-8].

2.4 Identificar quais abordagens da UML executável estão sendo desenvolvidas pelo meio científico e quais as abordagens utilizadas na indústria

Um estudo está sendo realizado para identificar quais abordagens da UML executável estão sendo desenvolvidas pelo meio científico e quais as abordagens estão sendo utilizadas na indústria. Este estudo visa descobrir quais ferramentas então sendo utilizadas, quais diagramas da UML têm sido utilizados para a execução dos modelos UML, como está sendo feita a transformação dos modelos UML para os modelos de

implementação, como tem sido a recepção da indústria de software na adoção da UML executável ou mesmo do Desenvolvimento Dirigido a Modelos. Para levantar este panorama e conhecer o estado da arte no uso da UML executável, uma revisão sistemática da literatura está sendo desenvolvida como um meio de avaliar e interpretar toda a pesquisa disponível e relevante sobre o tema.

Posteriormente uma proposta usando ontologias para prover a integração entre os artefatos produzidos na elicitação de requisitos e os artefatos produzidos para a execução dos modelos deverá ser avaliada.

3 Contribuições científicas

Com o desenvolvimento dos trabalhos descritos na seção anterior, pretende-se definir um marco ontológico para a área de Engenharia de Requisitos. A comunicação é um dos maiores desafios da área de Engenharia de Requisitos e o primeiro passo deve ser o estabelecimento de um vocabulário comum. Com este resultado inicial pretende-se ampliar as discussões sobre a conceitualização da área e estabelecer as fronteiras entre a área de Engenharia de Requisitos e demais áreas correlatas, como por exemplo, a área de Design de Software.

Uma visão geral do potencial das ontologias no apoio às atividades da Engenharia de Requisitos será obtida após a conclusão da revisão sistemática. Acredita-se que as ontologias sejam um importante recurso a ser aplicado no aprimoramento das comunicações tanto entre humanos como também entre máquinas na Engenharia de Requisitos.

O ambiente semântico visa integrar as ontologias que darão apoio a atividade de elicitação de requisitos. O objetivo é que o ambiente elimine a complexidade do uso de ontologias, garantindo o uso de um vocabulário comum, formalizado e interoperável entre os artefatos produzidos durante esta atividade. O ambiente deve permitir a interação entre usuários e técnicos de maneira que os artefatos sejam produzidos colaborativamente. O ambiente deverá proporcionar a aplicação de ontologias na atividade de elicitação e que poderá ser expandido para apoiar outras atividades tanto da área de Engenharia de Requisitos como de outras atividades da área de Engenharia de Software.

Por fim, com os resultados da revisão sistemática do estudo da UML executável pretende-se obter uma visão geral da aplicação científica e na indústria deste tema e a partir desta visão apontar pontos de avanço.

4 Conclusões

As ontologias têm sido aplicadas em diversas áreas com o objetivo de definir conceitualização formal e consensual dentro de um domínio do conhecimento. Acredita-se que há muitas contribuições a serem feitas na Engenharia de Requisitos aplicando-se as ontologias. Espera-se que com os resultados da revisão sistemática

fique evidente os principais pontos de aplicação e oportunidades de novas contribuições das ontologias para a Engenharia de Requisitos.

Inclusive o uso de ontologias pode ser estendido às fases posteriores a Engenharia de Requisitos, como por exemplo, Design e Implementação, permitindo maior integração entre os artefatos produzidos ao longo do desenvolvimento do software.

5 Trabalhos futuros e andamento

No momento está em andamento a revisão sistemática sobre a aplicação das ontologias na Engenharia de Requisitos que tem previsão de término em julho. Na sequência serão iniciados os trabalhos para o desenvolvimento do ambiente semântico integrado com ontologias obtidas por meio da revisão sistemática. É possível que ontologias sejam expandidas ou novas sejam propostas. Após a finalização do ambiente serão realizados experimentos para verificar a usabilidade do ambiente bem como os benefícios que o ambiente pode proporcionar quando aplicado em uma situação real de elicitação de requisitos.

No que diz respeito a UML executável a revisão sistemática também está sendo realizada no momento. Após a sua finalização serão desenvolvidas propostas em pontos identificados como oportunidades de avanço. Uma das possíveis propostas diz respeito integração entre os artefatos produzidos na elicitação de requisitos e os artefatos produzidos no modelos executável.

6 References

1. Noy, N.F., McGuinness, D.L.: Ontology Development 101: A Guide to Creating your First Ontology, Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880 (2001)
2. Corcho, O., Lopez, M.F., Perez, A.G.: Methodologies, tools and languages for building ontologies. Where is their meeting point? Data & Knowledge Engineering, 46, 41-64 (2003)
3. Mellor, S.J., Balcer, M.J.: Executable UML: A Foundation for Model Driven Architecture. Addison-Wesley (2002)
4. Valaski, J., Malucelli, A., Reinehr, S.: Ontologies Application in Organizational Learning: A Literature Review. Expert Systems with Applications, 39, 7555-7561 (2012)
5. Valaski, J., Stancke, W., Reinehr S., Malucelli, A.: Retrospective and Trends in Requirements Engineering through the WER. In: Workshop em Engenharia de Requisitos (WER), Montevideo (2013)
6. Valaski, J., Reinehr, S., Malucelli, A.: Environment for sharing learning materials in software development teams. In: Informatica (CLEI), 2012 XXXVIII Conferencia Latinoamericana En, Medellin (2012)
7. Valaski, J., Malucelli, A., Reinehr, S., Santos, R.: Ontology to Classifiy Learning Material in Software Engineering Knowledge Domain. In: ONTOBRAS-MOST, vol. 776, pp. 37-47 (2011)
8. Valaski, J., Malucelli, A., Reinehr, S.: Recommending Learning Materials according to Ontology-based Learning Styles. In: The 7th International Conference on Information Technology and Application, Sydney (2011)

Um Modelo para Negociação de Requisitos em Ecossistemas de Software

George Valença, Carina Alves

Centro de Informática
Universidade Federal de Pernambuco (UFPE)
Recife, Brasil
{gavs,cfa}@cin.ufpe.br

Abstract. In the current globalised software industry, the notion of Software Ecosystems emerges based on theories from Ecology to analyse organisations that operate as networks and have a common interest on a central software platform. Studies suggest that the relationship among actors and an effective requirements management are relevant aspects for these ecosystems. Therefore, this research investigates the social dimension of Software Ecosystems to understand how the relationships among its actors should be established to support requirements negotiation. This central goal will be achieved through the development of a Requirements Negotiation Model for Software Ecosystems, defining negotiation strategies along the ecosystem lifecycle. This paper provides an initial discussion about requirements negotiation in Software Ecosystems, presenting an outline of our proposal.

Resumo. Na globalizada indústria de software atual, a noção de Ecossistemas de Software surge com base em teorias de Ecologia para analisar organizações que operam em rede e possuem interesse comum em uma plataforma de software central. Estudos sugerem que o relacionamento entre atores e a efetiva gestão dos requisitos são aspectos importantes para tais ecossistemas. Diante disso, esta pesquisa investiga a dimensão social de Ecossistemas de Software para entender como o relacionamento entre seus atores deve ser estabelecido para apoiar a negociação de requisitos. Este objetivo geral será atendido através da construção de um Modelo de Negociação de Requisitos para Ecossistemas de Software, definindo estratégias de negociação ao longo do ciclo de vida do ecossistema. Este artigo traz uma reflexão inicial sobre a negociação de requisitos em Ecossistemas de Software, apresentando um delineamento da nossa proposta.

Palavras-chave: negociação de requisitos, tomada de decisão, aspectos sociais, ecossistemas de software.

1 Introdução

Nos primórdios da indústria de software, as empresas desenvolviam produtos monolíticos e de difícil integração com outros sistemas. Para aumentar a qualidade e o reuso de sistemas de software surgiram abordagens de desenvolvimento baseado em com-

ponentes, linhas de produtos de software e arquitetura orientada a serviços. Atualmente, a indústria de software é uma das mais competitivas e inovadoras do mercado mundial. Neste contexto, uma tendência recente é o surgimento de Ecossistemas de Software. Este conceito se inspira em teorias da Ecologia para explicar os mecanismos de cooperação e competição entre organizações presentes no ecossistema [2].

Segundo [4], *"um ecossistema de software consiste de um conjunto de soluções de software que suportam e automatizam atividades e transações de atores que estão associados a um ecossistema social ou de negócio"*. A interação entre os participantes do ecossistema cria uma complexa rede de atores desenvolvedores em volta de uma plataforma tecnológica central, em geral fornecida por uma empresa denominada *keystone*. Baseados nela, desenvolvedores externos fornecem produtos para atender as necessidades de uma comunidade de usuários. O sucesso destas aplicações em um ecossistema não depende somente do *keystone*, mas da forma pela qual ela gerencia suas relações com empresas de software, desenvolvedores, parceiros e usuários [6].

A análise de Ecossistemas de Software pode ser feita sob três dimensões: técnica, de negócio e social [7]. A dimensão técnica envolve a plataforma e a infraestrutura tecnológica em que o ecossistema está inserido. A dimensão de negócio envolve conhecimento do mercado, modelos de negócio, portfólio de produtos, e estratégias de licenças e vendas. Por fim, a dimensão social define como atores se relacionam para atingir seus objetivos, com proposições de valor em que todos possam obter ganhos.

O surgimento de um Ecossistema de Software favorece a criação de modelos de negócio envolvendo novos papeis, padrões de colaboração e competição; inovação e proposição de valor. Tais redes organizacionais também impõem novos desafios à indústria de software [8]. O gerenciamento de requisitos é diretamente impactado por esse contexto. À medida que um produto de software aumenta sua complexidade e atinge um amplo mercado consumidor, o fornecedor percebe que, para atender a diversidade de requisitos dos usuários, é necessário envolver empresas parceiras que desenvolvam customizações e extensões para funcionalidades básicas do produto [6].

Este cenário demonstra a migração de um processo de Engenharia de Requisitos centrado em projetos de software pontuais para um panorama mais amplo, com uma ecologia de sistemas e usuários interdependentes [9]. *Stakeholders* diversos e espalhados por todo o ecossistema dificultam a definição e negociação de requisitos. Em particular, a negociação de requisitos deve ser analisada e comunicada entre os diversos atores de um ecossistema de software. A relevância de uma adequada tomada de decisão afeta a qualidade do produto e o tempo gasto para satisfazer as necessidades dos *stakeholders*. Numa análise social, negociações devem considerar objetivos compartilhados e relacionamentos entre participantes em um contexto interorganizacional.

A motivação desta pesquisa é compreender as atividades envolvidas durante o processo de negociação de requisitos em um ecossistema de software. Consideramos esse um desafio significativo para o sucesso do ecossistema e acreditamos no seu impacto direto na prosperidade da plataforma e no atendimento dos objetivos dos envolvidos.

O restante do artigo está organizado da seguinte forma. A Seção 2 detalha os objetivos da pesquisa. Na Seção 3 descreve as contribuições previstas para este estudo. A Seção 4 apresenta considerações finais. A Seção 5 conclui o artigo com a indicação de trabalhos futuros, bem como descrição da situação atual da pesquisa.

2 Objetivos da pesquisa

Em um Ecossistema de Software, organizações atuam numa dinâmica de cooperação e competição para desenvolver novos produtos, atender as demandas do mercado e incorporar novos ciclos de inovação [12]. Tais ecossistemas podem ser classificados como comerciais ou sociais [4]. Em um ecossistema comercial os atores são empresas fornecedoras, integradores externos e clientes que se relacionam via transações financeiras. iPhone, iPad, iMac são exemplos de produtos da Apple que criaram ricos ecossistemas em torno de suas infraestruturas tecnológicas. Já um ecossistema social consiste em usuários, seus relacionamentos e trocas de informação entre eles. Exemplos nesse contexto incluem comunidades de software *open source* como Linux e Android.

Ecossistemas de software também podem ser analisados através do nível de abertura de suas plataformas [5]. Em um ecossistema aberto, os participantes possuem total ou grande influência sobre mudanças e evoluções da plataforma tecnológica. O relacionamento entre os participantes é pautado em confiança mútua e novos parceiros podem facilmente entrar no ecossistema. Já em ecossistemas fechados, o *keystone* possui um papel controlador forte, definindo as evoluções do ecossistema e exigindo certificações formais dos parceiros. A Figura 1 resume as classificações discutidas, descrevendo brevemente as características de cada tipo de Ecossistema de Software.

	Social	**Comercial**
Aberto	Participação ativa de membros da comunidade. Membros podem evoluir a plataforma livremente.	*Keystone* fornece a plataforma tecnológica básica. A comunidade de desenvolvedores externos e usuários pode tomar decisões livremente.
Fechado	Existe um comitê que centraliza as decisões da plataforma. A comunidade pode fazer extensões desde que aprovadas pelo comitê.	O *keystone* centraliza todas as decisões de evolução da plataforma e aprova a participação de novos membros.

Fig. 1. Classificação de Ecossistemas de Software

Atualmente, organizações *keystone* que suportam um ecossistema fechado são criticadas por usarem formato de dados proprietários, gerarem barreiras para mudanças tecnológicas de usuários que desejem sair da plataforma, e utilizarem direitos de propriedade intelectual para prejudicar outras empresas [3]. Em particular, o papel do *keystone* assume importância fundamental para garantir a prosperidade de um ecossistema [2][5]. Suas decisões quanto ao grau de abertura e colaboração entre os membros podem influenciar a estrutura e o crescimento do ecossistema.

Nesse cenário, as empresas passam a compor uma cadeia de valor e são associadas à Gestão da Plataforma de Software. Essa última é responsável pelo planejamento da plataforma como um *framework* que envolve a gestão do portfólio, o planejamento de produtos e *releases*, e o gerenciamento de requisitos [11]. Uma efetiva gestão da plataforma alinha produtos e participantes do ecossistema a macro-objetivos, condições de mercado e interesses dos atores. Obtêm-se então benefícios como produtos que satisfazem as necessidades de nichos do mercado, clientes e participantes internos.

Estas redes de atores são fundamentais para o sucesso dos esforços de desenvolvimento de software do ecossistema. Elas refletem os resultados de atividades de negociação entre atores em termos de necessidades, capacidades e conhecimento. Sob a perspectiva de Engenharia de Requisitos, a fase de negociação envolve diversos *stakeholders* para decidir quais serão os requisitos que farão parte do produto final [1]. Tal processo de tomada de decisão ocorre ao longo da evolução de Ecossistemas de Software, baseada em um ciclo de vida de quatro estágios: nascimento, expansão, liderança e autorrenovação [12]. Ao nascer, o ecossistema têm enfoque na definição inicial de requisitos, quando participantes se associam a ele e apoiam o desenvolvimento de produtos e serviços. Durante a expansão, sua plataforma de software e base de clientes cresce. Na fase de liderança, o ecossistema deve provar ser rentável e ocorrem disputas de poder entre participantes. Por fim, durante a autorrenovação o ecossistema se mantém forte e capaz de se adaptar a novas demandas.

Diante do contexto acima, esta pesquisa busca compreender a atividade de negociação de requisitos considerando a relação entre atores do ecossistema e a influência que possuem na definição dos requisitos ao longo do ciclo de vida. Em particular, pretendemos investigar as seguintes questões:

- **RQ1: Como *stakeholders* do Ecossistema de Software interagem no contexto de negociação de requisitos?**

Buscamos descrever a tomada de decisão realizada durante a negociação de requisitos em Ecossistemas de Software. Sob a dimensão social desse processo, consideraremos três perspectivas: ciclo de vida, modelos de negócio e estratégias de interação.

- **RQ2: Quais perspectivas teóricas podem apoiar a negociação de requisitos em um Ecossistema de Software?**

À semelhança do que vem ocorrendo em Ecossistemas de Software, uma abordagem de construção de teorias será adotada aqui, considerando que podemos evoluir a base conceitual de uma área adaptando teorias de outros campos [10].

- **RQ3: Como alinhar a negociação de requisitos com a Gestão da Plataforma de Software?**

Pretendemos explorar uma perspectiva mais ampla da negociação de requisitos. Investigaremos como a definição e priorização adequadas dos requisitos colaboram para gerar um *roadmap* de sucesso da plataforma de software, fortalecendo o ecossistema.

3 Contribuições científicas

Nosso objetivo é desenvolver um Modelo para Negociação de Requisitos, estabelecendo um conjunto de estratégias de negociação sobre o contexto social do ecossistema e associado à Gestão da Plataforma de Software. Assim, favoreceremos uma adequada definição de requisitos, aprimorando o desempenho dos produtos e apoiando o crescimento do ecossistema. Para investigar como estratégias de negociação de requi-

sitos contribuem para o sucesso e sustentabilidade do ecossistema, consideraremos três elementos da saúde: produtividade, robustez e criação de nicho [2].

O modelo abrangerá o ciclo de vida de Ecossistemas de Software, apoiando a tomada de decisão inerente à negociação de requisitos ao longo da evolução do ecossistema. As estratégias serão construídas segundo diferentes modelos de negócios, apresentando origens e fluxos de influência na negociação de requisitos. O modelo também detalhará os fatores a serem considerados durante a tomada de decisão, com um conjunto de parâmetros para lidar com objetivos e restrições dos atores.

Será possível sugerir, por exemplo, que durante a fase de expansão de ecossistemas sociais abertos, a alta participação de desenvolvedores externos faz com que esses definam os requisitos junto ao *keystone*. Outro exemplo de situação tratada pelo modelo seria: em ecossistemas comerciais fechados, o *keystone* não permite uma ampla negociação dos requisitos que farão parte plataforma, centralizando as decisões.

4 Conclusões

Ecossistemas de Software têm se estabelecido como uma importante área de estudo dentro da Engenharia de Software. Uma vez que esta é uma área relativamente recente, é necessário estabelecer uma fundamentação teórica comum para direcionar pesquisas futuras e agregar de forma coerente conhecimento de estudos experimentais. Esta pesquisa pretende contribuir nesta direção ao tratar de novos desafios associados à Engenharia de Requisitos para Ecossistemas de Software.

A negociação de requisitos nestes ecossistemas deve considerar as necessidades e relações entre as organizações, as quais estão baseadas em uma plataforma tecnológica comum e associadas a uma estratégia de negócios central. Neste contexto, apresentamos um delineamento inicial da proposta de um Modelo de Negociação de Requisitos para Ecossistemas de Software. Tal modelo definirá estratégias de negociação ao longo do ciclo de vida do ecossistema; influenciando positivamente seu crescimento ao favorecer decisões racionais no contexto da Engenharia de Requistos.

5 Trabalhos futuros e em andamento

Essa pesquisa busca fornecer maior familiaridade com um fenômeno emergente e que ainda requer sólida base conceitual. De modo a responder as questões de pesquisa e conceber a solução delineada, conduzimos uma revisão da literatura sobre negociação de requisitos em Ecossistemas de Software e áreas relacionadas para avaliar o estado da arte e identificar necessidades nesse contexto. Com base nisso, trabalhamos atualmente na estruturação do modelo. Nossos esforços também se concentram na análise dos estudos primários das revisões sistemáticas em [7] e [5], reunindo descrições do funcionamento de Ecossistemas de Software. Tal resultado trará evidências sobre a atividade de negociação de requisitos e a dinâmica dos *stakeholders*.

Como trabalhos futuros, investigaremos teorias de negociação em áreas como a Inteligência Artificial, Ecossistemas de Negócio e Ciências Sociais de modo a estabelecer estratégias de negociação de requisitos com base na dimensão social de um Ecos-

sistema de Software. Numa fase seguinte, construiremos o Modelo de Negociação de Requisitos através de uma abordagem de construção de teorias, quando o conhecimento de um tema ou fenômeno específico pode ser formado a partir de outras disciplinas [10]. Apoiando uma abordagem de pesquisa qualitativa, desenvolveremos o protocolo do estudo de caso que será aplicado em organizações de software em contextos privados e *open-source*. Isso nos permitirá identificar e comparar padrões durante a negociação de requisitos em diferentes Ecossistemas de Software. Por fim, avaliaremos o Modelo de Negociação de Requisitos considerando tais resultados.

6 Referências

1. Fricker, S.: Requirements Value Chains: Stakeholder Management and Requirements Engineering in Software Ecosystems. In: 16th International Working Conference Requirements Engineering: Foundation for Software Quality, pp. 60-66 (2010).
2. Iansiti, M., Levien, R.: Strategy as ecology. Harvard Bus. Review, vol. 82, 68-78. (2004).
3. Jansen, S., Brinkkemper, S., Souer, J., Luinenburg, L.: Shades of gray: Opening up a software producing organization with the open software enterprise model. Journal of Systems and Software, vol. 85, 1495-1510 (2012).
4. Bosch, J.: From Software Product Lines to Software Ecosystems. In: 3th Software Product Line Conference, pp. 11-119 (2009).
5. Manikas, K., Hansen, K.M.: Software Ecosystems – A Systematic Literature Review. Journal of Systems & Software, In Press.
6. Berk, I.V.D., Jansen, S., Luinenburg, L.: Software Ecosystems: A Software Ecosystem Strategy Assessment Model. In: 2nd Int'l Workshop on Soft. Ecosyst., pp. 127-134 (2010).
7. Barbosa, O., Santos, R., Alves C., Werner, C., Jansen, S.: A Systematic Mapping Study on Software Ecosystems through a Three-dimensional Perspective. Software Ecosystems: Analyzing and Managing Business Networks in Software Industry. Edward E. Pub. (2013).
8. Santos, R., Werner, C., Barbosa, O., Alves, C.: Software Ecosystems: Trends and Impacts on Software Engineering. In: 26th Brazilian Sym. on Software Eng., pp. 206-210 (2012).
9. Jarke, M., Lyytinen, K.: High Impact Requirements Engineering. Business & Information Systems Engineering, vol. 2, 123-124 (2010).
10. Valença, G., Alves, C., Alves, V.: Analysing Variability Management in BPM and SPL: A Knowledge Mapping. In: IX Simpósio Brasileiro de Sistemas de Informação (2013).
11. Peeters, S.A.: How Software Product Management becomes Software Platform Management in Software Ecosystems. Master's thesis, Faculty of Science Theses (2012).
12. Moore, J.F.: Predators and prey: a new ecology of competition. Harvard Bus. Review, vol. 71, 75-86 (1993).

Expanding Empirical Studies to Better Understand Requirements-driven Collaboration

Sabrina Marczak[1], Irum Inayat[2], Siti Salwah Salim[2]

[1]Computer Science School, Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Porto Alegre, Brazil
sabrina.marczak@pucrs.br
[2]Faculty of Computer Science and Information Technology, University of Malaya
Kuala Lumpur, Malaysia
irum@siswa.um.edu.my; salwa@um.edu.my

Abstract. Requirements engineering involves collaboration among project members. Ineffective collaboration may result in project failure. To study the collaboration of those who need to coordinate work due to interdependencies in requirements, in our previous work we have introduced the concept of requirements-driven collaboration as the collaboration that occurs during requirements engineering and have defined a framework to guide its study. The framework is based on social network theory and provides techniques to study diverse aspects that underlie collaboration driven by requirements. Two case studies were conducted to apply the framework and to reveal empirical insights about requirements-driven collaboration. The investigated projects were globally distributed, of medium-size, and used the waterfall model to guide the development lifecycle. In this paper we posit that additional case studies of projects with distinct characteristics can help us to better understand requirements-driven collaboration. We introduce our intent to study requirements-driven collaboration in agile projects as an example of our plan to further knowledge on the topic. Broader insights can be used by researchers and practitioners to reason about how tools and processes can be improved to better support collaboration throughout the development life-cycle.

Keywords: Requirements-driven Collaboration, Coordination, Social Network Analysis, Empirical Studies, Agile Software Development

1 Introduction

Requirements engineering plays an important role in a project life-cycle, since requirements drive the development of the subsequent project phases. The later in the development life-cycle that a software error is detected, the more expensive it is to repair it. Therefore, it is cost effective to define and specify requirements early on in the project, and to manage them throughout the development cycle.

To develop the project requirements, cross-functional software teams composed of members representing different functional groups need to establish a shared under-

standing or a common ground about the requirements. Effective communication and fleeting knowledge (also known as awareness [1]) are important to foster this common ground [2]. Moreover, communication and awareness are also important for the coordination of work necessary to develop the project requirements since requirements engineering involves highly collaborative activities. For example, requirements analysts often collaborate intensively with customers and end users to gather and to specify requirements.

Coordination in general is the act of managing interdependencies between activities [3]. In order to coordinate properly, team members need to maintain up-to-date knowledge about the requirements, to exchange information about tasks related to a certain requirement, and to propagate information about changes on requirements. In addition, they need to be able to locate expertise when help is necessary to complete their tasks, and to identify who is currently available to help.

In order to study the coordination of those who need to coordinate work due to interdependencies in requirements, in our previous work [4] we have introduced the concept of requirements-driven collaboration (RDC) as the collaboration that occurs during the elicitation, definition, specification, implementation, testing, and management of requirements. In addition, we have also defined a framework to study RDC. Two case studies were conducted to apply the framework and to reveal empirical insights about RDC. Our findings are reported on [5], [6], [7], and [8]. The investigated projects were globally distributed, of medium-size, and used the waterfall model to guide the development life-cycle.

In this paper we posit that *additional case studies of projects with distinct characteristics can help us to better understand RDC*. We believe that specific project characteristics such as number of requirements and development methodology followed might influence team members' behavior towards collaboration with others and therefore should be observed for a better comprehension of RDC.

We present in this paper the RDC framework proposed in our previous work, detail the characteristics we are interested in studying, discuss expected contributions of expanding our initial empirical investigation, and our current intent to study agile distributed software projects as an example of our plan to further obtain insights about RDC to broad knowledge on the topic.

2 Objectives of the research

This paper builds up on our previous research that defined a framework to study RDC and empirically investigated two case studies of a large IT multinational to reveal insights about RDC based on the defined framework. The investigated projects were globally distributed among the headquarter office in the US and the offshore development unit in Brazil, of medium-size (18 and 40 members, respectively), attended internal customers from two distinct business areas of the company (shipping products to the customer and HR applications), of different maturity (new team and a team working together for about 5 years), and mostly used the waterfall model to guide the development life-cycle. Additionally, each project followed a pre-defined organizational structured (one fully centralized in the leaders and another more decentralized due to the members maturity working together) and followed different requirements engineering processes. The number of defined requirements and their level of specification were also distinct in each project: one project consisted of 18 high

level requirements and the other of 120 low level requirements distributed among several legacy applications. Our preliminary findings (e.g., [5][8]) suggested that such characteristics influence to a certain extent collaboration patterns of team members.

Inspired on our initial findings, in this position paper we posit that *additional case studies of projects with distinct characteristics can help us broad the empirical insights and, as a consequence, to better understand RDC in software projects*. Therefore, our long-term goal is *to conduct empirical studies of projects with distinct characteristics*. Larger teams, larger number of requirements dependencies, different types of development methodologies, higher physical distribution, and background are among the characteristics that we aim to investigate when conducting additional studies of RDC. We believe these characteristics might influence team members' behavior towards collaboration with others and therefore should be observed for a better comprehension of RDC.

We briefly present next the defined framework to study RDC in order to provide contextual knowledge for the understanding of our current research proposal.

2.1 The Defined Framework to Study Requirements-driven Collaboration

The defined framework [4] uses concepts and measures from social network analysis [9] to obtain insights about coordination patterns of those involved in requirementsdriven collaboration. The framework is based on a social structure that focuses on the requirement as the unit of work around which collaboration occurs. We termed this structure a requirements-centric team. A requirements-centric team (RCT) is a crossfunctional group whose members' work activities are related to one or more interrelated requirements, as well as downstream artifacts such as design, code and tests. By 'related to' we considered relationships such as 'assigned to', 'communicating about', 'aware of', among others [4].

We also defined a requirements-centric social network to analyze the collaboration within requirements-centric teams. A requirements-centric social network (RCSN) is a social network that represents the members, also called actors, and relationships, also called ties, in a RCT [4]. The actors in a RCSN are among the members of the RCT, and the ties in the network are representations of different relationships during these members' collaboration. For example, a tie can represent project members' requirements-related communication, assignment to work on the same requirements, or awareness of another's requirements-related work.

Based on the requirements-centric social networks, the framework presents a set of measures from social network analysis as mechanisms to explore collaboration driven by requirements. These measures were selected based on literature review and on the two empirical studies conducted. Each measure can answer one or a set of questions regarding RDC. For instance, one can use the RCSN size measure to identify how many team members are collaborating to get a requirement (or a set of dependent requirements) implemented, or use the RCSN cutpoint measure to identify which members would disrupt communication flow if removed from the team. Yet, one can apply the RCSN reachability measure to find out to what extent information can be shared with everyone involved in the development of a certain requirement or run the ties reciprocity test to learn whether awareness is reciprocal between pair of members.

3 Scientific contributions

We previously provided researchers with a framework that can be used as a mechanism to learn fine-grained details about requirements-driven collaboration. Our approach allows the study of collaboration that spans the entire project life-cycle and the examination of collaboration beyond the relationships established among developers only as traditionally reported in literature (e.g., [10]).

To broaden our understanding of RDC we posit that new projects with distinct characteristics should be investigated. Based on our preliminary findings, we aim to investigate projects that attend to one or multiple of the characteristics listed below. Additional characteristics can be suggested and investigated at any given time.

(1) Larger teams: to check if the project team size affects RDC behavior;
(2) Larger number of requirements dependencies: to identify to what extent a larger number of requirements in a same set of dependency and a larger number of dependency sets result in different RDC patterns;
(3) Different project types (e.g., innovative, new product development): to identify whether the nature of the work to be done has any influence on RDC patterns;
(4) Different business background: to identify to which extent domain knowledge affects RDC patterns;
(5) Different types of development methodologies (e.g., agile): to identify whether working practices and processes have any influence on RDC patterns; and
(6) Higher physical distribution (e.g., more sites involved, no overlapping working hours): to further knowledge of the effect of distance on RDC behavior.

The fine-grained requirements-driven collaboration patterns identified by using the framework in each additional project of distinct characteristics can bring more relevant insights to researchers and tool designers of how to propose better tools and processes to support cross-functional teams' collaboration. For instance, we can inform distinct features to the design of tools to support agile and traditional teams according to the identified needs. We can also provide insights on how to and with which periodicity share information about RDC of teams who have never met face to face or do not have overlapping working hours and need to use asynchronous communication tools to coordinate work. In addition, we can identify how requirements experts are located in small and in large teams.

Our preliminary work ([5][6][7][8]) introduces the research design adopted in our previous case studies. It mainly consists of three phases, namely design, data collection, and analysis. Data collection methods were as follows: on-site observation to observe daily working practices; interviews to gather information about organization and team structures and to ask about requirements engineering processes; document inspection to identify the RCTs and to build the assignment RCSNs aiming to set a baseline for the expected collaboration patterns; and questionnaire and work diaries to collect data on the actual collaboration interactions. This research design can be followed in the additional case studies. Data collected should be analyzed considering the project's background and development context. Once data has been analyzed for each single case study individually, results should be compared among cases aiming to identify similar or specific RDC patterns. Similarities or differences among cases suggest the influence (or lack of) of certain characteristics in RDC patterns.

4 Conclusions

In this paper we argue that expanding empirical studies can be beneficial for a better understanding of requirements-driven collaboration. We presented the framework we have proposed to study RDC and discuss the additional characteristics we would like to investigate to broaden knowledge on the topic. We briefly presented next our ongoing investigation of two agile teams of two large companies hosted Asian offices.

5 Ongoing and future work

Our first step putting in practice expanding empirical studies to better understand RDC focuses in investigating teams that follow a different development methodology than the one we have previously studied. The previous teams adopted the waterfall model. By convenience, we are now investigating two agile teams using Scrum.

Agile methods are based on iterative and incremental development where product scope, software requirements, and the overall technical solution evolve throughout the project through highly collaborative behavior of self-organizing and cross-functional teams. The dynamic nature of such methods promotes and encourages rapid and flexible responses to changes at any given stage of the project life-cycle. We are mainly interested to investigate which requirements-driven collaboration patterns are identified in such teams and how they differ from our previous findings.

The investigation of agile RDC invites us to think about how to conceptualize the definitions involved in our framework starting from the unit of analysis−requirements, defined in the Scrum methodology as a feature and transformed into a user story which is used as reference to derive project tasks, and passing through the concepts of requirements-centric teams (who are the members assigned to work on a certain requirement if there is no previous planning and team members should self-assign themselves to implement tasks?) and requirements-centric social networks (e.g., how to define and measure communication when it is expected that members communicate face to face and constantly?). Thus the Scrum-based agile perspective invites us to reconsider the previous definitions of our proposed framework.

We have so far conducted a literature review to better understand the aspects that underlie RDC in Scrum-based agile teams. We found that communication and awareness are the two main aspects involved in such collaboration. The knowledge acquired has been used to design the empirical investigation study that will collect data about RDC in practice in two Scrum-based agile teams located in Asia. We have a researcher on site observing the team members in their work environment and have just recently deployed a questionnaire that will collect about communication and awareness in RDC. Applying the same measures proposed in the framework and reporting early findings (including those from the literature review) are our next steps in order to contribute to a broader understanding of requirements-driven collaboration.

Acknowledgments

We would like to thank our dear colleagues Prof. Dr. Daniela Damian and Dr. Irwin Kwan from University of Victoria, Canada, for their partnership in defining the concept

of requirements-driven collaboration, developing the framework to investigate it, and discussing the meaning of the empirical insights of the two previous case studies.

References

1. Espinosa, A., Slaughter, S., Kraut, R., Herbsleb, J.: Team Knowledge and Coordination in Geographically Distributed Software Development, Journal of Management Information Systems, vol. 24, issue 1, pp. 135-169, 2007.
2. Coughlan, J., Macredie, R.: Effective Communication in Requirements Elicitation: A Comparison of Methodologies, Requirements Eng., vol. 7, issue 2, pp. 47-60, 2002.
3. Malone, T., Crowston, K: The Interdisciplinary Study of Coordination, ACM Computing Surveys, vol. 26, issue 1, pp. 87–119, 1994.
4. Damian, D., Kwan, I., Marczak, S.: Requirements-Driven Collaboration: Leveraging the Invisible Relationships between Requirements and People, Collaborative Software Engineering, Mistrik, I., Grundy, J., van der Hoek, A, Whitehead, J. (Eds.), Chapter 3, pp. 5776, Springer-Verlag, Computer Science Editorial Series, London, England, March 2010.
5. Damian, D., Marczak, S., Kwan, I.: Collaboration Patterns and the Impact of Distance on Awareness in Requirements-Centred Social Networks", in Proc. of the Int'l Requirements Engineering Conference, New Delhi, India, pp. 59-68, IEEE Computer Society, 2007.
6. Marczak, S., Damian, D., Stege, U., Schröter, A., Information Brokers in RequirementsDependency Social Networks, in Proc. of the Int'l Requirements Engineering Conference, Barcelona, Spain, pp. 53-62, IEEE Computer Society, 2008.
7. Marczak, S., Kwan, I., Damian, D., Investigating Collaboration Driven by Requirements in Cross-Functional Teams", in Proc. of the Collaboration and Intercultural Issues on Requirements Communication, Understanding, and Softskills Workshop, in conjunction with the Int'l Requirements Eng. Conf., Atlanta, United States, (available online), IEEE, 2009.
8. Marczak, S., Damian, D.: How Interaction between Roles Shapes the Communication Structure in Requirements-driven Collaboration, in Proc. of the Int'l Requirements Engineering Conference, Trento, Italy, pp. 47-56, IEEE Computer Society, 2011.
9. Wasserman, S., Faust, K.: Social Network Analysis: Methods and Applications, Cambridge University Press, Cambridge, United Kingdom, 1994.
10. Hinds, P., McGrath, C.: Structures that Work: Social Structure, Work Structure and Coordination Ease in Geographically Distributed Teams, in Proc. of the Conference on Computer Supported Cooperative Work, Banff, Canada, pp. 343-352, ACM, 2006.

Elicitação de Requisitos a partir de Modelos de Processos de Negócio e Modelos Organizacionais: Uma pesquisa para definição de técnicas baseadas em heurísticas

Marcos A. B. de Oliveira[1], Sérgio R. C. Vieira[1, 2], Davi Viana dos Santos[1], Sabrina Marczak[3], Tayana Conte[1]

[1]USES – Grupo de Pesquisa em Usabilidade e Engenharia de Software
Instituto de Computação, Universidade Federal do Amazonas (UFAM) Manaus-AM, Brasil
[2]Fundação Centro de Análise, Pesquisa e Inovação Tecnológica (FUCAPI) Manaus-AM, Brasil
[3]Faculdade de Informática, Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS) Porto Alegre, Brazil
[1]{mabo, tayana, davi.viana}@icomp.ufam.edu.br, [2]sergio.vieira@fucapi.br,
[3]sabrina.marczak@pucrs.br

Resumo. A modelagem de processos de negócios e a modelagem organizacional são abordagens para representação de processos e outros elementos que caracterizam contextos empresariais. Ambas as modelagens podem ser fontes importantes de requisitos. Este trabalho descreve o andamento de uma pesquisa sobre a definição de duas técnicas baseadas em heurísticas para elicitação de requisitos a partir de modelos de processos de negócios em *Business Process Modeling Notation* e modelos organizacionais em *Enterprise Knowledge Development*. Com esta proposta, espera-se apoiar a indústria de software no desenvolvimento eficaz e adequado de aplicações que sirvam aos propósitos da organização onde o software irá operar.

Palavras-chave: Técnicas de Elicitação de Requisitos, Modelagem Organizacional, Modelagem de Processos de Negócio, Heurísticas.

1 Introdução

Uma das formas de obter a qualidade do produto de software é realizar o seu desenvolvimento buscando entender claramente o domínio do negócio, considerando os processos de negócio e a compreensão do ambiente organizacional como fontes relevantes para a elicitação de requisitos [1]. O uso de modelos que descrevem os processos de negócios e o contexto de uma organização agrega benefícios para o desenvolvimento de software, tais como: (i) os requisitos passam a refletir as necessidades do negócio; (ii) baixo número de redundâncias de requisitos e (iii) o desenvolvimento do software passa a ser guiado pela necessidade do negócio [2].

Em cada organização, os processos de negócios apresentam características próprias e, por esta razão, é importante dar atenção à modelagem desses processos, explorando as razões e intenções que motivam os diversos componentes do universo organizacional. No entanto, abstrair o contexto de uma organização pode não ser um

trabalho trivial. Portanto, antes de se executar este procedimento, é necessário ter a plena certeza de que a organização se conhece, que entende o seu próprio funcionamento, seus recursos e suas limitações.

A Modelagem Organizacional é um processo onde um modelo empresarial integrado é criado, descrevendo uma empresa específica de várias perspectivas diferentes, como: processos, informações, recursos, pessoal, objetivos e restrições de diversos tipos de organização [3]. Diferentemente da Modelagem de Processos de Negócio que se detém somente aos elementos componentes dos processos que integram a cadeia funcional do negócio.

Tanto os modelos criados pela Modelagem Organizacional quanto os modelos criados pela Modelagem de Processos de Negócios são fontes de informação relevantes. Ambos devem ser considerados na elicitação de requisitos durante o desenvolvimento de novas aplicações ou evoluções de sistemas computacionais.

Este trabalho apresenta uma pesquisa que tem como objetivo a definição de duas técnicas para elicitação de requisitos a partir de modelos organizacionais e modelos de processos de negócios, as técnicas denominadas REMO e REMO-EKD. A técnica REMO (*Requirements Elicitation oriented by business process MOdeling*) [2] propõe heurísticas para apoiar a identificação de requisitos funcionais, não-funcionais e regras de negócio a partir de modelos de processos de negócios descritos em *Business Process Modeling Notation* (BPMN). A técnica REMO-EKD apoia a extração de requisitos a partir de modelos organizacionais descritos usando a metodologia *Enterprise Knowlegde Development* (EKD).

As próximas seções do artigo estão organizadas da seguinte forma: a Seção 2 apresenta os objetivos da pesquisa. A Seção 3 descreve as contribuições científicas a serem alcançadas. A Seção 4 descreve as conclusões. Por fim, a Seção 5 expõe os trabalhos em andamento e futuros.

2 Objetivos da pesquisa

A Modelagem de Processos de Negócio consiste na formalização das atividades de uma organização, capturando o contexto em que estes processos são executados. A notação BPMN é um padrão sugerido pelo *Object Management Group* (OMG) para orientar a modelagem de processos. A BPMN permite a identificação das atividades, os fluxos de tarefas e os controles de dependências [1].

Algumas abordagens de elicitação de requisitos propõem utilizar os modelos de processos de negócios para identificar as funcionalidades que um software deve possuir [2]. Porém, além das funcionalidades, faz-se necessário identificar os demais requisitos (não-funcionais) e regras de negócio que refletem as reais necessidades para automatizar os processos. Por essa razão, foi elaborada a técnica REMO, para apoiar a identificação de requisitos funcionais, não-funcionais e regras de negócio a partir dos modelos de processos. A técnica REMO propõe o uso de heurísticas para análise de diagramas de processos de negócios modelados em BPMN. Requisitos de software são extraídos a partir da análise guiada pelas heurísticas que compõem a técnica. A abordagem empregada utiliza a modelagem de processos de negócios para compreender o contexto no qual o software irá funcionar, antes mesmo de identificar as funcionalidades [2].

No entanto, a técnica REMO se aplica somente para organizações que utilizam BPMN para realizar modelagem de negócios [2]. Desta forma, esta abordagem pode ficar limitada em termos de abrangência de utilização, pois existem outros modelos com informações importantes sobre o cenário da organização. Com o intuito de agregar outras formas de modelagem, optou-se por realizar uma adaptação da REMO para modelagem organizacional em EKD.

A EKD é uma abordagem que fornece uma maneira sistemática e controlada de analisar, entender, modelar e documentar uma empresa e seus componentes, usando Modelagem Organizacional [3]. É composta por seis diferentes modelos (Modelo de Objetivos, Modelo de Regras de Negócio, Modelo de Conceitos, Modelo de Processos de Negócio, Modelo de Atores e Recursos, Modelo de Requisitos e Componentes Técnicos). Esses modelos descrevem a organização em aspectos determinados e complementares. Esta abordagem foi selecionada devido a sua relevância na literatura nos últimos dez anos, além de apresentar modelos com múltiplas visões. Para apoiar a elicitação de requisitos de software a partir dos diferentes modelos organizacionais em EKD, foi proposta uma nova técnica, a REMO-EKD. Da mesma forma que a técnica REMO, a REMO-EKD propõe heurísticas para análise de cada componente dos modelos EKD, apresentando instruções sobre como identificar possíveis requisitos funcionais, requisitos não-funcionais ou regras de negócio relacionados.

3 Contribuições Científicas

A técnica REMO foi avaliada e evoluída através de uma metodologia baseada em experimentação [1]. Os resultados quantitativos dos estudos experimentais apontaram que o uso da técnica REMO contribui para a identificação de requisitos adequados ao contexto dos processos de negócios [2]. A Tabela 1 apresenta parte dos resultados obtidos no segundo estudo experimental da técnica REMO com relação ao indicador de adequação dos requisitos.

Tabela 1. Resultados quantitativos do segundo estudo envolvendo a técnica REMO [2].

Medida	Grupo Tradicional	Grupo Técnica REMO
Requisitos Identificados	339	330
Falsos Positivos	95	56
Média de Adequação dos Requisitos	77,15%	84,39%

Deste modo, a técnica torna-se relevante para o desenvolvimento de software, evitando a aplicação de esforços dos analistas de sistemas em encontrar requisitos inadequados (falso-positivos).

Através da análise de informações qualitativas, a técnica foi evoluída buscando tornar mais fácil sua aplicação. A versão 2 da técnica REMO contempla um conjunto de nove heurísticas, classificadas de acordo com o tipo de requisito que se espera identificar: RF (requisito funcional), RNF (requisito não-funcional) ou RN (regra de negócio). A Fig. 1 mostra parte das Heurísticas da Técnica REMO versão 2.

ELEMENTOS	HEURÍSTICAS	INSTRUÇÕES
 *Tarefa*	**H1 – Atividades / Tarefas do Processo**	**RF → Transforme** em um **requisito funcional (RF)**, caso a atividade/tarefa **possa / deva** tomar-se uma **ação do sistema**; **RNF →** Descreva **como um requisito não-funcional (RNF)**, caso a atividade / tarefa possua restrições para ser realizada.
 *Gateway ou Decisão*	**H2 – Condições de Decisão**	**RF →** Verifique se é necessário **descrever** um ou mais **RF,** a partir da condição identificada. **RN → Identifique** qual/quais **regras de negócios (RN)** podem ser atendidas, **relacionada ou não ao requisito**.
 *Evento de Mensagem*	**H3 – Eventos de Mensagens / Comunicados**	**RF →** Verifique se é necessário **descrever o envio da mensagem como um RF.** **RNF →** Para cada mensagem que deverá ser exibida pelo sistema verifique se é necessário **descrever um RNF** para o seu tempo de resposta.

Fig. 1. Exemplos de Heurísticas da Técnica REMO (v2) [2].

A técnica REMO-EKD está na fase de definição inicial e ainda não foi avaliada experimentalmente. Sua versão inicial compreende dez heurísticas. Cada heurística é composta por três informações: (i) componente ou componentes do conjunto de modelos EKD a que se aplica; (ii) a descrição da heurística; e, (iii) instruções de como extrair requisitos funcionais, requisitos não-funcionais ou regras de negócio da informação expressada pelo componente.

A aplicação das heurísticas ocorre da mesma forma que na técnica REMO: a partir de um modelo em EKD, observa-se determinado componente no diagrama e julga-se, de acordo com a heurística referente ao tipo de componente. Se a descrição do componente expressa um requisito de software, segue-se as instruções da heurística para extração do requisito. A Tabela 2 apresenta a definição de uma das heurísticas propostas. A Fig. 2 apresenta um exemplo de aplicação da heurística H01.

Tabela 2. Heurística H01 da técnica REMO-EKD.

Componentes	**Heurística**	**Instruções**
Objetivo; Oportunidade;	**H01: Objetivos** *Expressa objetivos a respeito do negócio ou desejos individuais ou organizacionais a alcançar [3]. Pode ser automatizado por funções ou contemplado por características que o sistema irá possuir.*	**RF →** Extrair um ou mais requisitos funcionais caso o objetivo/oportunidade apresente uma funcionalidade do sistema; e/ou **RNF →** Extrair como um requisito não funcional caso o objetivo/oportunidade apresente uma restrição desejável pela organização para o sistema;

As outras heurísticas da técnica REMO-EKD se aplicam aos seguintes componentes dos modelos EKD: problemas, restrições, regras, processos, processos externos, dados, atores e recursos. Como todos os modelos EKD são inter-relacionados, optou-se por definir heurísticas por componentes e não por modelos específicos. Desta forma, as heurísticas podem ser aplicadas independentemente dos modelos criados.

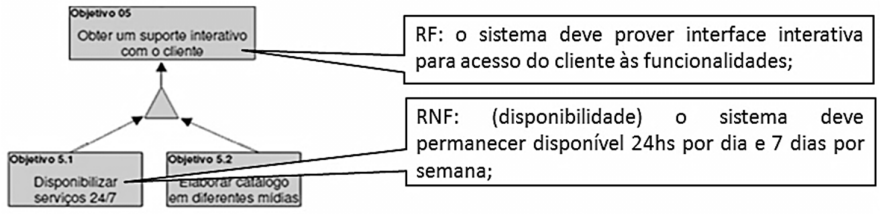


Fig. 2. Exemplo de aplicação da heurística H01

4 Conclusões

O contexto dos processos de negócio ou da organização pode ser importante fonte de requisitos. A modelagem de processos de negócio reflete o contexto onde os processos são executados. A modelagem organizacional é utilizada como técnica para representar e entender a estrutura e comportamento das organizações. Esta descrição é útil no processo de elicitação de requisitos, pois colabora para o entendimento do negócio, um ponto de grande discussão na prática da Engenharia de Software.

As técnicas apresentadas neste trabalho buscam incluir os modelos de processo de negócio e modelos organizacionais no processo de elicitação de requisitos. Essas técnicas utilizam heurísticas para apoiar a extração de requisitos a partir dos modelos abordados. Em outras palavras, as heurísticas guiam o analista na identificação dos requisitos, baseando-se nos propósitos dos componentes dos modelos.

As técnicas REMO e REMO-EKD objetivam contribuir para uma melhor qualidade dos requisitos, permitindo que o analista de sistema possa identificar requisitos mais adequados ao contexto dos processos de negócios e da organização, respectivamente. A técnica REMO-EKD é uma adaptação da técnica REMO, onde se expandiu o alcance das heurísticas para compreender organizações que utilizem o método EKD para modelagem de seus objetivos e atividades.

Espera-se que o conjunto de técnicas apresentado auxilie a indústria de software a fazer uso de diferentes tipos modelagem de processos como uma fonte de informação relevante para o desenvolvimento de software, a fim de obter produtos de software com maior qualidade.

5 Trabalhos futuros e em andamento

Atualmente a técnica REMO-EKD está sendo avaliada experimentalmente para verificar a viabilidade em relação à eficácia e adequação. Estes são os mesmos indicadores utilizadas nos estudos experimentais aos quais a técnica REMO foi submetida [2]. Após a finalização do estudo, algumas sugestões de mudança podem ser verificadas. Essas mudanças visam aperfeiçoar a técnica em relação à elicitação de requisitos em modelos EKD e à especificação precisa dos mesmos.

Sobre a técnica REMO, ela foi evoluída para uma terceira versão. Os resultados do segundo estudo experimental apontaram questões que devem ser consideradas ao realizar a elicitação de requisitos a partir de modelos de processos de negócio [1].

Esta terceira versão será avaliada experimentalmente para verificar sua aplicação no contexto industrial.

Referências

1. Vieira, S., Viana, D., Nascimento, R., Conte, T. Using Empirical Studies to evaluate the REMO Requirement Elicitation Technique. Proc. 24th International Conference on Software Engineering and Knowledge Engineering, Redwood City, CA, USA, 2012, pp. 33-38.
2. Vieira, S. Viana, D., Nascimento, R., Conte, T. Avaliando uma Técnica para Extrair Requisitos a partir de Diagramas de Processos de Negócios através de Estudos Experimentais. Anais do CLEI-IS – Simpósio Latino-Americano sobre Engenharia de Software. Medellín, Colômbia, 2012.
3. Bubenko, J,R., Stirna, J., Brash, D. EKD User Guide, Dept. of Computer and Systems Sciences. Stockholm: Royal Institute of Technology, 2001.

Integrando o Framework I* com a Gerência de Risco

Jean Poul Varela[1], Jaelson Castro[1], Victor F. A. Santander[2]

[1]Centro de Informática, Universidade Federal de Pernambuco, Recife, Brasil.
{jpv, jbc}@cin.ufpe.br
[2]Colegiado de Ciências da Computação - Universidade Estadual do Oeste do Paraná, Cascavel, Brasil
vfasantander@unioeste.br

Abstract. O gerenciamento de riscos é essencial para o sucesso do projeto. Contudo, a gerência de riscos é uma atividade difícil e subjetiva. O objetivo deste trabalho é extrair elementos de modelos organizacionais para auxiliar a realização do gerenciamento de riscos. Nós utilizamos o framework i* para modelar os relacionamentos entre as pessoas envolvidas no projeto. Através da análise dos relacionamentos será possível extrair elementos para guiar a realização do gerenciamento de riscos.

Keywords. Gerência de Riscos, Framework i*, Integração,

1 Introdução

Um dos grandes desafios atuais na computação está em desenvolver sistemas computacionais no tempo e orçamento definidos bem como que atendam aos requisitos dos seus clientes e usuários. O passo inicial para evitar problemas neste sentido é modelar a organização alvo utilizando modelos organizacionais. Entre as abordagens que tem recebido destaque nos últimos anos está o framework de modelagem organizacional i* [1]. Este framework permite descrever aspectos de intencionalidade e motivacionais envolvendo atores em um ambiente organizacional. Para descrever estes aspectos são propostos dois modelos: O Modelo de Dependências Estratégicas (SD) e o Modelo de Razões Estratégicas (SR). O SD é composto por nós e ligações. Os nós representam os atores no ambiente e as ligações são as dependências entre os atores. Por ator entende-se uma entidade que realiza ações para obter objetivos no contexto do ambiente organizacional. O SR é um modelo complementar ao SD Mais informações sobre o framework i* podem ser obtidas em [1].

Por outro lado, para o sucesso de um projeto de desenvolvimento de software, outro aspecto crítico que precisa ser considerado é o gerenciamento de projeto [2]. Entre as abordagens de gerenciamento de projeto percebe-se a ampla utilização da abordagem proposta pelo PMI (Project Management Institute), a qual é documentada no PMBOK (Project Management Body of Knowledge)[3]. O PMBOK é dividido em áreas de conhecimento, sendo que para trabalho foi optado pela utilização da área de conhecimento de gerenciamento de riscos, pois lidar com os riscos inerentes a

um projeto de software é um desafio. Comumente, riscos são negligenciados, não descobertos ou mesmo não são associados a pessoas ou tarefas específicas dentro do projeto, o que dificulta enormemente a mitigação dos mesmos.

O gerenciamento de riscos segundo o PMBOK é realizado executando vários processos, os quais podem ser resumidos em planejamento do gerenciamento de risco, identificação dos riscos, análise qualitativa de riscos, análise quantitativa de riscos, planejamento de respostas a riscos bem como monitoramento e controle de cada risco. Contudo, ainda persistem vários problemas no gerenciamento de riscos tais como o alto grau de dificuldade e subjetividade associado à identificação dos riscos e dificuldade para detectar quem ou quais tarefas estão envolvidas ocorrência de um risco.

Observando este contexto, verificamos que o uso do framework i* pode reduzir o alto grau de subjetividade na realização da gerência de riscos. Entendemos que os riscos de um projeto podem ser mais facilmente identificados, controlados e mitigados se modelarmos as intencionalidades estratégicas dos membros da equipe envolvida no projeto e associarmos a estas intencionalidades, informações sobre os riscos do projeto.

O trabalho está estruturado conforme segue. Na seção 2 é apresentado o objetivo do presente trabalho. Na seção 3, a contribuição deste trabalho é descrita e exemplificada. Na seção 4 são realizadas as considerações finais do trabalho. Na seção 5 são descritos os trabalhos futuros.

2 Objetivos

Este trabalho tem como objetivo a elaboração de uma proposta para auxiliar o gerenciamento de riscos, para este propósito se recomenda a geração de modelos organizacionais através da integração do framework i* com a área de conhecimento de gerenciamento de riscos, para que a partir desses modelos organizacionais auxiliares se tenha a possibilidade de identificar possíveis riscos, bem como identificar quem e quais tarefas estão envolvida na ocorrência do risco entre outras informações relacionadas com os riscos.

3 Proposta

Para apoiar a apresentação da proposta deste trabalho, consideraremos uma organização desenvolvedora do software composta por programadores, gerentes de projeto, um engenheiro de requisitos e um arquiteto, sendo que as responsabilidades de cada integrante da equipe são bem definidas. Considerando esta organização, segue a descrição das diretrizes propostas as quais permitem a partir do framework i* extrair elementos essenciais para realização do processo de gerência de riscos. Cabe ressaltar que por questões de espaço, apenas alguns modelos parciais são apresentados, a versão completa deste estudo é apresentada em [4].

Passo 1: Gerar o modelo SD e SR com foco nas pessoas envolvidas no projeto

Diretriz 1: A geração do modelo SD tem por objetivo explicitar as relações existentes entre todos os envolvidos no projeto. Inicialmente deve-se encontrar os papéis necessários para o gerenciamento e execução do projeto. Cada papel deve ser mapeado para um ator no modelo SD do framework i*. Em seguida deve-se encontrar e mapear as dependências do tipo objetivo, tarefa, recurso e objetivo-soft entre os atores descobertos. Recomenda-se também enriquecer este modelo adicionando a criticidade às dependências [1] existentes. A figura 1 apresenta o modelo SD com estas informações modeladas para nosso exemplo.

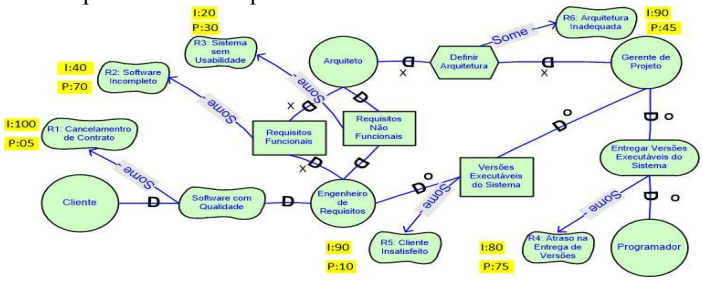


Fig. 1. Modelo SD com foco nas pessoas envolvidas no projeto incluindo a criticidade dos relacionamentos, representação da probabilidade e do impacto dos riscos

Diretriz 2: O modelo SR a ser gerado deve ter como base o modelo SD construído na diretriz 1. No modelo SR são detalhadas as razões internas necessárias para satisfazer cada dependência no modelo SD. A figura 2 apresenta o modelo SR gerado.

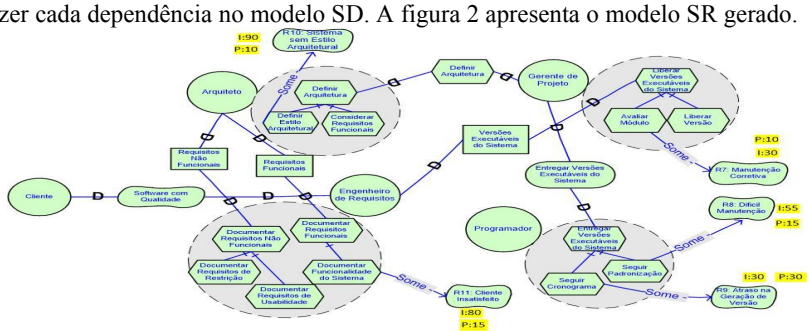


Fig. 2. Modelo SR com foco nas pessoas envolvidas no projeto a probabilidade e o impacto dos riscos.

Passo 2: Descoberta dos riscos nos modelos SD e SR

Tendo em vista que grande parte dos riscos que ocorrem durante o projeto, são oriundos de tarefas que não foram bem executadas por pessoas envolvidas no projeto, então descoberta de riscos é realizada através da análise dos relacionamentos contidos nos modelos SD (figura 1) e SR (figura 2). Os riscos nesta proposta são representados como objetivo-soft. O conceito original para este tipo de objetivo pode ser encontrado

em [1]. A premissa para representação de um risco do projeto como um objetivo-soft vem do fato de que um risco é um fenômeno que pode ocorrer no projeto, mas a sua identificação e sua ocorrência na maioria das vezes não é objetivamente definida, possuindo uma forte carga de subjetividade vinculada ao entendimento das pessoas envolvidas no projeto.

Diretriz 3: para a descoberta de riscos no modelo SD gerado na diretriz 1, as seguintes subdiretrizes devem ser observadas:

SubDiretriz 3.1: analisar as dependências estabelecidas no modelo SD gerado na diretriz 1, verificando para cada dependência a possibilidade da ocorrência de um risco caso a mesma não seja satisfeita.

Subdiretriz 3.2: para cada dependência analisada na subdiretriz 3.1 que gerou um risco, cria-se um objetivo-soft para representar este risco identificado. Para que o risco descoberto seja visualizado no modelo SD de forma mais clara, o objetivo-soft que representa o risco será descrito da seguinte forma:

R + Número do Risco: **Nome do Risco**, onde: **R**: indica que aquele objetivo-soft representa um risco; **Número do Risco**: número de identificação atribuído ao risco; **Nome do Risco**: descreve o nome do risco em questão.

Subdiretriz 3.3: cria-se um relacionamento do tipo (**some -**) [1] entre o *Dependum* de um relacionamento que gerou um risco e o seu respectivo objetivo-soft que representa o risco gerado. O relacionamento do tipo (**some -**) indica que caso a dependência seja satisfeita, a mesma contribuirá para que o risco indicado no objetivo-soft não ocorra (influência negativa). Os riscos estão representados na figura 1 que apresenta o modelo SD com foco nas pessoas envolvidas no projeto. Por exemplo, se o recurso *Versões Executáveis do Sistema* não for disponibilizado para o Engenheiro de Requisitos pelo Gerente de Projeto, o risco *R5:Cliente Insatisfeito* terá grandes chances de se manifestar.

Diretriz 4: para a descoberta de riscos no modelo SR com foco nas pessoas envolvidas no projeto, as seguintes subdiretrizes devem ser observadas:

Subdiretriz 4.1: analisar os subelementos do modelo SR gerado na diretriz 2. verificando para cada subelemento a possibilidade da ocorrências de um risco caso o mesmo não seja satisfeito.

Subdiretriz 4.2: para cada risco identificado na subdiretriz 4.1, cria-se um objetivo-soft para representar este risco. A representação deste risco é realizada de forma análoga a subdiretriz 3.2.

Subdiretriz 4.3: cria-se um relacionamento do tipo (**some -**) entre o subelemento que gerou um risco e o seu respectivo objetivo-soft que representa o risco gerado. Os riscos são representados na figura 2 que apresenta o modelo SD com foco nas pessoas envolvidas no projeto. Por exemplo, uma das tarefas internas que o ator programador deve realizar para *Satisfazer a Entrega de Versões Executáveis do Sistema* é *Seguir a Padronização* para o código adotada pela equipe. Caso esta tarefa não seja realizada, o risco iminente será a difícil manutenção deste código representado pelo risco *R8:Difícil Manutenção.*

Passo 3: Adição de elementos da gerência de riscos

Para enriquecer esta proposta é possível realizar o incremento dos modelos organizacionais gerados até o momento. As diretrizes a seguir auxiliam no processo identificação de probabilidade de riscos e avaliação do impacto dos mesmos.

Diretriz 5: realizar a inserção da probabilidade nos riscos descobertos no modelo SD (Diretriz 1). Pode-se realizar a inserção de probabilidade ao lado do objetivo-soft que representa o risco, agregando um comentário com a probabilidade de ocorrência do risco. A atribuição do valor estimado da probabilidade de cada risco é realizada de forma análoga ao processo tradicional de definição de probabilidade de riscos proposta no PMBOK, contudo, neste modelo é possível determinar a probabilidade analisando a dependência que gera o risco de forma pontual, avaliando a experiência, a qualidade dos atores envolvidos na dependência e o grau de complexidade da mesma. Para identificar a probabilidade do risco, a mesma deve adotar a seguinte nomenclatura: **P + Probabilidade**, onde: **P**: indica que aquela nota representa a probabilidade do risco acontecer; **Probabilidade**: é o valor numérico da probabilidade do risco acontecer. Por exemplo, na figura 1, para o risco R6:Arquitetura Inadequada definiu-se a probabilidade de 45% de ocorrência representada por P:45.

Diretriz 6: realizar a inserção da probabilidade nos riscos descobertos no modelo SR (Diretriz 2), a atribuição da probabilidade neste modelo é realizada de forma análoga a diretriz 5 mas também é levada em consideração a complexidade da tarefa interna que gera o risco. Por exemplo, na figura 2, para o risco R8:Difícil Manutenção a probabilidade definida é de 15% representada por P:15.

Diretriz 7: realizar a inserção do valor do impacto nos riscos descobertos no modelo SD (Diretriz 1). Pode-se realizar a inserção do valor do impacto ao lado do objetivo-soft que representa o risco, agregando um comentário com o valor do impacto da ocorrência do risco. Atribuição do valor do impacto de cada risco é realizada de forma análoga ao processo tradicional de definição do valores do impacto de riscos proposta no PMBOK[2]. Contudo, neste modelo é possível determinar a probabilidade analisando a dependência que gera o risco de forma pontual. Para identificar o impacto do risco, adota-se a seguinte nomenclatura: **I + Impacto**, onde: **I**: indica que aquela nota representa o valor do impacto do risco. **Impacto**: é o valor numérico do impacto risco. Por exemplo, na figura 1, para o risco R1:Cancelamento Contrato considera-se que o mesmo tem um efeito catastrófico (I:100) sobre o projeto.

Diretriz 8: realizar a inserção do valor do impacto nos riscos descobertos no modelo SR (Diretriz 2). A única diferença em relação a diretriz 7 é que agora podemos anotar o impacto para riscos oriundos de elementos específicos internos aos atores envolvidos no projeto. Por exemplo, na figura 2, no risco R11: Cliente Insatisfeito, o valor do impacto do mesmo é de 80, representado por (I:80).

Passo 4: Realizar uma descrição textual dos riscos encontrados na diretrizes anteriores. Para que a integração seja documentada de forma textual recomenda-se que toda vez que um risco seja identificado, o mesmo também seja documentado.

Diretriz 9: para documentar os riscos encontrados, sugere-se utilizar o template proposto na tabela 1. Nesta tabela, por exemplo, apresenta-se a documentação do risco R6:Arquitetura Inadequada. A maioria destas informações é extraída dos modelos SD e SR gerados ao longo do processo de integração proposto.

Table 1. Template para a documentação do Risco R6:Arquitetura Inadequada

Identificação do Risco	6
Nome do Risco	Arquitetura Inadequada
Dependum Causador	Tarefa Definir Arquitetura
Depender Causador	Gerente de Projeto
Dependee Causador	Arquiteto
Causas	1 – Não foram considerados os requisitos não-funcionais;
Modo de Prevenir	1 – Garantir que o engenheiro de requisitos documente os requisitos não-funcionais;
Conseqüências	Alguns requisitos não-funcionais críticos poderão não ser satisfeitos.
Probabilidade	45%
Impacto	90

4 Conclusão

Este trabalho teve como intuito propor a integração da modelagem organizacional via framework i* com o processo de gerência de riscos. Desta forma diminuiu-se o grau de subjetividade do processo de gerência de riscos. Através da análise dos relacionamentos de forma pontual entre os envolvidos no projeto, se torna possível verificar quais relacionamentos podem ocasionar a ocorrência de um risco. Em seguida determina-se os riscos do projeto e sua probabilidade e impacto. Definindo estas informações, é possível, em nível de gerência, traçar planos para que os mesmos não ocorram, bem como focar em ações preventivas ou corretivas específicas para determinados membros (e atividades) da equipe do projeto.

5 Trabalhos futuros

Com a elaboração desta proposta se torna necessário a construção de uma ferramenta para realizar a proposta. Também se pode criar um catálogo de riscos comuns entre projetos. Também se pode estudar uma forma de representar os planos de contingência nos modelos organizacionais gerados por essa proposta. Também se pode criar uma representação das ações preventivas dos riscos.

Referências

1. Yu, E., Giorgini, P., Maiden, N., Mylopoulos, J.: Social Modeling for Requirements Engineering. The MIT Press (2011).
2. Kerzner, H.: Gestão de Projetos as Melhores Práticas. 2ª Ed. Porto Alegre: Bookman Companhia Editora (2006) .
3. Project Management Institute (Corporate Author): A Guide to the Project Management Body of Knowledge (PMOBK Guide). Fourth Edition (2008).
4. Varela, J.P.: Integrando o Framework i* ao Processo de Gerência de Riscos (Monografia de Trabalho de Conclusão de Curso de Graduação). Colegiado de Ciência da Computação - Universidade Estadual do Oeste do Paraná, Cascavel - PR, Novembro (2011).

Mastem: A Mathematics Tutoring Multi-Agent System

Jéssyka Vilela[1], Ricardo Ramos[2], Jaelson Castro[1]

[1]Universidade Federal de Pernambuco
Centro de Informática
Av. Jornalista Anibal Fernandes, S/N, Cidade Universitária, 50.740-560, Recife-PE, Brasil
{jffv,jbc}@cin.ufpe.br

[2]Universidade Federal do Vale do São Francisco
Colegiado de Engenharia da Computação
Av. Antonio Carlos Magalhães, 510, Country Club, 48.902-300, Juazeiro-BA, Brasil
ricardo.aramos@univasf.edu.br

Abstract. The results of IDEB 2009 indicate the elementary students are failing mathematics exams. It is well known Intelligent Tutoring Systems could of be assistance in such cases because they are always available and resourceful. However, tutor systems need to be autonomous, flexible and adaptable. We claim that agent orientation has the natural paradigm for the development of such systems. The objective of this work is to propose the design and implementation of an educational software, called Mastem, to aid the teaching of mathematics targeted to students of 5th and 6th grades. This tutor system assists the students, adapts itself to the student's difficulties by identifying errors and presents tips/examples. We have successfully used the Tropos methodology and JADE frameworks to develop this software.

Keywords: Multi-Agent System, Framework i*, Tropos, JADE

1 Introduction

The agent paradigm is a technology that has been used in several areas. One of its applications is to improve significantly the process of developing of learning environments of educational software, especially those like Intelligent Tutoring Systems (ITS) [1]. This occurs in function of autonomy, flexibility, collaboration and adaptation provided by this paradigm [1].

In this context, it was found that the high failure rate of elementary students causes, among other things, reducing the number of students enrolled in these series. Furthermore, the results of IDEB 2009 [2] shows that performance in mathematics increased slightly compared to the performance in Portuguese. The poor performance in mathematics of students in upper grades of elementary school, combined with high repetition rate of these same series promotes the need for a specific intervention.

In this work, we performed the design and implementation of educational software like intelligent tutoring system (ITS), called Mastem (Multi-Agent System to Teaching

and Evaluation of Mathematic), to aid of the teaching of mathematics intended for students of elementary schools of Brazil, especially students of 5th and 6th grade.

The Mastem has the ability to adapt itself according with user difficulties, providing tips and detailed examples. We used Tropos framework [3] to guide the development of the Mastem. This framework adopts the i* organizational modeling framework, which offers the notions of actor, goal and (actor) dependency [3]. We developed the Mastem software in Java language and used the JADE library [4].

2 Objectives of the research

The objective of this work is to evaluate the use of Tropos and JADE to define, model, and implement intelligent tutor system to help the teaching of mathematics.

3 Scientific contributions

3.1 System Description

The Mastem addresses the following topics: multiple and natural number divisors, prime numbers and composite numbers, factoring, and fractions. These topics are the main difficulties found in students of elementary schools of Brazil [2]. The system works with an ordered sequence of 120 questions in three difficulty levels (easy, medium, and hard) and answers divided in four options each one. The user evolution depends on answering correctly the questions within a time interval previously established. If the user does not be successful in answering the questions of a specific topic, the agent Tutor will teach him with tips and examples related on the question. Tutor agent monitors the user performance and displays the ranking of the three top users with the highest scores, the Ranking agent updates the display after each answered question, stimulating the efforts to overcome the user.

When the user answers correctly the questions, it increases his score progressing to the next level, being awarded with medals displayed in a table. The medal table (shown in Figure 2) is another motivational factor for the user, since it challenges and encourages the learning of content worked in the software.

3.2 The use of tropos to develop an educational software

The Tropos framework [3] is a methodology for developing multi-agent systems. Its name is derived from the Greek “tropé”, which means "easily modifiable" or "easily adaptable". Thus, this framework allows the development of systems according to the actual needs of an organization, seeking to integrate the system and the changing environment in the best way.

The Tropos framework defines five phases in the development of multi-agent systems [1]: i) early requirements, ii) late requirements, iii) architectural design, iv) detailed design and v) implementation.

The modeling of early requirements of the Mastem was done by generating the models of Strategic Dependency (SD) and Strategic Rationale (SR) defined by framework i* [1]. In this phase, we identified the actors Student and Teacher and the Tutor Agent. In addition, we need the resources Time, Questions and Punctuation, the softgoals Entertainment and Usability and the following objectives: Mathematics be learned, Student's performance be monitored and Questions be answered).

The phase of late requirements consisted of the addition of the System actor, the Ranking and Difficulty agents and new modeling elements. The Strategic Dependency model in the phase late requirements was built using OpenOME [9] and it is shown in Figure 1.

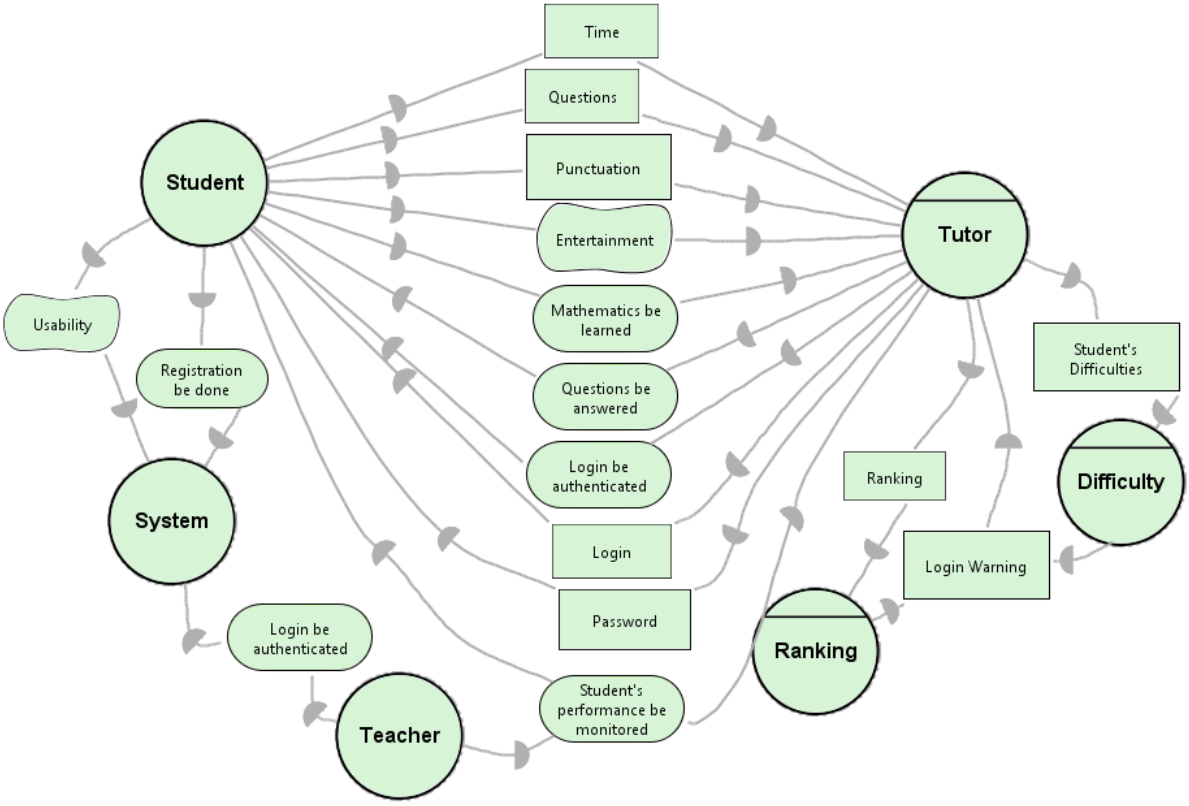


Figure 1. SD model of the Mastem in the late requirements phase.

A brief description of the dependence between actors and agents is given as follows: Teacher Actor - It can insert questions in the software and also check the performance of the students. Student Actor - It aims to learn mathematics, answering questions and

have his performance monitored by the agent Tutor, and this makes use of available resources by agent Difficulty. System Actor - It is responsible to register and to provide usability on the Mastem. Difficulty Agent - It manages the difficulties list that the student has, and deliver this list for the Tutor when the Difficulty agent receives the warning of login. Ranking Agent - It should provide the ranking with the three students with the highest scores for the Tutor agent to show this information to the Student actor. Similarly, the agent performs this task when the Tutor notifies him about the student login.

In architectural design phase, we constructed the Non-Functional Requirements (NFR) diagram [3] through the softgoals of the Mastem. Then, we consulted the table of attributes correlation and styles [6] to determine if they contribute positively or negatively on achieving the goals.

We chose the Strategic Union style because it is the most suitable for the architecture of Mastem because it involves agreements between two or more partners in order to obtain benefits from larger scale. Thus, agents will work collaboratively to achieve their goals.

In the detailed design phase, we performed a micro-level specification of agents [3]. The Mastem specification was designed by AUML language [8] and we created the class diagram, sequence and communication in the Astah Community tool [7]. All diagrams and more details can be found in [10].

The implementation phase consisted of developing the relational model and the Mastem's database using the PostgresSql. Then, the system was coded in Java through the IDE Netbeans and the JADE framework [4] to create the agents behaviors.

The main screen of the student, shown in Figure 2, consists of a button bar that contains other resources that he can access, information about his score and medals and the ranking with three users with the highest scores. In addition, this screen contains the field where the question appears, the option of response and the time in which the student must answer the question.

4 Conclusions

This paper presented the development of educational software Mastem of intelligent tutor system type, adaptable to the user difficulties, from the design phase through implementation. The Mastem was conceived with the objective of contributing to the learning process of the topics addressed in it since the system focus on the users' difficulties and when he identifies at least three errors of the same subject, he presents tips or examples related to the issue that is being worked on.

The characteristics of autonomy, deliberativeness, reactivity, organization, ability to socialize, interaction of the agents as well as their ability to be multi-threaded contributed to the modularization of the system. The use of this technology allowed each agent to have specific characteristics that collaborated with achieving of the objectives of Mastem.

The property of the JADE framework to be multi-thread was fundamental to Mastem since agents are permanently active and involved in an infinite loop to observe your

environment, update its internal state, select and perform an action. Moreover, it was observed that the use of this framework on Mastem resulted in a drop in performance since it depends on the communication of the agents.

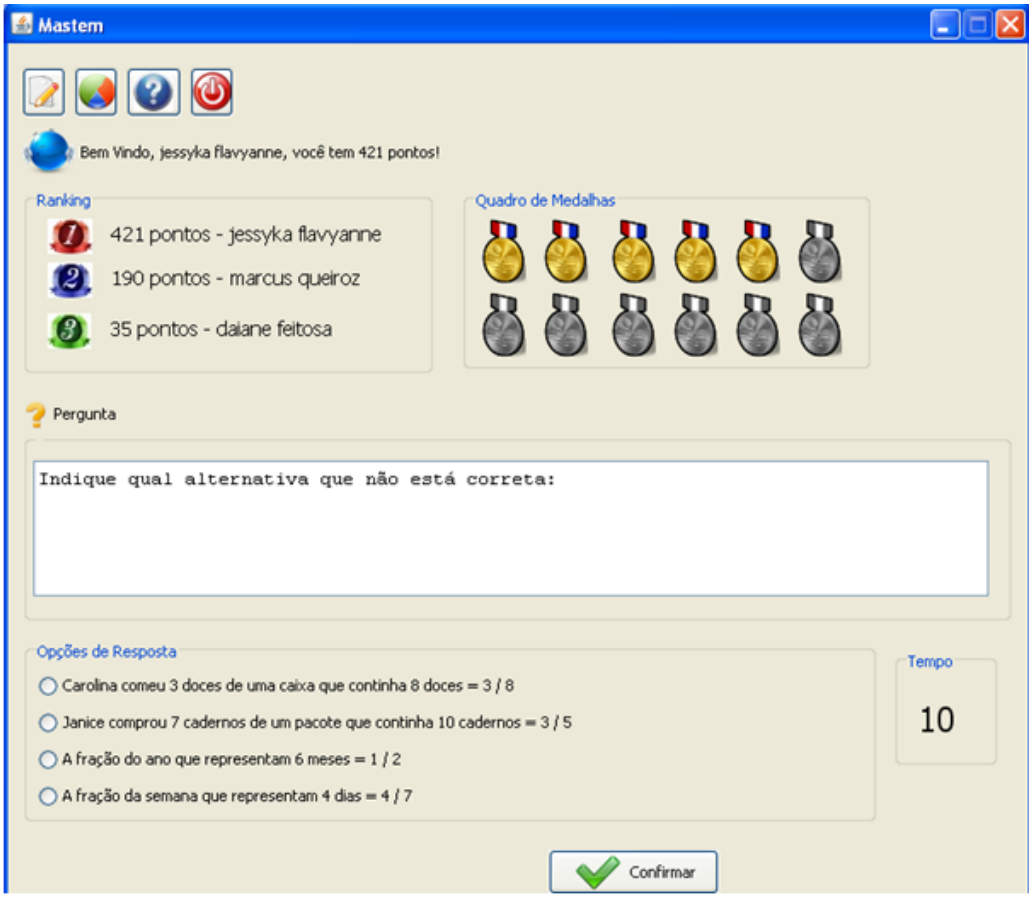


Figure 2. Main Screen of Student User.

The Tropos methodology proved to be suitable for the development of Mastem since it is a complete methodology that covers from the stage of the requirements collection to system implementation. The i* framework and language AUML allowed the full specification of Mastem through the models of strategic dependency and strategic rationale as well as class diagrams, sequence and communication respectively.

Some difficulties were obtained during the implementation of Mastem regarding the JADE framework. Among them, we can mention the complexity of creating agents automatically in software and the creation of an installer.

5 Ongoing and future work

As future work, we intend to create an installer for the Mastem and make it available under the GNU terms. Moreover, for the ranking be composed of students from different schools it is necessary to create and maintain a server that would be responsible for managing the accounts of all Mastem users. Finally, we intend to conduct a formal validation with mathematics teachers and consequently the use of Mastem by students during class.

6 References

1. Gago, V. Werneck, R. C.: Modeling an Educational Multi-Agent System in MaSE. In: Liu, Wu, J., Yao, Y., Nishida, T. (eds). Active Media Technology, vol. 5820, pp. 335–346. Springer, Heidelberg (2009)
2. Índice de Desenvolvimento da Educação Básica, http://portaliideb.inep.gov.br
3. Castro, J., Kolp, M., Mylopoulos, J.: Towards requirements-driven information systems engineering: The tropos project. In: Information Systems, vol. 27, no. 6, pp. 365-389 (2002)
4. Java Agent DEvelopment Framework, http://jade.cselt.it
5. Chung, L., Nixon, B., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. Kluwer Academic Publishers, Boston (1999)
6. Silva, I. G. L. da.: Projeto e Implementação de Sistemas Multi-Agentes: O Caso Tropos (in Portuguese). Dissertation. Universidade Federal de Pernambuco, Brazil, 2005.
7. Astah Community, http://astah.net/editions/community
8. Odell, J., Parunak, H. V. D., Bauer, B.: Extending UML for agents. In: Proc. of the Second International Bi-Conference Workshop on Agent-Oriented Information Systems, Austin, pp. 3–17 (2000)
9. Openome, http://www.cs.toronto.edu/km/openome/
10. Vilela, J. F. F.: Projeto e Implementação de um Software Educativo Multi-Agente com Tropos e JADE (in Portuguese). Monograph. Universidade Federal do Vale do São Francisco, Brazil, 2011.

Avaliação de Modelos i* com o Processo AIRDoc-i*

Cleice Souza[1], Cláudia Souza[1], Fernanda Alencar[2], Jaelson Castro[1], Paulo Cavalcanti[1], Monique Soares[1], Gabriela Guedes[1], Eduardo Figueiredo[3]

[1]Centro de Informática – Universidade Federal de Pernambuco (UFPE)
[2]Eletrônica e Sistemas – Universidade Federal de Pernambuco (UFPE)
[3]Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)
{ctns, jbc, plc2, mcs4, ggs}@cin.ufpe.br; {ccnsouza, fernandalenc}@gmail.com; figueiredo@dcc.ufmg.br

Resumo. Engenharia de Requisitos é uma atividade chave no processo de desenvolvimento de software, pois ela é responsável pelo entendimento e definição do problema e das necessidades dos usuários. A abordagem GORE (*Goal-Oriented Requirements Enginering*) apresenta como foco capturar as reais intenções dos stakeholders através de técnicas como: V-Graph, NFR-framework, KAOS e a linguagem i*. A linguagem i* se caracteriza por modelar os objetivos e as intencionalidades dos atores de um contexto organizacional. Entretanto, os modelos produzidos em i* não estão livres de erros oriundos do mau uso dos construtores sintáticos de modelagem. Assim, faz-se necessário um processo de avaliação capaz de verificar a sintaxe da linguagem e propor correções em modelo i*. Neste sentido, este trabalho apresenta um processo proposto chamado de AIRDoc-i*. Neste processo, foram acrescidas atividades sequenciais para avaliar modelos i*.

Palavras-chave: Engenharia de Requisitos, Linguagem i*, processo AIRDoc.

1 INTRODUÇÃO

A Engenharia de Requisitos (ER) corresponde à atividade de entendimento das necessidades dos usuários no contexto do problema a ser resolvido [1]. Problemas da ER é amplamente pesquisados no Brasil [16] e no mundo. A ER pode representar as necessidades do usuário através de metas e intenções. Esta forma é usada na abordagem *Goal Oriented Requirements Enginering (GORE)* [2], que tem como foco capturar as reais intenções dos *stakeholders* por meio de metas que eles pretendem satisfazer.

Entre as várias técnicas GORE, podemos citar as linguagens V-Graph [3], NFR-framework [4], KAOS [5] e a linguagem i* [6]. Esta última é usada neste trabalho por ser uma linguagem bastante difundida na comunidade de ER [7, 13, 14]. A linguagem i* se caracteriza por modelar objetivos e as intencionalidades dos atores e suas dependências dentro de um contexto organizacional, permitindo descrever os relacionamentos estratégicos e intencionais em alto nível de detalhamento.

É importante ressaltar que modelos i* descreve detalhadamente os participantes do processo e o contexto que está sendo modelado o que facilita o seu entendimento. Santos [8], por exemplo, ressalta a importância dos modelos i* apresentarem descrições detalhadas do problema e interações modelados. Portanto, devem existir métodos avaliativos que verifiquem se essas descrições seguem o uso correto da linguagem i*. Não basta aos modelos i* estarem bem detalhados, eles também precisam estar corretos com relação a sintaxe da linguagem [6]. Em um trabalho anterior [17], nós apresentamos um processo chamado de ***A**pproach to **I**mprove **R**equirements **Doc**uments for **i*** models* (AIRDoc-i*) para avaliar a sintaxe de modelos i*. Este novo processo surgiu através da análise de atividades e de artefatos do processo AIRDoc [9] que avalia a sintaxe de modelos de casos de uso com objetivo de identificar os erros e quantificá-los.

Este artigo estende o processo AIRDoc-i* através da análise das atividades e dos artefatos do processo AIRDoc [9] foi montado um processo gráfico em BPMN [10] que detecta erros sintáticos da linguagem i* em seus modelos. Além disso, o novo processo avalia os modelos i*, identifica, armazena, e quantifica os erros e os corrige através de atividades sequenciadas do processo AIRDoc-i*.

O restante deste artigo esta organizado da seguinte forma. A seção 2 apresenta o objetivo da pesquisa. A seção 3 apresenta as principais contribuições do trabalho. A seção 4 as conclusões do trabalho. A seção 5 discute trabalhos futuros e em andamento.

2 OBJETIVOS DA PESQUISA

O principal objetivo da pesquisa foi responder a pergunta: *Como o processo AIRDoc-i* detecta e corrige erros sintáticos da linguagem i* ?*.

Para responder a pergunta da pesquisa foi realizada uma análise das atividades do processo AIRDoc-i* [17] aplicando-o a modelos i*. Para isso, foi necessário adequar as atividades do processo, pois a linguagem i* apresenta características próprias [6] que a torna diferente da linguagem de modelos de casos de uso UML [11]. O nosso objetivo foi alcançado através de melhorias do processo AIRDoc [9] que agora adaptado é um processo que avalia a sintaxe dos modelos i*. Este processo tem o intuito de avaliar e melhorar os construtores da linguagem i* em seus modelos.

3 CONTRIBUIÇÕES CIENTÍFICAS

Esta seção mostra a contribuição do trabalho, o processo AIRDoc-i* cujo objetivo principal é detectar erros sintáticos e corrigi-los. Temos o processo, mostrado na Fig. 1, modelado em BPMN (*Business Process Modeling Notation*) [10]. BPMN foi escolhida por utilizar ícones padrão que facilita a modelagem do processo AIRDoc-i*. Este processo é composto por fases, atividades e artefatos para avaliar os modelos i*, detectar os erros, armazenar os erros, quantificá-los e corrigi-los através de atividades sequenciadas. O processo AIRDoc-i* é explicado nas seguintes seções

3.1 AIRDoc-i*

O processo AIRDoc-i* está dividido em dois estágios que são Avaliação e Melhoria. O estágio Avaliação consiste em quatro Atividades: (E.1) Elaboração do plano de atividade; (E.2) Definir as Atividades do GQM; (E.3) Coletar os Valores da Métrica; e (E.4) Interpretar as Atividades do GQM. Além disso, o estágio de Melhoria consiste em duas Atividades: (I.1) Identificar as Correções e (I.2) Aplicar as Correções.

Primeiro Estágio: Avaliação

Este estágio apresenta como foco principal identificar, apontar e contar os erros sintáticos encontrados nos modelos i*.

E.1 Elaboração do plano de atividade. Nesta atividade, são definidas as pessoas para trabalhar no processo, as ferramentas, os modelos i*, os cargos, as datas, os prazos, as equipes, e o tempo para compor o projeto. As informações produzidas nesta atividade serão armazenadas nos artefatos 1, 3, 4, 5, 6 e 7.

E.2 Definir as atividades do GQM. Nesta atividade, são definidos o objetivo da avaliação, a pergunta da avaliação, a métrica, e a estrutura da avaliação. As informações produzidas nesta atividade são armazenadas nos artefatos 2, 8, 9, e 10.

E.3 Coletar os valores da métrica. Antes de realizar a coleta dos valores da métrica deve ser realizada a identificação dos erros sintáticos nos modelos i*, que acontece da seguinte forma:

1º passo: Realizar uma inspeção no modelo i* que será avaliado;

2º passo: Realizar uma inspeção no catálogo de erros do i*;

3º passo: Comparar a sintaxe do modelo i* com a sintaxe do catálogo.

Ao realizar estes passos, é possível identificar se o modelo apresenta erros sintáticos e apontá-los. Caso apresente erros sintáticos a métrica a ser aplicada nos modelos i* chama-se número de erros (E_1) retirada do trabalho de [8]. A métrica número de erros (E_1) está inserida no AIRDoc-i*. O valor da métrica (E_1) é variável de acordo com erros sintáticos encontrados. Após identificar os erros sintáticos o resultado pode ser coletado e armazenado no artefato 11.

E.4 Interpretar as Atividades do GQM. Nesta atividade será interpretado o resultado da métrica utilizando o objetivo e as perguntas da atividade E.2. As informações produzidas nesta atividade serão armazenadas nos artefatos 12 e 13.

Segundo Estágio: Melhoria

O segundo estágio é composto por duas atividades que são: Identificar as Correções e Aplicar as Correções. Este estágio apresenta como foco corrigir os erros sintáticos encontrados nos modelos i*. As correções serão realizadas através da consulta do catálogo de erros frequentes em i*.

I.1 Identificar as Correções. Nesta atividade ocorre a identificação da correção. A identificação acontece através da inspeção no catálogo de erros frequentes em i*, artefato 14, que exibe a forma certa de utilizar os construtores sintáticos linguagem. O artefato 13 armazena os erros sintáticos encontrados no modelo i*.

I.2 Aplicar as Correções. Nesta atividade são aplicadas as correções dos erros sintáticos identificados nos modelos i* localizados no Catálogo de Erros Frequentes

em i*. O artefato 15 inclui todos os modelos i* corrigidos. As correções são aplicadas de acordo com erro. Por exemplo, o modelador identifica o erro, consulta a localização do erro no catálogo e, posteriormente, observa no catálogo a forma correta de realizar a modelagem e aplica a correção no erro encontrado.

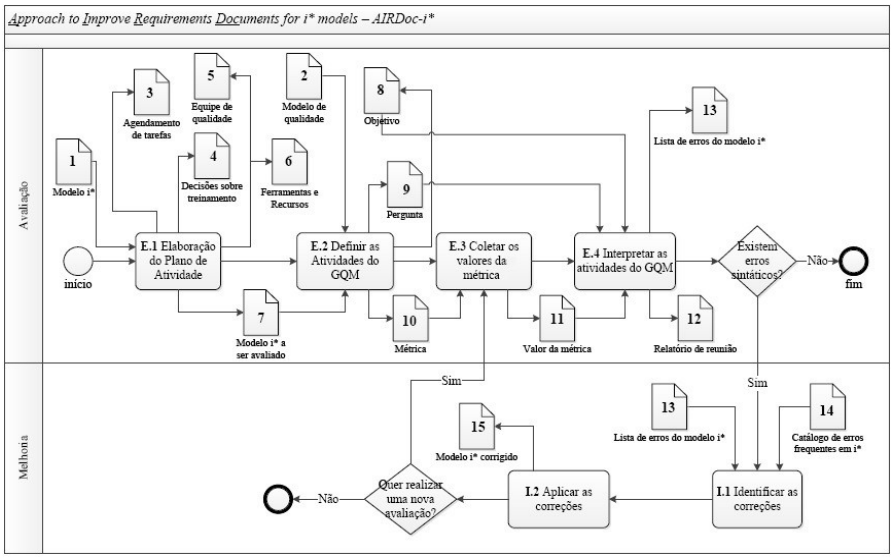


Fig. 1 O processo AIRDoc-i*

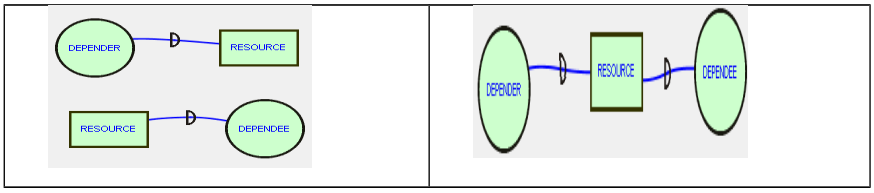
3.2 Catálogo de erros frenquentes da linguagem i*

Modelos i* não estão livres de descrições sintáticas erradas que prejudicam o entendimento por parte dos interessados. Para facilitar a identificação dos erros que podem aparecer nos modelos i* o catálogo proposto anteriormente [8] foi incrementado com novos erros sintáticos. Este novo catálogo é chamado de *Catálogo de Erros Frequentes da Linguagem i**. Os objetivos deste novo catálogo são:

- Ser inserido dentro do processo avaliativo de modelos i* chamado de AIRDoc-i*, para corrigir os erros sintáticos encontrados nos modelos i*; e
- Ser usado pela pessoa interessada em aumentar seus conhecimentos sobre erros sintáticos que os modelos i* podem apresentar.

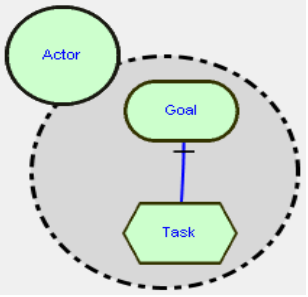
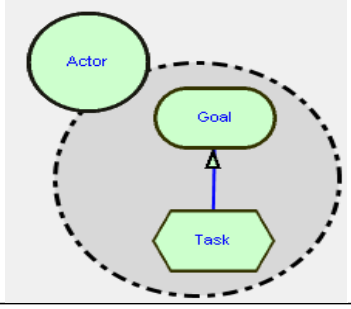
Tabela 1. Ligação de dependência.

Ligação de Dependência 001	Ligação de dependência só deve acontecer se houver *depender*, *dependum* e *dependee*.
Ligação de dependência sem *dependum*.	O *dependum* só deve ter uma única ligação vinda do *depender* e uma ou mais ligações indo para *dependee diferentes*.
Errado	Certo



A seguir estão alguns dos novos erros inseridos dentro do catálogo de erros frequentes em i*. A Tabela 1 apresenta o erro de Ligação de Dependência, no qual a ligação de dependência só deve acontecer se houver *depender*, *dependum* e *dependee*. Além disso, o *dependum* só deve ter uma única ligação vinda do *depender* e uma ou mais ligações indo para *dependee diferentes*. Ademais, a Tabela 2 apresenta o erro de *Goal Refinement*, no qual todo *goal* só deve ser o fim (*end*) num relacionamento do tipo *means-end link*.

Tabela 2. Refinamento: Goal.

Goal Refinement 001	*Goal element* deve ser o fim (*end*) em uma ligação do tipo *means-end*
Goal sendo refinado através de uma *task-decomposition*.	O elemento interno *goal* deve ser satisfeito através de uma ligação de *means-end link*.
Errado	Certo
	

4 CONCLUSÕES

Este artigo apresenta como contribuição um processo chamado AIRDoc-i* [17] que identifica e corrige os erros sintáticos encontrados em modelos i*. Este processo possui quatro atividades que são Elaboração do Plano de Atividade, Definir as atividades do GQM, Coletar os valores das métricas e Interpretar as atividades do GQM. Estas atividades têm como objetivo definir a equipe de avaliação, definir a pergunta, a métrica, apontar os erros sintáticos nos modelos i* e coletar o resultado da métrica. A fase de Melhoria possui duas atividades que são Identificar as Correções e Aplicar as Correções. A identificação das correções acontece quando o *stakeholder* recorre ao catálogo de erros frequentes em i* para encontrar a forma correta de realizar aquela modelagem na qual o erro identificado. A aplicação da correção do modelo i* acontece na segunda atividade desta fase produzindo um modelo i* melhorado.

5 TRABALHOS FUTUROS EM ANDAMENTO

Como trabalhos futuros, nós pretendemos fazer: (i) criar extensões do AIRDoc-i* para avaliar a completude, a ambiguidade e o detalhamento dos modelos i*, além de detectar erros semânticos; (ii) aumentar o catálogo de erros frequentes da linguagem i*, com o propósito de incluir novos erros sintáticos e adicionar erros semânticos que ocorram com frequência em modelos i*; (iii) especificar em OCL (*Object Constraint Language*) [12] os erros do Catálogo de Erros Frequentes da Linguagem i*, cuja importância é definir condições e/ou restrições para os construtores da linguagem i*, e (iv) aplicar o processo em aplicações modeladas em i* como linha de produtos de software [18] e sistemas multi-agentes [15].

REFÊNCIAS

1. Asghar, S., Umar, M.: Requirement Engineering Challenges in Development of Software Applications and Selection of Customer-off-the-Shelf (COTS) Components. International Journal of Software Engineering (IJSE), Volume (1): Issue (1), 2010.
2. Lamsweerde, A.: Goal-oriented requirements engineering: A guided tour. In: 5th IEEE International Symposium on Requirements Engineering (p. p. 249). Washington, DC, 2001.
3. Yu, Y., Leite, J. C., Mylopoulos, J.: From goals to aspects: Discovering aspects from requirements goal models. 12th Intl. Conf. on Requirements Engineering. 2004.
4. Chung, L., Nixon, B., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. 1. ed. [S.l.]: Springer, 1999.
5. Lamsweerde, A. V.: Requirements Engineering in the Year 00: A Research Perspective. In: Keynote paper, Proc. ICSE'2000 - 22nd Intl. Conference on Software Engineering. IEEE Computer Society Press, 2000.
6. Yu, E.: Modelling Strategic Relationships for Process Reengineering. Tese (Doutorado). Canadá: University of Toronto, Department of Computer Science, 1995.
7. Horkoff, J. M. : Using i* Models for Improvement. Dissertação (Mestrado). Toronto, Canadá: Department of Computer Sciences, 2006.
8. Santos, E. B. D.: Uma Proposta de Métricas para Avaliar Modelos i*. Dissertação (Mestrado). Recife: UFPE, 2008.
9. Ramos, R. A.: AIRDoc – An Approach to Improve the Quality of Requirements Documents: Dealing with Use Case Models. Tese (Doutorado). Recife: UFPE, 2009.
10. BPMI. Business Process Modeling Notation, OMG Avaible Specification. Object Management Group, 1.1 edition. 2008.
11. UML. Unified Modeling Language. 2009. Disponível em: http://www.uml.org/. Acesso em: 20 Setembro de 2011.
12. OMG. Object Management Group. 2001. OCL 2.0. Acesso em 23 de dezembro de 2012, disponível em Object Constraint Language: OMG Available Specification: http://www.omg.org/spec/OCL/2.0/
13. J. M. Conejero, E. Figueiredo, A. Garcia, J. Hernandez, and E. Jurado. On the Relationship of Concern Metrics and Requirements Maintainability. In Information and Software Technology (IST), 54(2), pp. 212-238, 2012.
14. D. Demerval; M. Soares; F. Alencar; E. Santos; C. Souza. Towards an i*-based Architecture Derivation Approach. In: Fifth Internatiol i* Workshop, Trento, 2011.
15. A. Oliveira, L. Cysneiros, J. Leite, E. Figueiredo, and C. Lucena. Integrating Scenarios, i*, and AspectT in the Context of Multi-Agent Systems. In proceedings of

the 16th Conference of the Center for Advanced Studies on Collaborative research (CASCON), Article No. 16. Ontario, Canada, 2006.
16. K. Oliveira; J. Pimentel; E. Santos; D. Demerval; G. Souza; C. Souza; M. Soares; J. Castro; F. Alencar; C. Silva. 25 years of Requirements Engineering in Brazil: a systematic mapping to WER 2013. Workshop Engineering Requirements, 2013.
17. C. Souza,; F. Alencar ; G. Souza ; M. Soares ; C. Souza; R. Ramos ; J. Castro. AIRDoc-i*: um Processo para Avaliação de Modelos i*. In: 15th Workshop on Requirements Engineering, Buenos Aires, 2012.
18. G. Souza; C. Silva; J. Castro ; M. Soares; D. Demerval ; C. Souza. GS2SPL: Goals and Scenarios to Software Product Lines. In: 24th International Conference on Software Engineering & Knowledge Engineering (SEKE), 2012.

RETRATOS: Requirement Traceability Tool Support

Gilberto Cysneiros Filho[1], Maria Lencastre[2], Adriana Rodrigues[2], Carla Schuenemann[3]

[1] Universidade Federal Rural de Pernambuco, Recife, Brazil
g.cysneiros@gmail.com
[2]Universidade de Pernambuco, Recife, Brazil
mlpm@ecomp.poli.br, adriana.rodrigues@hotmail.com
[3]Universidade Federal de Pernambuco, Recife, Brazil
carlotcha@gmail.com

Abstract. Software traceability is the ability to relate artefacts created during the development life cycle of software system. Traceability is essential in the software development process and it has been used to support several activities such as impact analysis, software maintenance and evolution, component reuse, verification and validation. Moreover, the importance of traceability in the software development process has been endorsed by several standards for quality management and process improvement such as ISO 9001:2000 and CMMI. Despite the importance of software quality, current support for traceability is inadequate. In this paper, we present a tool that tackle different aspects and issues of the traceability problem. In particular, the tool support a rule based approach to capture traceability relations between software models. The rules can be created to capture traceability relations of different types of software models.

Keywords: Software traceability, rule-based approach, traceability visualization.

1 Introduction

Software traceability has been defined as "the ability to describe and follow the life of a requirement, in both a forward and backward direction (i.e. from its origins, through its development and specification, to its subsequent deployment and use, and through periods of ongoing refinement and iteration in any of these phases)" [1].

Traceability relations can guarantee and improve software quality and can help with several tasks such as the evolution of software systems, reuse of parts of the system, validation that a system meets its requirements, understanding of the rationale for certain design decisions, identification of common aspects of the system, and analysis of implications of changes in the system.

Traceability has been studied for many years and several approaches have been proposed to tackle its different aspects and issues. Pohl [2] states that a traceability approach should provide answers to the following questions:

- What traceability information should be captured?
- How traceability information should be captured?
- How traceability information should be stored?

Sherba adds in [3] that a traceability approach should also to answer the following question:

- How traceability relations are going to be viewed and queried?

Although several approaches have been proposed in the literature, in general they only address one aspect of the traceability problem. This makes more difficult to take advantage of the benefit to use a traceability tool in an industrial setting.

This paper presents a work in progress on a tool that address all the four questions above presented. In particular, the tool supports automatic generation of traceability relations, consistency and completeness checking in models created during the development life cycle of software systems. The tool is based on a rule-based approach that allows capturing traceability relations between different types of models (e.g. BPMN, UML, Java code) created during the development of software systems. Rules can be created to define what and how the traceability information should be captured. A visual editor to create rules and a visualization tool to support different forms of visualization is been developed.

2 Objectives

The goal to develop RETRATOS tool is to extend and tackle some weak points of the traceability approach of one the authors presented in [4,5,6]. The approach was developed to support (i) automatic generation of traceability relations between heterogeneous software models created during the development of multi-agent systems, and (ii) identification of missing elements in these various software models created during the development of multi-agent systems (completeness checking).

The figure 1 presents an overview of our framework. As shown in the figure, initially, the models of our concern represented in their native format are generated using proprietary tools (e.g. TAOME4E, Astah, or any diagram editor tool). These models are translated into XML format by using a Model Translator component based on XML Schemas proposed for the models, whenever the tools used to create the models do not generate them directly in XML.

The XML based models and rules are used as inputs to the Traceability Generator and Consistency Checker component to generate traceability relations between the models and to identify missing elements based on the rules. The engine also uses WordNet to support the identification of synonyms between the names of elements in the models. The WordNet is important component because in general naming conventions can change from high-level models (e.g. i*) to low-level representations (e.g. Java code).

The traceability relations and identified missing elements are represented in an XML document. The use of a separated document to represent the traceability relations and missing elements is important to preserve the original models, to allow the use of these models by other applications and tools, and to allow the generated

relations to be used to support the identification of other traceability relations that depend on the existence of previously identified relations.

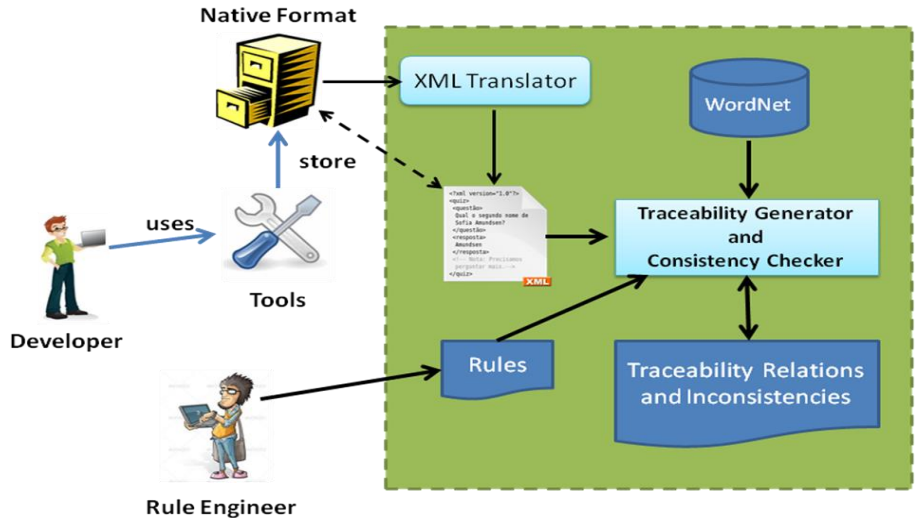


Figure 1- Cysneiros´s Approach

3 Our Approach

The Cysneiros´s approach only address the problem of identifying traceability relations. The figure 2 shows that two components are added to the original approach: a traceability visualization tool and a rule editor tool.

Our experience has shown that a large number of traceability relations can be generated for the various models. Therefore visualization support is fundamental to: i) allow the user to browse the traceability relations through various types of user interactions; ii) allow the user to add, remove, and modify properties of existing traceability relations and their related artefacts; iii) integrate with tools used to develop, test and maintain the system; iv) capture and maintain browsing history for traceability relations; v) support user querying and filtering of the traceability relations; vi) offer flexible and user customizable view of the traceability data; vii) provide tools to analyse and summarize the data on the traceability process and relations.

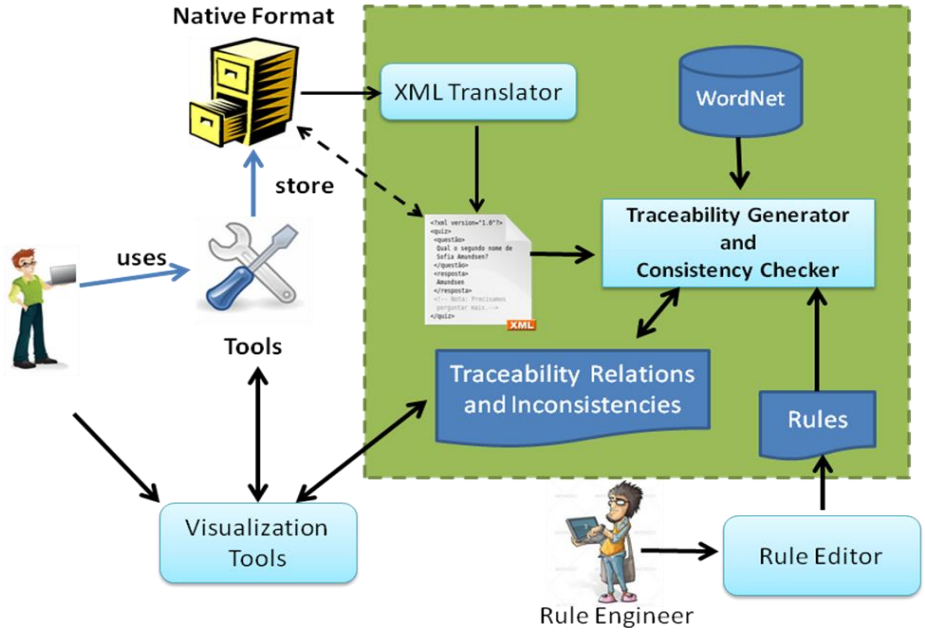


Figure 2 - RETRATOS overview

Currently, the approach support matrix, tree and Sunburst visualization techniques and generation of reports in HTML (see Figure 3).

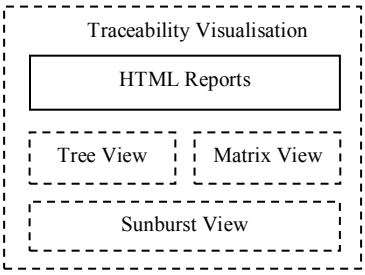


Figure 3 – Visualization Tools

HTML reports (see Figure 4) show the total number of traceability relations identified by the tool and to each relation created shows: (i) id of the rule used (e.g. rule1), (ii) type of the relation (e.g. overlaps), (iii) name of the elements (e.g. Allocate Runway Slot) and (iv) element´s type (e.g. SD goal). A more comprehensive and customizable reporting tool is been built.

Root element of the doc is Traceability Total no of traceability relations : 85

Traceability Relations Types between *i and Prometheus**

Rule ID	Type	SD Goal	Goal
rule1	overlaps	Allocate Runway Slot	Allocate Runway Slot
rule1	overlaps	Find Best Landing Time for an Aircraft	Landing
rule1	overlaps	Find Best Landing Time for an Aircraft	Find Best Land Time for an Aircraft
Rule ID	**Type**	**SR Goal**	**Goal**
rule3a	overlaps	Allocate Runway Slot	Allocate Runway Slot
rule3a	overlaps	Find Best Landing Time for an Aircraft	Landing
rule3a	overlaps	Find Best Landing Time for an Aircraft	Find Best Land Time for an Aircraft
Rule ID	**Type**	**SR Task**	**Goal**

Figure 4 - HTML Report

Sunburst is a graph (see Figure 5), which nodes are arranged in a radial layout. Nodes are drawn on adjacent rings representing a tree structure. Each child of a node with depth n is represented in the ring n + 1 on the same radian space as its parent(s). Sunburst visualization performs better on large amounts of nodes than traditional tree view representation. Tree view representations grow rapidly in the vertical direction if many branches are expanded while using Sunburst the nodes are distributed uniformly in all directions.

Figure 5 – Sunburst visualization

The Cysneiros´s approach relies on the use of traceability and completeness checking rules specified in an extension of XQuery. The creation of these rules is not a straightforward activity and requires knowledge of XQuery. To address this problem we are developing a visual rule editor that allows to create rules using a drag and drop approach. The editor also provides a graphical interface that allows selecting only the rules that are applicable to a specific software project or domain.

4 Conclusion and Future works

In this paper, we described a traceability tool that extends a rule-based approach to identify automatically traceability links and missing information between artifacts created during the development of software systems. The traceability tool adds support to traceability visualization and visual creation of rules.

One of the challenges to build a traceability tool is to provide support to the vast number of methodologies, platforms, tools, and languages available to develop software. Currently, the tool supports models created to the development of multi agent systems when using i* framework, Prometheus methodology, and JACK code. Our goal is to use and evaluate the approach in the other programming paradigms such as web services and object oriented programming.

As future work, we also intend to extend the tool to other types of visualization techniques and evaluate these techniques in terms of usability.

References

1. Gotel O. and Finkelstein A.: An Analysis of the Requirements Traceability Problem. International Conference on Requirements Engineering. - Colorado, USA: IEEE Computer Society. (1994) 94-101
2. Pohl k. Process-Centered Requirements Engineering. John Wiley & Sons, Inc., New York, NY, USA (1996)
3. Sherba S. and Anderson K.: A Framework for Managing Traceability Relationships between Requirements and Architectures. Second International Workshop from Software Requirements to Architecture (2003)
4. Cysneiros G. and Zisman A.: Traceability and Completeness Checking for Agent-Oriented Systems. 23rd Annual ACM Symposium on Applied Computing (2008)
5. Cysneiros G. and Zisman A.: Traceability for Agent-Oriented Design Models and Code. TEFSE/GCT'07. - Lexington, KY, USA (2007)
6. Cysneiros G. and Zisman A.: Tracing Agent-Oriented Systmes. The Nineteenth International Conference on Software Engineering and Knowledge Engineering (2007)

Integrando Modelagem Organizacional ao Processo de Engenharia de Requisitos

Victor Santander, Ivonei Freitas da Silva, Elder Elisandro Schemberger

UNIOESTE – Universidade Estadual do Oeste do Paraná, Laboratório de Engenharia de Software, Cascavel – PR – Brasil
victor.santander@unioeste.br,ivonei.silva@unioeste.br,
elderes@gmail.com

Resumo: O uso de modelos organizacionais no contexto do processo de engenharia de requisitos tem sido alvo de pesquisas nos últimos anos. Um dos principais desafios percebidos consiste em integrar modelos organizacionais às demais etapas do processo de engenharia de requisitos. Neste trabalho apresentamos nossa pesquisa relacionada à integração do framework i* aos modelos funcionais na engenharia de requisitos bem como à perspectiva de incluir a técnica BPMN neste processo de combinação. Também são apresentadas algumas dificuldades e desafios associados a esta temática considerando os trabalhos já realizados por nosso grupo de pesquisa.

Keywords. Engenharia de Requisitos, Framework i*, Modelagem Funcional

1 Introdução

A utilização de modelos organizacionais no processo de engenharia de requisitos tem recebido destaque nas pesquisas da área nos últimos anos. Entre os benefícios percebidos com o uso destes modelos estão: A melhora do entendimento do ambiente organizacional no qual o software atuará; Elicitação e modelagem das metas estratégicas da organização e possíveis meios de satisfação das mesmas; e, A possibilidade de representação das relações existentes entre os atores da organização bem como melhor compreensão dos fluxos de negócio. Mais recentemente tem se buscado utilizar os elementos presentes em modelos organizacionais em outras atividades e artefatos do processo de engenharia de requisitos. Alguns exemplos incluem a possibilidade de gerar artefatos de modelagem funcional em UML tais como casos de uso e diagramas de classe, gerar descrições textuais de requisitos, e apoiar atividades da engenharia de requisitos orientada a objetivos (GORE).

Do ponto de vista de engenheiros de requisitos que se defrontam diariamente com os desafios inerentes a área tais como clientes com dificuldades em expressar o que desejam, cronograma reduzido, pressão para redução de custos e requisitos evoluindo constantemente, percebe-se como benéfico utilizar os modelos organizacionais para apoiar a realização das demais etapas do processo de engenharia de requisitos. Isto significa especificar requisitos consistentes com as metas da organização, mais completos e não ambíguos por serem derivados de discussões que incluem todos os atores da organização envolvidos e representados nos modelos, possibilidade de antecipar

prioridade e volatidade de requisitos com base em elementos organizacionais estratégicos bem como melhorar o processo de rastreamento de requisitos desde a sua origem baseado nas necessidades organizacionais.

Considerando essas possíveis melhorias, o framework de modelagem organizacional i* [16] tem se destacado por permitir representar as intencionalidades do ambiente organizacional sob a forma de dependências entre atores do ambiente (Modelo de Dependências Estratégicas (SD)). Estas dependências podem ser do tipo objetivo, recurso, tarefa e objetivo-soft e as razões associadas a sua satisfação/realização podem ser expressas em um modelo denominado de Razões Estratégicas (SR). Em [9] [10], propõe-se o uso deste framework para gerar casos de uso em UML, tanto na forma diagramática quanto textual. Contudo, esta proposta precisa ser complementada incluindo o estudo de outras perspectivas de modelagem de processos de negócio.

Neste contexto, cabe destacar a técnica BPMN, a qual é um padrão de modelagem de processos de negócio desenvolvido pelo BPMI e mantido pelo OMG, bastante utilizado atualmente. Esta técnica tem sido apontada como de fácil aprendizado e uso como também suportada por diversas ferramentas computacionais. Mais recentemente, na abordagem proposta em [18], são apresentadas diretrizes para que a elicitação e análise de requisitos de sistemas computacionais possa ser apoiada pela modelagem de processos de negócios com o uso da técnica BPMN. Na nossa pesquisa, i* e BPMN são a base da investigação para auxiliar engenheiros de requisitos na difícil tarefa de especificação de requisitos sendo que os trabalhos já citados são o ponto de partida para os estudos.

Desta forma, neste artigo são apresentadas algumas contribuições já realizadas pelo grupo de pesquisa "Laboratório de Engenharia de Software – LES" da UNIOESTE sobre a temática exposta bem como algumas metas e desafios associados à linha de pesquisa. Os estudos já realizados, em sua maioria, referem-se aos trabalhos do primeiro autor deste artigo, o qual tem trabalhado com a integração de modelagem organizacional i* com Engenharia de Requisitos. O terceiro autor recentemente tem iniciado estudos sobre integração de BPMN com a Engenharia de Requisitos, sendo o seu trabalho essencial para a proposta neste artigo. O segundo autor, embora tenha colaborado com o primeiro autor em alguns estudos, tem focado atualmente suas pesquisas no contexto de linhas de produtos de software. Futuramente, os autores pretendem extender este trabalho para este novo paradigma de desenvolvimento de software.

2 Objetivos da Pesquisa

O foco de nossa pesquisa está na definição de estratégias para apoiar a utilização de modelos organizacionais i* e BPMN, de forma integrada, nas demais etapas do processo de engenharia de requisitos. Estas etapas basicamente consistem da elicitação, análise, validação e documentação de requisitos tanto em forma textual quanto em diagramas UML. Para apoiar a validação das estratégias propostas, propõe-se o uso das mesmas na especificação de requisitos de sistemas computacionais em vários domínios de aplicação bem como em âmbito acadêmico conforme relatado em [5].

Adicionalmente, objetiva-se estudar como as variações de i* apresentadas na proposta original descrita na tese de Eric Yu, no wiki i*, na metodologia Tropos e na GRL podem afetar as estratégias propostas bem como estudar trabalhos relacionados

ao uso de BPMN e impacto da técnica no processo de especificação de requisitos de sistemas computacionais. Neste primeiro momento a pesquisa não envolverá investigações relacionadas à gerencia de requisitos, embora alguns estudos já tenham sido realizados considerando o framework i* e a gerência de riscos. Após uma compreensão aprofundada de como modelos organizacionais i* e BPMN podem derivar modelos de requisitos, o grupo, então, investigará o gerenciamento de requisitos neste novo contexto.

3 Contribuições Científicas

3.1 Integrando o framework i* e BPMN ao processo de engenharia de requisitos

A geração de artefatos posteriores a modelagem organizacional na engenharia de requisitos é repleta de desafios. Especificar requisitos funcionais e não-funcionais é uma tarefa árdua que exige experiência e estabelecimento de um processo sistemático para elicitar e especificar estes requisitos em uma linguagem clara [13]. Neste contexto, também observa-se que em muitos casos, artefatos descrevendo requisitos de software são gerados sem um prévio desenvolvimento de modelos organizacionais. Isto tem gerado sistemas computacionais que não consideram as metas organizacionais e informações organizacionais estratégicas que são tipicamente especificadas na etapa de modelagem organizacional.

Considerando esta problemática, em [9] [17] apresenta-se uma proposta de derivação de requisitos funcionais na forma de casos de uso em UML a partir do framework i*. Na proposta de derivação de requisitos funcionais foram descritas as diretrizes que permitem mapear atores, casos de uso e descrições textuais a partir dos atores, metas, recursos, tarefas e softgoals expressos nos modelos organizacionais de Dependências Estratégicas (SD) e Razões Estratégicas (SR) em i*. Este processo de derivação é suportado pela ferramenta computacional JGOOSE que será apresentada na próxima seção. Considerando como base este trabalho, em [3] foi proposta uma abordagem para gerar casos de uso com aspectos a partir do framework i*. Nesta abordagem, aspectos são usados para expressar e modelar requisitos comuns que podem ser reutilizados. Por outro lado, visando avaliar o uso do framework i* nas etapas iniciais do processo de engenharia de requisitos e posterior utilização na derivação para Casos de Uso, em [6], [8] e [11] temos utilizado este framework na especificação de requisitos de software educacional para pessoas com deficiência visual. Isto tem nos possibilitado avaliar o uso do framework i* em sistemas reais com foco na avaliação de melhoria do processo de engenharia de requisitos.

Também cabe ressaltar as pesquisas do nosso grupo incluindo os trabalhos apresentados em [7] e [12] propondo o uso do framework i* para modelar, em um contexto de engenharia reversa, requisitos organizacionais e funcionais de sistemas legados construídos utilizando a abordagem estruturada de desenvolvimento de sistemas. Recentemente os estudos também têm incluído outras atividades do processo de engenharia de software, mais especificamente a gerência de projetos. Neste sentido, a proposta apresentada em [18] considera o uso do framework i* como base para a realização das atividades de gerência de riscos conforme proposto no modelo PMBOK.

Assim, com base nos trabalhos realizados acreditamos que o framework i* pode ser um forte aliado na especificação de requisitos de sistemas computacionais e uso em outras atividades do processo de engenharia de software/requisitos.

Por outro lado, é importante observar neste contexto, outras técnicas de modelagem organizacional como BPMN, amplamente usada no âmbito acadêmico e industrial. Neste sentido, cabe destacar o trabalho defendido em forma de dissertação de mestrado [18], cujo objetivo foi traçar diretrizes para que a Elicitação e Análise de Requisitos possa ser realizada por equipes geograficamente distribuídas, principalmente atuando no modelo de negócio chamado de *Offshoring* (caracterizado por empresas localizadas em países diferentes e que o relacionamento entre elas seja o de terceirização de tarefas). Estas diretrizes foram diretamente encaminhadas e apoiadas pela modelagem de processos de negócios com o uso da técnica BPMN.

A execução das diretrizes guia, com apoio da modelagem de processos de negócio segundo a notação BPM, a geração de Casos de Uso, bem como a concepção de um BRD – Business Requirement Document, que contém a especificação de requisitos funcionais e não funcionais a serem considerados nas etapas seguintes, a saber: implementação, testes, implantação e manutenção. Essa proposta foi aplicada em um projeto real, seguindo o protocolo da Engenharia Experimental, a fim de ser aferida sua viabilidade e validade como contribuição acadêmica e apoio direto na indústria de software na forma de metodologia de Elicitação e Análise de Requisitos.

Desta forma, considerando estes trabalhos, temos um indicativo de que é viável e benéfico especificar requisitos de software a partir de modelos organizacionais. Contudo, estas abordagens são complementares em vários aspectos e a integração de i* e BPMN deveria ser investigada para gerar um documento de requisitos mais completo e consistente.

Um aspecto essencial nas pesquisas já realizadas é a maneira como elas foram avaliadas empiricamente. Como o grupo pretende, futuramente, transferir tecnologia para a indústria de software, surge a necessidade de avaliação mais intensiva das diretrizes propostas e da ferramenta que está em evolução em projetos reais. Sendo assim, o grupo pretende adotar a metodologia proposta por [19], que propõe, sistematicamente, sucessivos estudos primários ou secundários para avaliação empírica e transferência de tecnologia para a indústria. Ao final da execução desta metodologia, espera-se que as empresas participantes dos estudos de avaliação empírica apropriem-se dos resultados de nossa proposta.

3.2 Ferramenta JGOOSE

A derivação de casos de uso em UML a partir do framework i* é apoiada pela ferramenta JGOOSE. A versão original desta ferramenta foi apresentada em [1] e [4]. Posteriormente, ela foi alvo de melhorias em [2] e [15], sendo que a última versão implementa as seguintes características: permite derivar atores e casos de uso a partir dos modelos em i*, permite derivar passos do cenário principal e secundário para cada caso de uso descoberto, permite alterar e salvar as descrições textuais dos casos de uso gerados bem como permite gerar o diagrama de casos de uso em formato XMI (XML Metadata Interchange) para ser importado por outras ferramentas. Esta ferramenta vem sendo usada em âmbito acadêmico conforme relatado em [5].

4 Conclusões

Especificação de Requisitos compreende uma área de conhecimento bastante importante no desenvolvimento de software. A literatura especializada é unânime ao apontar essa como fase fundamental de um desenvolvimento de software com qualidade, bem como afirma de maneira categórica que se mal realizada, pode gerar implicações no custo e no prazo de um projeto de software, entre outros problemas possíveis. Nesse contexto, há uma grande lacuna existente quando a Especificação de Requisitos é tratada unicamente de forma acadêmica ou, de forma isolada, apenas na indústria. É preciso, cada vez mais, que pesquisas nesta linha de trabalho agrupem esforços acadêmicos e alinhem estes esforços com limitações e/ou particularidades de empresas desenvolvedoras de software. A presente pesquisa visa contribuir diretamente para que a etapa de Especificação de Requisitos, apoiada pela modelagem organizacional com o framework i* e a técnica BPMN, estejam em conformidade com as características encontradas em um ambiente de produção de software, associando e mitigando lacunas existentes para que estas sejam minimizadas, favorecendo um melhor aproveitamento, em qualidade e quantidade, dos requisitos especificados.

5 Trabalhos futuros e em andamento

Considerando os desafios da pesquisa abordada, os trabalhos futuros conduzem inicialmente a estudos mais aprofundados, preferencialmente por meio de revisões sistemáticas, de i* e BPMN e integração dos mesmos com modelos funcionais, estudo e melhoria das propostas apresentadas em [9] e [18], e posterior definição de uma abordagem integrada voltada a especificação de requisitos de software com base nas técnicas escolhidas. Também a ferramenta JGOOSE deverá incluir suporte necessário a execução da abordagem proposta. Cabe destacar que esta ferramenta no momento está sendo melhorada para incluir um editor próprio para modelos em i* bem como para casos de uso. Estes editores serão agregados a ferramenta, facilitando a construção, evolução e integração dos modelos SD e SR em i* com casos de uso em UML e descrições textuais de requisitos.

6 Referências

1. Briscke, M., Santander, Victor Francisco Araya., Castro, J. F. B., GOOSE: Uma Ferramenta para Integrar Modelagem Organizacional e Modelagem Funcional In: Jornadas Chilenas de Computación - V Workshop Chileno de Ingeniería de Software, Valdivia, Chile. (2005).
2. Brischke, M. ; Santander, Victor F. A ; Silva, I. F., Melhorando a Ferramenta JGOOSE. In: 15th Workshop on Requirements Engineering, Buenos Aires, 24 a 27 de Abril.(2012).
3. Lizana, K. A., Santander, Victor Francisco Araya., Alencar, F. M. R., Castro, J. F. B., Diaz, J. S. Derivacion de Casos de Uso con Aspectos a partir de Modelos Organizacionales i* In: XII Conferencia Iberoamericana de Ingeniería de Requisitos y Ambientes de Software, p.253 - 258, Medellin. (2009).
4. Pedroza, F.P., Alencar, F.M.R., Castro, J., Silva, F.R.C., and Santander, V.F.A. Ferramentas para Suporte do Mapeamento da Modelagem i* para a UML: eXtended GOOD - XGOOD e GOOSE. In Proceedings of WER. 2004, 164-175.

5. Santander, Victor F. A . Avaliando a utilização da Técnica i* no Processo de Ensino e Aprendizagem na Engenharia de Requisitos - Um Relato de Experiência. In: IV Fórum de Educação em Engenharia de Software, XXV Simpósio Brasileiro de Engenharia de Software (SBES), São Paulo. (2011).
6. Santander, V. F. A., Silva, D. R.: Requirements Engineering Contributions on the Development of Educational Software for the Blind or People with Impaired Vision – An Experience Account. CLEI Electronic Journal 9. (2006).
7. Santander, Victor Francisco Araya., Vicente, Andre Abe., Koerich, Fabio., Castro, J. F. B., Modelagem de Requisitos Organizacionais, Não-Funcionais e Funcionais em Software Legado com Ênfase na Tecnica i* In: X Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes de Software. p.47 – 60, Isla Margarita. Caracas. (2007).
8. Santander, Victor Francisco Araya., Silva, D. R., Vicente, A. A., Castro, J. F. B., Utilizando a Técnica i* para Modelar a Concepção de Vigotski visando auxiliar o Processo de Desenvolvimento de Software Educacional para Pessoas com Deficiência Visual In: X Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes de Software. p.269 – 282, Isla Margarita, Caracas: F.Losavio, G.H.Travassos, V.Pelechano, I.Diaz, A. Matteo, (2007a).
9. Santander, V. F., Castro, J. F. B., Deriving Use Cases from Organizational Modeling In: IEEE Joint International Requirements Engineering Conference - RE'02, p. 32-39, Essen, Germany, (2002).
10. Santander, V. F. A., CASTRO, J. F. B. . Integrating Use Cases and Organizational Modeling. In: Simpósio Brasileiro de Engenharia de Software, 2002, Gramado. Brazilian Symposium on Software Engineering, Gramado, Rio Grande do Sul, October, 16-18. Porto Alegre: Edição Leila Ribeiro, 2002. v. 1. p. 222-253.
11. Santander, Victor Francisco Araya ; SILVA, D. R., Contribuições da Engenharia de Requisitos no desenvolvimento de software educacional para pesssoas cegas e pessoas com visão reduzida: um relato de experiência, In: Jornadas Chilenas de Computación, V Workshop Chileno de Ingeniería de Software, 2005, Valdivia, Chile.
12. Santander, Victor Francisco Araya ; SILVA, L. P. . Uma Proposta de Evolução em Sistemas Legados. In: VII Workshop on Requirements Engineering (WER), December, 9-10, 2004, Tandil, Argentina, v. 1. p. 201-213.
13. Teixeira, E.P., Santander, Victor Francisco Araya., Integrando a Teoria da Atividade e a Técnica i* na fase de Requisitos Detalhados. XIII Congresso Ibero Americano em Software Engineering, p.57 – 62. Cuenca, Equador, Universidad del Uzuay, (2010).
14. Varela, J. P. ; Santander, Victor Francisco Araya ; SILVA, I. F. . Integrando o Framework i* ao Processo de Gerência de Riscos. In: XV Ibero-American Conference on Software Engineering, 2012, Buenos Aires, 24 a 27 de abril. Anais do XV Ibero-American Conference on Software Engineering, 2012.
15. Vicente, André A., Santander, Victor Francisco Araya, Castro, Jaelson B., Freitas da Silva, Ivonei., Reyes Matus, Francisco G., JGOOSE: A Requirements Enginnering Tool to Integrate i* Organizational Modeling with Use Cases in UML. Ingeniare. Revista chilena de ingeniería. , v.17, p.6 - 20,(2009).
16. Yu, E.: Modelling Strategic Relationships for Processes Reengineering. Toronto, Canadá: University of Toronto, PhD Thesis (1995).
17. Yu, E., Giorgini, P., Maiden, N., Mylopoulos, J., Social Modeling for Requirements Engineering. The MIT Press (2011).
18. Schemberger, E. E., MoRE-GSD Uma abordagem para elicitação e análise de requisitos a partir das regras de negócio no contexto de desenvolvimento distribuído de software, Dissertação de Mestrado, Universidade Estadual de Maringá, Departamento de Informática, Maringá, PR, Março de 2013.
19. Spinola, R. O., Dias-Neto, A. C., and Travassos, G. H. (2008). Approach to develop software technology to support primary and secondary studies (in portuguese). In: Experimental Software Engineering Latin American Workshop - ESELAW, Salvador, Bahia, BRAZIL.

APAI: Uma Proposta de *Framework* para Análise e Projeto de Aplicações Interativas para TV Digital

Adicinéia Aparecida de Oliveira[1], Diego Armando de Oliveira Meneses[1], Andriel da Silva Argollo[1], Jislane Silva Santos de Menezes[1]

[1] Universidade Federal de Sergipe - Brasil
diegoarmandoo@hotmail.com, {adicineia,andrielargollo,jislanesds}@gmail.com

Abstract. There are many difficulties in implementation of interactive applications. The new technologies used in the development process for Digital Television and the meeting of different areas such as audiovisual and information technology with vocabularies and their own styles of work brought new challenges for the workflow. The APAI: Analysis and Design of Interactive Applications *framework* was created to assist the process of developing these applications with a focus on requirements gathering and process flow definition.

Key-words: Digital TV, interactive applications, framework, requirements identification, software process

1. Introdução

Os sistemas produtores de mídia evoluíram ao longo do tempo. As necessidades criadas por essas mudanças são mitigadas pela convergência das tecnologias e suas novas infraestruturas. A TV Digital é uma dessas estruturas de convergência.

O processo de desenvolvimento de aplicações interativas para TV Digital é recente, complexo e com poucas ferramentas que facilitam sua implementação. O *framework* de Análise e Projeto de Aplicações Interativas (APAI) foi criado para auxiliar o processo de desenvolvimento dessas aplicações com foco no levantamento de requisitos e definição do fluxo de atividades.

1.1 Motivação

Existem muitas dificuldades na construção de aplicações interativas para televisão digital. Uma destas dificuldades é a falta de maturidade dos processos de desenvolvimento para este tipo de aplicação. Outra dificuldade é a interdisciplinaridade e heterogeneidade das áreas ciêntificas que compõe o escopo das

aplicações interativas. Os campos de estudo do audiovisual e tecnologia da informação e comunicação formam a base para o desenvolvimento destas aplicações.

O encontro destas diferentes áreas proporciona novos desafios [1] como por exemplo: trabalhar com equipes heterógeneas com vocabulários distintos e estilos de trabalho diferente.

Outro desafio encontrado é a elicitação e especificação dos requisitos de interatividade a partir de um roteiro de produção audiovisual [3].

1.2 Proposta do *Framework* APAI

A estrutura principal utiliza-se de alguns aspectos do processo unificado [5]. Estes aspectos são: **fase, disciplina, tarefa, papel e artefato** que compõe o fluxo de trabalho coerente facilitando as etapas de análise e projeto do processo [5]. Este fluxo é exibido através da notação de modelagem de négocio *Business Process Modeling Notation (BPMN)*, onde podemos identificar em que etapas o processo se inicia e finaliza, podemos também visualizar as atividades que são executadas em sequência e seus respectivos responsáveis identificados pelos pápeis.

Ainda no fluxo conseguimos identificar os artefatos que são necessários para a realização de algumas atividades bem como os artefatos produzidos durante o fluxo proveniente da execução das atividades descritas no *framework* (Fig. 1).

As disciplinas criadas no *framework* APAI são: **criar roteiro, classificação do projeto, escolher linguagem de programação, definir requisitos e criar roteiro interativo**. Essas disciplinas são exectudas durante as fases do processo. Cada disciplina possui atividades específicas. Por exemplo: a disciplina **definir requisitos** é composta das atividades **"Listar requisitos – métodos tradicionais"** e **"listar requisitos – análise do roteiro",** neste caso o papel de **desenvolvedor** é quem executa está disciplina. Esta disciplina tem como entrada os artefatos **roteiro** e **documento de classificação** e tem como saída o artefato **documento de requisitos.**

O APAI possui 7 atividades, são elas: **classificar níveis de interatividades** [3] e [8]**, classificar tipos de interatividade** [3]**, classificar tipo de aplicação interativa** [3], estes fazem parte da disciplina classificação do projet**o**. **Listar requisitos – métodos tradicionais, listar requisitos – análise do roteiro, modelar requisitos funcionais e não funcionais – aspecto**, estes fazem parte de disciplina definir requisitos e por último a atividade de **criar roteiro interativo** que faz parte da disciplina de mesmo nome. O *framework* é constituído de 21 componentes, sendo 3 papéis, 8 artefatos e 11 diagramas. O papel de **roteirista** é responsável pela criação do roteiro inicial. O **diretor de interatividade** é elemento de ligação entre o desenvolvedor e o roteirista, diminuindo a distância entre as duas áreas de conhecimento. O **desenvolvedor** fica responsável pelas disciplinas: escolher a linguagem de programação e definir requisitos.

Os artefatos são: **Documento inicial do projeto, Roteiro, Documento de classificação, Documento de definição da linguagem, Documento de Requisitos, Caso de uso genérico, Glossário e Roteiro interativo**: principal documento de requisito possui o nome do projeto, autor, ações e suas interatividades associadas se existir. Define o tempo inicial, final e total da interatividade, Requisito, Localização

na Tela, Forma de Acesso, Ícone da Interatividade, Tempo de Aparição do Ícone de Interatividade.

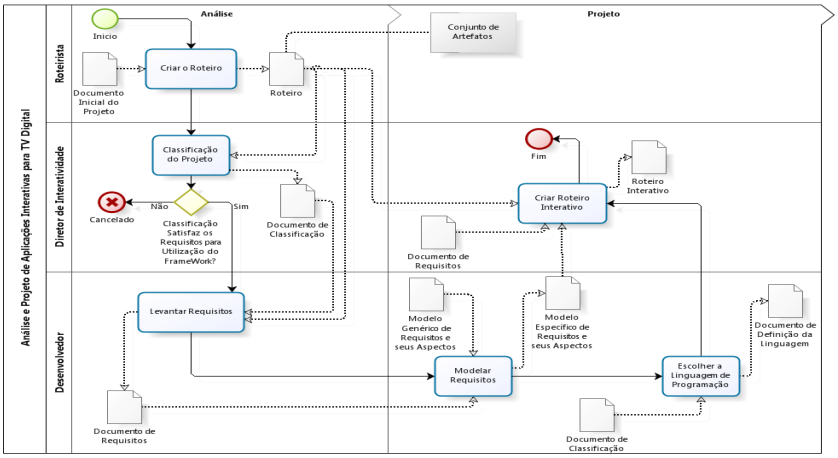


Fig. 1. Diagrama do Processo de Negócio do *Framework* APAI.

Os documentos estão dispostos no diagrama principal do modelo de negócio do *framework* APAI conforme mostra Fig. 1.

2. Objetivos de Pesquisa

O objetivo deste trabalho é a proposição do um *framework* conceitual para análise e projeto de aplicações interativas para TV digital, que façam uso de conteúdo áudio visual ou seja, que utilizem um roteiro como conteúdo principal e as interatividades como complemento.

Tendo em vista que a engenharia de requisitos visa produzir documentos que contribuirão para o projeto de software, este trabalho propõe a utilização de práticas e conceitos já consolidados para construir os documentos de requisito. Um destes conceitos é a definição de disciplinas que orientam a elicitação através de técnicas como entrevista e questionários aplicados aos envolvidos no projeto, e também a elecitação de requisitos através da análise do argumento/sinopse do roteiro (*OutLine*) [7] e análise da idéia do roteiro (*Storyline*) [7].

Este *framework* também propõe um fluxo lógico através de um modelo de negócio a fim de orientar a condução da análise e projeto das aplicações interativas, auxiliar na modelagem dos requisitos não-funcionais usando conceito de orientação a aspectos e definir a linguagem de programação a ser utiliza no desenvolvimento.

3. Contribuições Científicas

As necessidades de levantar requisitos para aplicações interativas surgiu da união de 3 áreas distintas: **TV Digital** e suas novas tecnologias, **interatividade** e suas inovações e o **desenvolvimento de softwar**e. Para suprir essas necessidades alguns conceitos e boas práticas foram compilados em um *framework* conceitual. Estes conceitos e boas práticas são: alguns aspectos do **processo unificado**, boas práticas da **metodologia ágil** e **modelagem de requisitos orientada a aspectos** (Fig. 2).

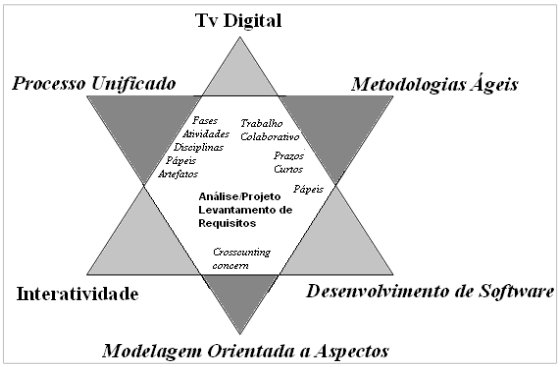


Fig. 2. Proposta e contribuição do *Framework* APAI.

Algumas das contribuições proporcionadas pelo *framework* de Análise e Projeto de Aplicações Interativas (APAI) são: o uso de alguns aspectos do processo unificado que facilitam a visualização do processo de forma geral, definindo responsabilidades e o fluxo lógico de execução. Na área de engenharia de requisitos podemos destacar a definição de uma atividade que auxilia na elicitação de requisitos para TV Digital através da ánalise do *storyline* [7] e *outline* [7] do roteiro, está é uma colaboração muito relevante pois ainda não existem técnicas definidas para levantamento de requisitos de aplicações interativas de audiovisual. Outra contribuição ainda no campo da engenharia de requisitos é a modelagem dos requisitos não funcionais atrelados aos requisitos funcionais elicitados. Por Exemplo: os requisitos não funcionais: **segurança, usabilidade e eficiência** são tratados com a modelagem orientada a aspectos criando um caso de uso de cada requisito funcional listado, em função dos aspectos de requisitos não funcionais (Fig. 3).

Para auxiliar no entendimento de como os requisitos não funcionais agem sobre os requisitos funcionais, utilizamos uma identificação para mostrar como o aspecto afeta o requisito funcional, esta definição foi retirada da abordagem Aspects Oriented Requirement Engineering (AORE) [4], [6]. Essas definições são: **Overlap** é quando o aspecto é aplicado antes ou depois do requisito. **Wrap** é quando o aspecto encapsula o requisito, ou seja, comportamento descrito pelo requisito está envolvido pelo comportamento descrito pelo aspecto. A Fig. 4 exemplifica como os requisitos funcionais RF#1, RF#2, e RF#3 e seus métodos, são afetado pela modelagem orientada a aspectos dos requisitos não funcionais RNF#1, RNF#2 e RNF#3.

No campo da metodologia ágil a prática de priorizar indivíduos e iterações, proporciona satisfação e aumento da motivação e efetividade das equipes multidisciplinares. Espera-se com o uso do framework obter vantagens como: reusabilidade, extensibilidade, abstração das soluções, eficiência na resolução de problemas específicos e otimização dos recursos utilizados [2]. O *framework* auxilia na preparação da etapa inicial, fornecendo os requisitos já definidos, locais onde serão exibidos os requisitos e em que parte do roteiro se pode prospectar os requisitos não funcionais, tempo de aparição na tela, formas de acesso e casos de uso com aspectos de requisitos não funcionais já definidos, pronto para serem implementados seguindo os caminhos traçados.

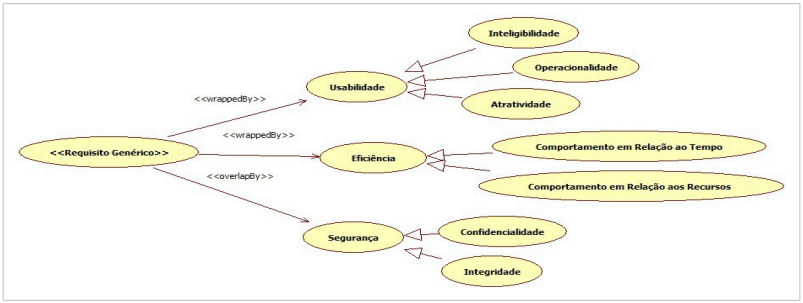


Fig. 3. Diagrama do Modelo Genérico de Requisitos.

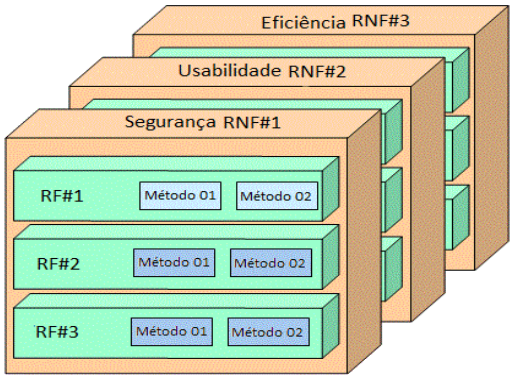


Fig. 4. Exemplo da orientação a aspecto em relação ao requisitos funcionais e não funcionais.

4. Conclusões

Este trabalho abordou o desenvolvimento de um *framework* conceitual de desenvolvimento de aplicações interativas que ajuda na captação e modelagem de requisitos funcionais e não funcionais.

O *framework* proporciona um fluxo coerente definido por um modelo de negócio, diagramado em linguagem BPMN, indicando as principais disciplinas e atividades, e seus respectivos papéis e artefatos. Algumas limitações são encontradas no trabalho proposto. São elas: o fato de apenas poder ser utilizado para determinados tipos de aplicação interativa, abrange somente as etapas de análise e projeto, apenas 3 papéis são definidos e por fim indica quais as melhores formas de se resolver os requisitos não funcionais, porém não propõe padrões para uma solução unificada.

5. Trabalhos Futuros e em Andamento

Como trabalhos futuros podemos listar as seguintes sugestões: Expandir o *framework* para todas as etapas do processo de desenvolvimento de aplicações. Criar os papéis de Designer, Cinegrafista, Testador e definir suas respectivas responsabilidades. Sugerir padrões de projeto para solucionar os requisitos não funcionais. Aplicação em um Estudo de Caso. Integrar o *framework* ao Model-driven Architecture (MDA).

Existem alguns trabalhos em andamento como: uma proposta para aliar desenvolvimento de aplicações interativas em conjunto com metodologias de produção de conteúdo [9], uma proposta utilizando metodologias ágeis para suprir as necessidades do desenvolvimento e foco em ferramentas de autoria para produção de conteúdo digital [10].

Referências

1. Crocomo, Fernando. TV Digital e Produção Interativa. Florianópolis (2007)
2. Meneses, Diego A. O. APAI: Uma Proposta de *Framework* para Análise e Projeto de Aplicações Interativas para TV Digital. 2011. Trabalho de Conclusão de Curso – Curso de Sistemas de Informação, Departamento de Computação, Universidade Federal de Sergipe, São Cristovão (2011)
3. Montez, Carlos; Becker, Valdecir. TV Digital Interativa: conceitos, desafios e perspectives para o Brasil. Florianópolis (2005)
4. Araujo, J. Aspect-Oriented Requirements with UML. Workshop on Aspect-Oriented Modelling with UML. Enschede, The Netherlands (2002)
5. SCOTT, K. O Processo Unificado Explicado. Ed. Bookman, (2003)
6. BRITO, A.MOREIRA e J. ARAÚJO, A Requirements Model to Quality Attributes, Workshop on Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design, 1st International Conference on Aspect-Oriented Software Development, Holanda (2002)
7. PUCCINI, Sérgio. DOCUMENTÁRIO E ROTEIRO DE CINEMA: da pré-produção à pós-produção, pp. 90--91. São Paulo (2007)
8. REISMAN, Richard R., Rethinking Interactive TV. (2002)

9. Souza, Marcia; Santos, Adriana; Amaral Sérgio. Infraestrutura Tecnológica e Metodologia de Produção de Conteúdo para TV Digital Interativa – Uma Proposta para Embrapa. (2009)
10. Velga, Elba; Tavares, Tatiana. Um Modelo de Processo para Desenvolvimento de Programas para TV Digital e Interativa baseado em Metodologias Ágeis. (2007)

An Approach for the Elicitation of Usability Requirements in the Development of Web Applications

Luis Rivero[1], Sabrina Marczak[2], and Tayana Conte[1]

[1]Instituto de Computação, Universidade Federal do Amazonas (UFAM)
Manaus, Brazil
[2]Faculdade de Informática, Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Porto Alegre, Brazil
{luisrivero,tayana}@icomp.ufam.edu.br; sabrina.marczak@pucrs.br

Abstract. Non-functional requirements specify software restrictions and how well the software shall perform its functions. In Web applications, usability requirements are the most important type of non-functional requirements because they determine the acceptability of such applications. Nevertheless, usability requirements are usually forgotten or unconsidered, which increases the cost of their development and the cost of fixing usability problems. This paper presents the research agenda for the development of a set of technologies for the elicitation and specification of usability requirements for Web applications. We will propose and develop a set of techniques and a tool support that will allow software engineers to produce descriptive requirements specifications. Such specifications will guide the development team through the implementation of Web applications meeting usability principals. Finally, to verify the feasibility of the techniques and tool; identify improvement opportunities; and safely apply the technologies in the industry, we will evaluate the technologies through empirical studies.

Keywords: Usability; Requirements Specification and Elicitation Technologies; Web Applications

1 Introduction

Requirements elicitation is one of the most important phases of the software development process since it is when the software functional and non-functional requirements are identified and defined. When poorly performed it can lead to software failure or increases in time and cost of the development process. Non-functional requirements play a fundamental role in the software because they specify software restrictions and how well the software shall perform its function. Furthermore, non-functional requirements try to address aspects that affect software quality such as: usability, maintainability, performance, security, and flexibility.

Usability has been pointed as the most important type of non-functional requirement for guaranteeing software quality [1]. Usability is the absence of obstacles that

stop the users from carrying out their tasks in the system with efficiency, effectiveness, and satisfaction [2]. In Web applications, usability requirements are even more important because in this type of applications, the user interface plays a central role and determines their acceptability and success. In this sense, usability requirements for Web application describe needs in the outlined system, such as the ability to prevent errors and reduce mental workload, in order to enhance its quality.

In our previous research [1], we identified that despite the importance of including usability requirements, their evaluation was mostly performed in later stages of the development process (more specifically during the testing and implantation activities). The main reason for this lack of attention to usability requirements is that the technologies (techniques, processes or tools) that have been proposed to address them are usually applied late in the development, where it is too expensive to develop usability requirements or fix usability problems. Moreover, we identified that there were only a few methods that could be used in the elicitation and analysis of usability requirements [3]. It is important to guarantee the usability of the software in its earlier stages since correcting usability problems after writing the source code can be much more costly than if it was done at the beginning.

In this research, we aim to provide a solution for the late identification of usability requirements in Web projects. We are proposing an approach that will integrate several technologies to understand the users' needs and identify usability requirements in the beginning of the development process. First, we will propose a set of techniques for the elicitation of usability requirements that will allow software engineers to produce descriptive requirements specifications such as: textual specifications, mockups specifications and navigational models. Such specifications will guide the development team through the implementation of Web applications meeting usability principals. Then, we will develop a tool support for the identification and validation of usability requirements, and a monitoring system to examine how users perform tasks in low fidelity prototypes in order to identify further usability requirements.

2 Objectives of the Research

The main goal of this research is to propose an approach composed of a set of technologies for software requirements specification to allow the identification of usability requirements of Web applications. By developing usability requirements along with functional requirements, we intend to improve the quality of Web applications in terms of usability.

In this research, we intend to use mockups which are a software model that makes use of images to show how the software would look like once its source code is written [2]. By using mockups along with descriptive textual specifications and navigation models, we intend to identify possible user interaction problems with the software, and to integrate usability requirements in the development of Web applications. We also aim to reduce the cost of integrating Human Computer Interaction Practices to Software Engineering practices by identifying and developing usability requirements early in the development process of Web applications.

There are several software available in the market for the creation of mockups (e.g. Balsamiq Mockups[1], Mockingbird[2] and Pencil Project[3]). These software can be used to design mockups and link them so that the software development team can come up with interactive mockups and wireframes that can be used for sharing with clients, reviewing workflow, and streamlining user experience. Nevertheless, to the best of our knowledge, we are not aware of any prototype creation software that is able to support the identification of usability requirements and the verification of usability attributes in the mockups, which is essential in order to guarantee the quality of Web applications in their early stages [3]. In this context, our secondary goal is to develop a tool support for the elicitation process of usability requirements.

Our tool will provide several functionalities in order to support the use of our approach by software engineers. First, we will support requirements specification techniques that consider usability requirements along with functional requirements. Then, we will allow software engineers to create requirements specifications such as: (a) a specification using mockups with a hand-drawn look to describe the user interface of the Web application and its usability requirements; (b) a textual specification with thorough descriptions of the requirements of the Web application; and (c) navigational models containing the overall design of the interaction among the different pages within the Web application. The tool will also allow software engineers to perform usability inspections so that they can identify usability problems or further usability requirements. Furthermore, the tool will support the validation of these requirements with users. Finally, we will integrate a user monitoring system in order to analyze the real interaction between users. This last functionality will allow software engineers to identify further usability requirements regarding the navigational models in order to improve the navigability, interactivity and design of Web applications.

3 Methodology and Scientific contributions

To achieve and evaluate the previously proposed research goals we will follow a research methodology based on empirical studies (refer to Figure 1). The analysis of the results from the empirical studies will be used to answer research questions evaluating if the proposed technologies present difficulties that could affect their use. We present here the steps of the methodology and our expected contributions.

First, we will perform a secondary study, through a systematic literature review, to understand the current state of the field of Web usability requirements. We will perform this literature review in digital libraries to identify related research on elicitation techniques that considers usability requirements. Therefore, after applying a research string in the digital libraries, we will select papers addressing the specification of usability requirements in Web applications. Each returned paper will be evaluated through inclusion criteria, such as the quality of the journal or conference in which it was published or the degree of description of the proposed technologies for require-

[1] http://www.balsamiq.com/

[2] https://gomockingbird.com/

[3] http://www.evolus.vn/Pencil/Home.html

ments specification and elicitation. Then, the accepted papers according to the inclusion criteria will be fully reviewed, and the identified technologies will be categorized according to the type of document they generate and in which phases of the requirements engineering process they can be applied. The results from the systematic literature review will be used to propose a set of techniques for usability requirements specification that will be used to create requirements specifications considering usability.

We will also develop a tool support in order to assist software engineers using the specification techniques. Also, such tool we will provide questionnaires, checklists, and any other artifacts that might be used during the requirements specification activities. The tool will support five main activities (*Functionalities to be Developed* activity in Figure 1): (a) requirements specification documentation, (b) usability evaluation, (c) requirements validation, (d) user monitoring, and (e) report creation. In the requirements specification documentation phase, we will support the creation of specifications such as descriptive specifications using mockups and navigational models. Then, software engineers will be able to carry out usability inspections using the tool to identify any problems and guaranteeing that the final software product will meet usability requirements. Also, during the requirements validation, the software development team will show the interaction design to the end users of the application enabling the validation of usability requirements and correction of misunderstandings. Afterwards, to understand how users will really use the software, we will allow monitoring their interaction with the mockups. We will automatically check and verify the number of clicks that would be needed in order to perform a task in the Web application and which design elements catch the user attention. Software engineers can analyze this information to make suggestions and improvements in the user interface. Finally, we intend to automate the creation of different types of reports: (a) validation reports: containing correct, unspecified, misunderstood, or incomplete functional or usability requirements; (b) usability inspection reports: containing the identified usability problems and suggested modifications; and (c) user monitoring reports: containing the comparison of the real use of the application with the interaction models, and further improvement opportunities or identified usability requirements.

We intend to carry out three empirical studies aiming at evaluating the feasibility and suitability of the proposed technologies for the elicitation of usability requirements. Initially, we will evaluate the set of elicitation technologies by comparing them with other technologies that have been proposed for similar purposes (*Feasibility Study* activity in Figure 1). Thus, we will perform quantitative analyses and measure the accuracy (percentage of correct usability requirements), effectiveness (percentage of elicited usability requirements), and efficiency (speed when performing the elicitation) of our approach. Also, the qualitative results from observational (*Observational Study* activity in Figure 1) and industrial case (*Industrial Case Study* activity in Figure 1) studies will be used to improve the performance of our approach in order to be used in the software industry. Finally, we will also use the qualitative data obtained from the opinions and behavior from the subjects participating in the studies to evaluate: (a) if the proposed approach meets the needs of software engineers and if they can follow it accordingly; and (b) how much do the proposed technologies interfere in the development of a real Web application.

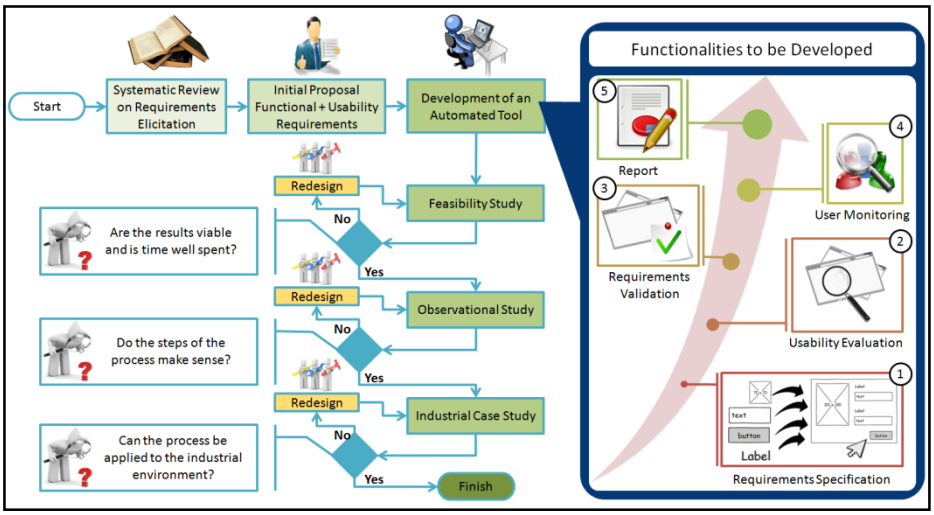


Fig. 1. Research methodology for the elicitation of usability requirements.

4 Conclusions

Our results in [1] and [3] suggested that despite the effort in including or evaluating usability requirements in the development of Web applications, few technologies provide assistance in assuring usability in the beginning of the development. To address this gap, we propose an approach composed of a set of technologies for the inclusion of usability requirements during the design of Web applications mockups.

We will use the research methodology presented in this paper as a plan for the development of the set of proposed technologies for the inclusion of usability requirements. By proposing and supporting requirements specification techniques that consider usability requirements, generating descriptive specifications and models, evaluating the usability in the requirements specification documents, and monitoring user behavior, we intend to improve the usability of Web applications in its early stages, reducing the cost of correcting usability issues later on in the development process.

We hope that the proposed technologies assist software engineers in improving the quality of Web applications in terms of usability, particularly in: (a) diminishing the percentage of dropout in Web applications, (b) diminishing the time of performing tasks, (c) increasing usage and user satisfaction, and (d) diminishing usability errors that prevent users from carrying out their tasks.

5 Ongoing and Future Work

The starting point of this research is the evolution of our previous work [1], the Mockup Design Usability Evaluation (Mockup DUE) tool, which allows the inspec-

tion and validation of Web applications mockups. The Mockup DUE tool currently supports: (a) the inspection of Web applications mockups and (b) the visualization/interaction and validation of mockups with clients.

Figure 2 shows screens of the current version of the Mockup DUE tool. First, we can load and map mockups to simulate interaction (see first half of Figure 2 on the left). Element 1 shows the different functionalities available in this stage: (a) add mockup, (b) add links, and (c) preview. The user uploads image files of mockups which are shown as miniatures (see Element 2). In Element 3 users can view the selected mockups in real scale and add links to connect them. Also, by using the "preview" functionality it is possible to simulate interaction by clicking in the created links. After that, it is possible to use the Mockup DUE tool to identify usability issues (see second half of Figure 2 on the right). Currently the tool allows users to (see Element 4): (a) point errors, (b) add notes, and (c) create reports. In Element 5 the tool shows the supported usability inspection techniques. The inspectors can use the set of verification items or heuristics from the available technique to point usability problems or add notes in the selected mockups (which are shown in Element 6).

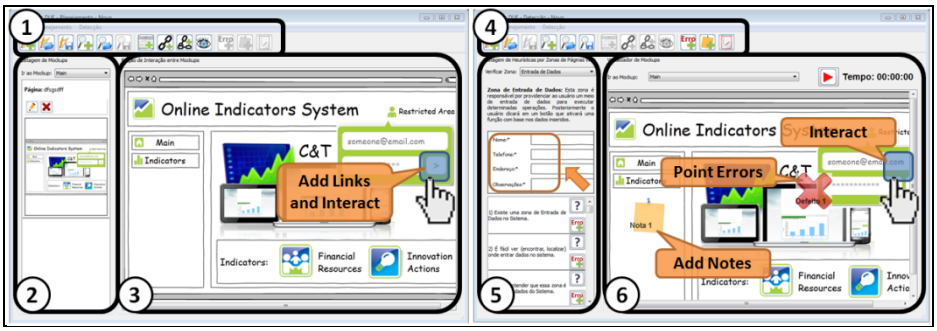


Fig. 2. Current version of the Mockup DUE tool – Mapping mockups and inspection.

We are currently evaluating the Mockup DUE tool in terms of ease of use and user satisfaction. After using the results from this evaluation to improve the tool, we will follow the methodology in Figure 1 to integrate the proposed functionalities that will support the elicitation of usability requirements along with functional requirements.

6 References

1. Rivero, L., Barreto, R., Conte, T.: Characterizing Usability Inspection Methods through the Analysis of a Systematic Mapping Study Extension. Latin-american Center for Informatics Studies Electronic Journal, Volume 16, Issue 1, 2013.
2. Rivero, L., Conte, T.: Using an Empirical Study to Evaluate the Feasibility of a New Usability Inspection Technique for Paper Based Prototypes of Web Applications. Proc. 26th Brazilian Symposium on Software Engineering, Brazil, 2012, pp. 81-90.
3. Rivero, L., Conte, T.: Using the Results from a Systematic Mapping Extension to Define a Usability Inspection Method for Web Applications. Proc. 24th International Conference on Software Engineering and Knowledge Engineering, USA, 2012, pp. 582-587.

Gerenciamento de Requisitos em Scrum baseado em Test Driven Development

Rafael Soares[1], Thiago Cabral[1], Fernanda Alencar[1,2]

[1] Programa de Pós Graduação em Engenharia da Computação, Universidade de Pernambuco, Rua Benfica, 455 – Madalena – Recife/PE, Brasil
{rhas, tclm,fernandaalenc}@ecomp.poli.br
[2] Universidade Federal de Pernambuco, DES-CTG, Av. de Arquitetura, s/n., CDU - Recife/PE, Brasil.
fernanda.ralencar@ufpe.br

Abstract. In order to improve the success rate of software projects various development methodologies have been proposed as alternative to traditional models of development, the agile methodologies. Some steps of the traditional methodology were changed or even canceled, on agile methods, particularly at requirements engineering phase. Traditional methodologies rely on the documentation to manage the project and share knowledge. Meanwhile, the agile methodologies focus on the interaction between those involved in the project to deal with the same goals. There are several challenges with respect to requirements engineering in agile methods. Among these are problems at the process of requirement's elicitation and analyses. This work proposes to analyze tools, techniques, or patterns, like Test Driven Development, in order to better manage the process of requirement's elicitation and validation at Scrum.

Resumo. No intuito de melhorar a taxa de sucesso dos projetos de software várias metodologias foram propostas como alternativa aos modelos tradicionais de desenvolvimento de software, as metodologias ágeis. Nas metodologias ágeis alguns passos foram substituídos ou até mesmos cancelados, sobretudo na fase de engenharia de requisitos. As metodologias tradicionais dependem da documentação para gerenciar o projeto, tanto como para disseminar o conhecimento. Enquanto isso, as metodologias ágeis focam nas interações entre os envolvidos no projeto para lidar com o mesmo propósito. Existem diversos desafios com relação à engenharia de requisitos dentro das metodologias ágeis. Dentre esses, estão problemas no processo de elicitação e análise de novos requisitos. Esse trabalho propõe analisar ferramentas, técnicas ou padrões, como *Test Driven Development*, a fim de melhorar o cenário do processo de elicitação e validação de requisitos no Scrum.

1 Introdução

Tradicionalmente a gerência de requisitos é uma atividade chave no processo da engenharia de requisitos e tem como objetivo principal controlar a evolução dos requisitos, seja por constatação de novas necessidades, seja por constatação de deficiências nos requisitos registrados nos produtos em desenvolvimento. Nesse contexto o rastreamento é fundamental e a documentação primordial.

Nas metodologias tradicionais de desenvolvimento, mantêm-se o foco na geração de documentação sobre o projeto e no cumprimento rígido de processos. Na proposta ágil a ideia é concentrar as atenções no desenvolvimento em si e nas relações entre os participantes [1]. Nos métodos ágeis os requisitos são desenvolvidos de forma incremental, de acordo com as prioridades do cliente.

O Scrum é uma das mais populares metodologias ágeis para o planejamento e gerenciamento de projetos. É especialmente utilizada para projetos de manutenção e desenvolvimento de software. No Scrum os requisitos são tratados de forma incremental, a cada iteração (sprint). Assim, a preocupação com a interação entre os envolvidos no projeto, de forma que os processos de elicitação, validação e manutenção sejam realizados, deve ser maior do que a preocupação em documentar tais processos de fato [1]. Porém, vários são os problemas com relação à engenharia de requisitos no Scrum apontados na literatura e registrados por Jaqueira et al. [11], [5]. Dentre esses, destacam-se os problemas enfrentados durante o processo gerenciamento de requisitos, sobretudo relacionados às elicitação e análise de requisitos. Tais problemas podem levar a requisitos falhos, mal entendidos, ou pouco especificados, podendo impactar negativamente no produto final e na evolução de novos produtos.

Nos dias atuais, a técnica de desenvolvimento orientada a testes (do inglês, Test Driven Development – TDD), proposta por Kent Beck [3], que se baseia em um ciclo curto de repetições, vem ganhando notoriedade entre os projetistas de software. Sobretudo em projetos que usam metodologias ágeis, que como dito, apresentam algumas fragilidades. Em TDD, primeiramente, o desenvolvedor escreve um caso de teste automatizado que define a melhoria desejada ou a nova funcionalidade. A partir daí é produzido um código que pode ser validado pelo teste. Posteriormente, esse código será refatorado em um novo código, mas em padrões aceitáveis.

Com base nisso, buscou-se identificar trabalhos relacionados que tivessem a mesma preocupação com relação às fragilidades dos métodos ágeis, em particular, o Scrum. Da análise inicial, concluiu-se que alguns estudos [12], [13], [14] indicam que a utilização de TDD juntamente com o Scrum traz benefícios ao projeto. Em [12], aponta-se que proporciona melhora no projeto do código e na cobertura de testes das aplicações. Em [13], aponta-se que a utilização da técnica de TDD faz com que os desenvolvedores tenham mais confiança durante a manutenção do código; e, que a produtividade das equipes de desenvolvimento aumenta. Em [14], afirma-se que a utilização de TDD melhora o entendimento dos requisitos.

Considerando os benefícios encontrados com o uso de TDD, e as fragilidades com relação às metodologias ágeis, neste trabalho propõe-se integrar TDD ao Scrum, com vistas a melhorar o processo gerenciamento da elicitação e análise de requisitos. Para

isso, está em curso uma revisão sistemática da literatura, de modo a aprofundar a identificação das abordagens que tratam da mesma temática e quais as questões ainda não trabalhadas.

Esse artigo está estruturado como segue: na seção 2 são apresentados os objetivos do trabalho; as possíveis contribuições da proposta são discutidas na seção 3; algumas conclusões esperadas são apontadas na seção 4; e, por fim, na seção 5 tem-se o andamento da pesquisa e os trabalhos futuros.

2 Objetivos da Pesquisa

O Scrum defende que a interação entre os envolvidos no projeto pode ser uma ferramenta poderosa na elicitação de novos requisitos, tanto como na análise de requisitos. Todavia, as pesquisas [11], [5] mostram que o processo de elicitação e análise de requisitos, no Scrum, encontra alguns desafios, tais como: disponibilidade do cliente; equipe multifuncional, negligência dos requisitos não funcionais, falta de documentação formal, dentre outros. Já os problemas encontrados na fase de análise são: falta de integrantes especializados em análise; falta de documentos formais para análise; dentre outros.

Dessa forma, estamos investigando as vantagens em se propor e utilizar casos de testes, com base no paradigma TDD, bem como um mecanismo de como fazer essa integração. O foco está na melhoraria do processo de gerenciamento da elicitação e análise dos requisitos dentro do Scrum. Tem-se como preocupação, preservar os princípios básicos do Scrum: indivíduos e interações são mais valorizados do que processos e ferramentas; software funcional no lugar de documentação extensiva; colaboração com o cliente sobre a negociação de contratos; e, respostas a mudanças em vez de seguir um plano. Pretende-se chegar a um conjunto de boas práticas para a adoção dos casos de testes que possa ser utilizado pela indústria. Para tanto, deve-se pensar em um processo de execução relativamente simples e que possa ser executado com facilidade e agilidade dentro de uma iteração de desenvolvimento, cobrindo o maior número possível de riscos.

Para a automatização dos testes, identificou-se, na literatura e nos ambientes de desenvolvimento na indústria, a existência de algumas ferramentas que ajudam na implementação de casos de teste com métodos ágeis [15]. Dentre essas, em virtude de da familiaridade no seu uso em projetos reais, elegeu-se considerar, inicialmente, a ferramenta JUNIT [10].

A ferramenta JUNIT, além de ajudar a implementar os casos de testes ágeis, em TDD, dentro do ambiente de desenvolvimento, dão indicação, ao desenvolvedor, onde as mudanças podem interferir e impactar no projeto como um todo.

Argumentamos que o uso de casos de testes em TDD juntamente com o JUnit auxilia a equipe de desenvolvimento tanto na hora da realização de mudanças e evolução no código, como na detecção de requisitos falhos. Lembrando-se que tal detecção de falhas, pode ainda levar à descoberta de novos requisitos antes não pensados pela equipe.

3 Contribuições Científicas

Com já explicitado na seção anterior, pretende-se utilizar casos de testes através da técnica TDD, com auxílio da ferramenta JUNIT, visando melhorar o processo de elicitação e análise de requisitos, em projetos de software desenvolvidos com o framework Scrum. Pressupõe-se que a utilização da técnica TDD, com auxílio do JUNIT, pode oferecer uma alternativa mais simples e rápida para o processo de elicitação e análise de novos requisitos dentro do Scrum.

Algumas impressões iniciais já foram coletadas a partir do desenvolvimento de um projeto real, onde se utilizou o Scrum com a integração de casos de testes desenvolvidos com a ferramenta JUNIT.

O projeto teve como objetivo desenvolver um sistema de apoio a decisões gerenciais na área de fiscalização financeira. A equipe de desenvolvimento era composta por 11 pessoas, programadores, testes, analistas e o gerente de projetos. A duração do projeto foi de 3 anos e 8 meses e, a pedido do cliente, a equipe de desenvolvimento utilizou a técnica de TDD com auxílio da ferramenta JUNIT. Essa proposição se deu com o intuito de amenizar os problemas da engenharia de requisitos, já que a metodologia de desenvolvimento utilizada era o Scrum.

No início a equipe de desenvolvimento teve a sensação de que havia uma perda de tempo, pois antes que a implementação fosse iniciada todos os cenários de testes deveriam ser escritos. Porém, constatou-se que a discussão gerada, proveniente da utilização do TDD, entre os envolvidos do projeto, ajudou também na elicitação de novos requisitos e na detecção de requisitos falhos, diminuindo-se o retrabalho.

Pelo fato do JUNIT oferecer uma visão de quais testes estão sendo afetados, em função de alguma alteração no código, houve não apenas um aumento da confiança dos desenvolvedores em realizar tais mudanças, mas também houve uma melhora na gerencia dessas alterações.

Assim, pretende-se constatar com a pesquisa que um dos principais pontos positivos do TDD é justamente estimular o desenvolvedor a escrever casos de testes antes mesmo de escrever o código funcional e que pode ser facilmente integrado ao Scrum.

Espera-se, melhorar o processo de elicitação e análise de novos requisitos no Scrum. No processo de elicitação de novos requisitos, discutir, escrever e executar os casos de testes pode fazer com que novos requisitos, que não foram detectados previamente pela equipe, possam surgir. Pretende-se verificar que a integração de TDD com o JUnit no Scrum possa também melhorar o processo de análise de requisitos falhos.

4 Conclusões

Mesmo que o Scrum esteja em ascensão, no mundo da indústria de software, e que se mostre eficiente em alguns casos, existem muitos problemas, sobretudo com relação à engenharia de requisitos. Gerenciar os requisitos é uma prática, que dentre muitos

benefícios, serve para comunicar a equipe de desenvolvimento qual o comportamento do sistema e serve de auxilio na manutenção do código já existente.

Tendo em vista a importância do gerenciamento dos requisitos dentro de um projeto de software e que essa fase da engenharia de requisitos não está bem abordada no Scrum é proposta a utilização de casos de testes com base na técnica TDD e com o auxílio da ferramenta JUNIT. A experimentação e uso inicial dessa ferramenta permitiram concluir que a detecção dos impactos resultantes das mudanças ou da evolução dos requisitos de um produto de software fica facilitada com o uso de casos de testes.

Objetiva-se melhorar o cenário dos impactos das mudanças de requisitos em um projeto desenvolvido com o Scrum. O desenvolvimento orientado a testes pode se tornar excessivamente trabalhoso devido a dependências entre classes da aplicação, mas é justamente na dependência que se acredita ter a chave para o gerenciamento das mudanças e do descobrimento de novos requisitos.

Para guiar o processo, pretende-se propor um conjunto de boas práticas de forma a integrar esses casos de testes em paralelo com o uso do Scrum, sem infringir os princípios básicos dos métodos ágeis. Isso já começou a ser testado e parece se adequar razoavelmente bem. Espera-se ampliar o universo da pesquisa através de uma revisão sistemática da literatura mais expressiva.

5 Trabalhos Futuros e em Andamento

Esse é um trabalho que inicia uma nova área de pesquisa no grupo de requisitos, onde se investiga encontrar uma solução para a problemática de gerenciamento em métodos ágeis, o Scrum em especifico, e a criação de casos de testes segundo TDD.

Como mencionado, uma revisão sistemática da literatura está sendo executada com o fim de investigar se já existem indícios sobre os benefícios que o Test Driven Development traga para a elicitação e análise de requisitos em Scrum. Após esse levantamento, serão propostas as boas práticas e implantadas em alguns projetos, a fim de se atestar experimentalmente os resultados da integração proposta.

Pretende-se usar o método de pesquisa experimental a fim de assegurar o teste das hipóteses por meio de um experimento controlado, projetado e aplicado em um projeto real de forma a produzir dados necessários. O processo de experimentação pretendido compreenderá: definição da hipótese; concepção do protocolo experimental; definição de um conjunto de regras no qual se enquadra o experimento; a execução do experimento; e, por fim, a análise e interpretação dos resultados.

Pensa-se também a criação de uma ferramenta que possa interagir com o JUNIT no intuito de levantar dados e gerar artefatos baseados nos casos de testes, escritos no JUNIT. Objetiva-se a geração automática de documentação de forma a se ter uma indicação prática e simples, possivelmente através de forma gráfica, da dependência entre requisitos e o impacto das mudanças que possam advir.

6 Referências

1. Cohn, M.: Aplicando Métodos Ágeis com Sucesso: Desenvolvimento de software com Scrum. Boston, MA. (2011)
2. Schwaber, K.: Agile Software Development with SCRUM. New York, NY. (2010)
3. Beck, K.: TDD Desenvolvimento Guiado Por Testes. New York, NY. (2010)
4. Sommerville, I.: Engenharia de software. 8° Ed, São Paulo, Brasil. (2001)
5. Jaqueira, A.: Uso de Modelos i* para Enriquecer Requisitos em Métodos Ágeis. Dissestação UFRN, Natal, RN. (2013)
6. Cohn, M.: User Stories Applied: For Agile Software Development. Boston, MA. (2004)
7. Glenford, J., Wiley, J.: The Art of Software Testing. Jersey City, NJ. (2004)
8. Cohn, M.: Agile Estimating and Planning. Boston, MA. (2005)
9. Shore, J.,Warden, S.:The art of agile development, San Francisco, CA. (2008)
10. Massol, V. ,Husted, T.: JUnit in action. Los Angeles, CA. (2010)
11. A. Jaqueira, E. Andreotti, M. Lucena, E. Aranha: Desafios de Requisitos em Métodos Ágeis: uma revisão sistemática. 3rd Brazilian Workshop on Agile Methods, São Paulo, (2012)
12. M. Siniaalto, P. Abramhamsson.: A Comparative Case Study on the Impact of Test-Driven Development on Program Design and Test Coverage. Oulu, Finland. (2010)
13. D. Janzen, H. Saiedian.: On the Influence of Test-Driven Development on Software Design. Kansas,USA. (2011)
14. F. Ricca, M. Torchiano, M di Pienta, M. Ceccato, P. Tonella.: Using Acceptance Tests as a Support for Clarifying Requirements: a Series of Experiments. Trento, Italy. (2011)
15. Silva, M.; Moreno, A. Automação em Testes Ágeis. Revista de Sistemas e Computação, v. 1, n. 2, p. 139-164, Salvador (2011).

Business Process Configuration with NFRs and Context-Awareness

Emanuel Santos[1], João Pimentel[1], Tarcisio Pereira[1], Karolyne Oliveira[1], and Jaelson Castro[1]

Universidade Federal de Pernambuco, Centro de Informatica, Cidade Universitaria. S/N, 50741-000 Recife, Brazil
{ebs,jhcp,tcp,kmao,jbc}@cin.ufpe.br

Abstract. [**Context**] Business process models are an important source of information for the development of information systems. Good business processes need to be up-to-date and automated to represent the organizational environment. Representing and configuring business processes variability for a specific organization allows the proper execution of processes. In addition, dynamic environment calls for flexible configuration processes that can meet stakeholders' goals. [**Question/Problem**] Even though current approaches allow the representation of variability of business process models, the selection of business variants in a given context remains a challenging issue. [**Main idea**] In this proposal, we advocate the use of Non-Functional Requirements (NFR) and context-awareness information to drive the configuration of process models at run-time. In particular, we evaluate the use of NFRs to describe the stakeholders' preferences. [**Contribution**] We propose a model-driven business process configuration approach that is driven by NFRs and contextual information.

Keywords: Business Process Configuration, Non-Functional Requirements, Context-Awareness

1 Introduction

Business process models are designed to represent organization practices that create value to a business. By modeling the business processes in terms of activities, the stakeholders can analyze, control and specify systems that will achieve their business goals. Despite the existing methodologies and frameworks to support the development of such systems, a new challenge arises when considering dynamic environments. Sometimes the surroundings of an organization may affect the execution of business processes, which in turn may impact vital areas of a business. In this kind of dynamic environment, variations of external factors such as weather, seasonal variation on demand, dependency on the supply chain and so on, may impact the ability of the business process to properly maintain its activity. In this context, business process flexibility is required to allow continuity in the business process even under these dynamic circumstances. For

self-adaptive systems, the adaptation strategy is supported by software systems that automatically evaluate the situation and use some mechanism to change the business process being enacted, in order to fit the current needs.

In order to face the challenge of providing guidance and support for dynamic business processes configuration we proposed the Business Process Variability Configuration with Contexts and NFRs (BVCCoN) approach [1]. The configuration of a business process is driven by various additional models including information about contexts that affect the process and quality attributes that influence the choice of variants. BVCCoN also represents the variability of business process models, expressing alternative ways to perform the business process represented by BPMN models. This variability description is essential to achieve flexibility, albeit not enough to do it automatically at run time.

We also use Non Functional Requirements (NFRs) [2] models to represent stakeholders' preferences over process variants, allowing the identification of the best configuration when a change in the process is required. Moreover, we make use of context-awareness [3] to trigger these changes and to define what are the valid variants in a process model for a given context. The context-awareness is also the mechanism that aligns the process with the environment and allows the identification of requirement for runtime reconfiguration.

This paper is organized as follow: section 2 presents the objectives of our research. In section 3, we detail the scientific contributions. In section 4 we present related works. At the end we present on going and future works.

2 Objectives of the research

The core objective of this research is to provide guidance to the configuration of business process models, to offer a mechanism that take into account multi-criteria during configuration, as well as a way to reduce user's intervention during the configuration. Therefore, we propose the Business process Variability Configuration with Contexts and NFRs (BVCCoN) approach. The BVCCoN is a business process configuration approach that aims to provide support to configuration of business process based on NFRs and contextual information. It is composed by two parts the BVCCoN model and the BVCCoN process. The model describes the information necessary to perform the configuration including the variability description, the NFR model and the contextual information. The BVCCoN process identifies variation points and variants in a business process model. Moreover, a selection mechanism is used to create new business process models (configurations) with the variants that better suit the system's context and its NFRs. An essential step in this approach is the identification and linking of business process variants with NFRs and contextual variables, which will guide the configuration of the business process.

3 Scientific contributions

The goal of BVCCoN model to provide a variability model which can be applicable to business process models expressed in BPMN language, and integrate Non-Functional Requirements and Contextual information with the variability model. In order to formalize the BVCCoN model we described an abstract and concrete syntax that integrates the various models in a same framework. The meta-model uses a BPMN meta-model to represent business process models. We also extend other models to represent perspectives of variability, NFRs model and Context model. These models are designed following the BVCCoN process. The process in BVCCoN approach is presented in Figure 1. It is composed of five main tasks: Elicit Variability, Describe Variability, Analyze Context, Link NFRs and Variants, and Perform Configuration.

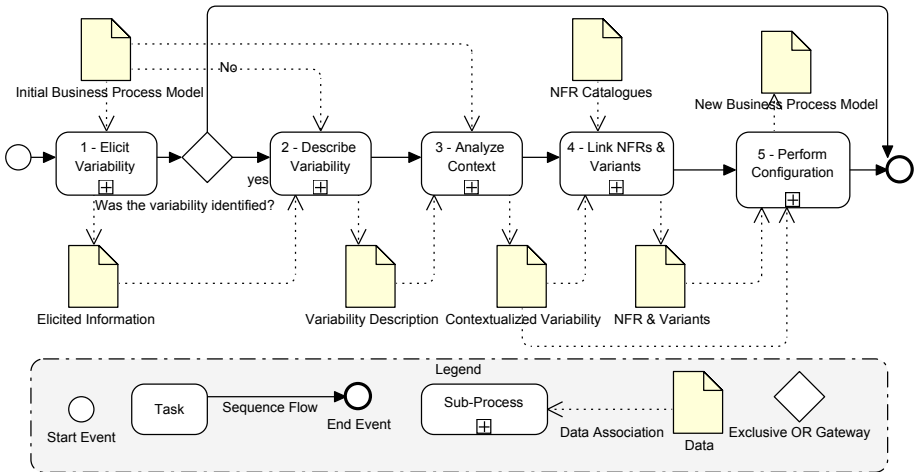


Fig. 1. Process of the BVCCoN approach

The variability elicitation starts by analyzing a reference model. We derived a set of questions to help the identification of process variants. By applying these questions to the model elements we can identify some variations in the way the process is performed. This raw data will be analyzed and then be represented as variants, variation point or as contextual information. It is up to the analyst to decide which data is actually relevant to the process in question. Based on the business analyst knowledge the information is described as process parts using BPMN. The process parts are associated to variants and grouped in variation points.

The process variability is expressed in terms of variants and variation points. The Variation Points indicate where the process can vary, by means of *begins* and *ends* points. The idea is to indicate how and where the main flow (reference

process) can change. The variation points have an operator that indicates how the variants are grouped - they can be AND, OR, or XOR. The variants are associated to business process fragments. The use of process fragments allows the business analyst to describe coherent processes that will still be valid after the process change. To do that it is necessary to assure that the granularity of the process parts are equivalent to the variation point, and that the begins /ends of both match.

After the description of variability, the next step is to associate contexts to process variants. The contexts describe states of the world that can affect the process. In some cases the process may be affected by several factors that can occur within or outside of the control sphere of the organization, such as the availability of resources or changes in weather conditions. We represent the contextual information through logical expressions that can be assessed through data monitoring. The contextual data may be obtained by sensors, by consulting information services, by analyzing profiles, or by user inputs.

We propose the use of NFRs to describe the quality attributes that may be relevant for the stakeholders. In our approach we use the qualitative analysis, where the analyst expresses the way he believes a variant may affect the process configuration in a controlled range: from the most positive (++) to the most negative (−−). Based on this analysis and using prioritization it is possible to select a configuration that is closer to the analyst expectations. Of course the model obtained by this NFR is highly subjective and depends of analyst. Hence, alternative models could have been generated.

The last step of BVCCoN approach generates a new business process model as output. In this step we consider the Variation Points and the Variants of the business process, and how they impact the non-functional requirements. This information can be used to support the configuration itself. It can be performed based on Variants selection for the most critical NFRs. Since we are dealing with runtime adaptability, it may not be possible to rely on experts (e.g., they could not be available). Thus, the definition of priority allows solve potential conflicts at runtime. In our proposal NFRs with higher priority have higher weights. In order to obtain a ranking that takes into account the NFRs contributions, we adopted the Analytic Hierarchy Process (AHP) method which generates a global preference measure based on the choice among alternatives. The contexts defined in our model will be monitored to identify changes in the context variables. Once context changes are detected the selection algorithms run and define the variants that will be part of solution. Using model transformations designed in a specific language (QVTO), we generate new business process models with the variants selected by the algorithms.

Figure 1 presents an example of our proposal. The model represents the check-in and boarding process in an airport domain. The business process in the bottom is a reference process that represents the standard workflows with activities such as *Verify passport or identification*, *Perform Check-in*, *Baggage Drop-off* and so on. The reference process is marked with Variation Points (triangle shape). For example, the VP1 covers three tasks and is associated with the

Perform Check-in, *Perform Check-in manually*, and *Perform On-line Check-in* variants. Observe that variants may be associated to contexts (box shape), which indicates when variants are valid. In the example, the *Perform On-line Check-in* is valid when passengers can check-in for the flight through a website. Moreover, the model also presents a representation of NFRs contribution. The NFRs are represented by clouds and linked to variants through contribution links.

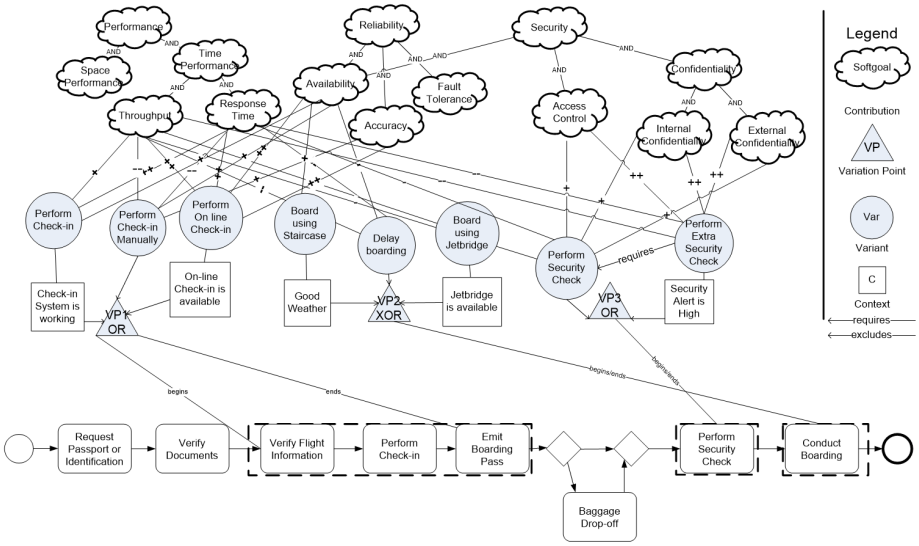


Fig. 2. Example of BVCCoN Model

4 Conclusion

The adoption of NFRs as configuration criteria is one of the key points in the solution proposed by BVCCoN. NFRs describe preferences of stakeholders over variants allowing the selection of the preferred variants in the situation. It is worth mentioning that in BVCCoN we are not dealing with the standard satisfaction notion of NFRs. Since we are dealing with a mutable set of variants, the analysis of NFR satisfaction may not give proper results. In order to address this limitation we applied the Analytic Hierarchy Process (AHP) as part of the selection mechanism. Since AHP is a robust multi-criteria decision analysis method, it can deal with the inconsistencies and still produce a reasonable result.

The second main decision during in BVCCoN design was the use of contextual information to trigger changes in the process models. With that we provide the means to perform process configuration at runtime.

5 Ongoing and future work

Currently, the BVCCoN approach tool support is still under development. We already have concluded some modules such as the BVCCoN modeling tool and the model transformations. Moreover, the algorithms used in the configuration have been implemented as proof of concept and are in process of integration to create the first prototype. However, an integrated environment is still far from the end.

As future work we can highlight: the development of the runtime modules including the context monitor and the integration with a BPMN execution engine, as well as to finish the implementation of tool support. Moreover, we also plan to improve the user interface of our BVCCoN tool.

We may also investigate the use of metrics as contextual information as proposed by [4]. Some study is still necessary to see how to adapt our approach to the use of metrics. One possible solution is to try to include this information during the selection of variants since the AHP supports numerical values. Another solution may be to investigate if it is possible to the represent metrics as context information.

References

1. Santos, E., Pimentel, J., Castro, J., Finkelstein, A.: On the Dynamic Configuration of Business Process Models. In: Enterprise, Business-Process and Information Systems Modeling. Proceedings of the 13th International Workshop, BPMDS 2012, and 17th International Conference, EMMSAD/EuroSymposium 2012, held at CAiSE 2012, Gdansk, Poland, Gdansk, Poland (2012) 331–346
2. Santos, E., Pimentel, J., Castro, J., Sánchez, J., Pastor, O.: Configuring the Variability of Business Process Models Using Non-Functional Requirements. In: Enterprise, Business-Process and Information Systems Modeling. Proceedings of the 11th International Workshop, BPMDS 2010, and 15th International Conference, EMMSAD 2010, held at CAiSE 2010, Hammamet, Tunisia, June 7-8, 2010, Hammamet, Tunisia, Springer Berlin Heidelberg (2010) 274–286
3. Santos, E., Pimentel, J., Dermeval, D., Castro, J., Pastor, O.: Using NFR and Context to Deal with Adaptability in Business Process Models. In: Proceedings of the 2nd International Workshop on Requirements@Run time 2011, Trento, Italy (2011)
4. Oliveira, K., Castro, J., Santos, E., Fidalgo, R., España, S., Pastor, O.: A Multi Level approach to Autonomic Business Process. In: Proceedings of the 26th Brazilian Symposium on Software Engineering (SBES 2012), Natal, Brasil, IEEE Computer Society (2012)

Goals and Scenarios to Software Product Lines: the GS2SPL Approach

Gabriela Guedes, Carla Silva, Jaelson Castro

Centro de Informática – Universidade Federal de Pernambuco (UFPE)
CEP 50740-540, Recife/ PE – Brasil
{ggs, ctlls, jbc }@cin.ufpe.br

Abstract. Goal-oriented requirements engineering (GORE) approaches for Software Product Lines (SPLs) offer a natural way to capture not only stakeholders' goals but also similarities and the variability of an SPL. Goals to Software Product Lines (G2SPL) is an approach that guides the systematic creation of an SPL feature model from i* models with cardinality. However, by using G2SPL it is not possible to specify the behavioral characteristics of an SPL. In order to capture the system's behavior, it is common to use a scenario specification technique. In this paper, we present GS2SPL (Goals and Scenarios to Software Product Lines), an approach for the Requirements Engineering phase of SPL development that combines G2SPL and PLUSS use case scenarios. Our approach also includes a sub-process for configuring specific applications of an SPL based on the priority given to non-functional requirements.

Keywords: Requirements Engineering, Software Product Lines, Goal Models, Feature Model, Scenarios.

1 Introduction

In Requirements Engineering (RE) for Software Product Lines (SPL), feature models are used to capture similarities and the variability of product families. However, according to Silva, Borba and Castro [1] it is a challenge to establish a relationship between features of a software product and stakeholders' goals, since feature models do not capture which stakeholder's need originated each feature.

In this context, some Goal-Oriented Requirements Engineering (GORE) approaches were proposed to model requirements variability in SPL [2-5], these approaches can trace a relationship between features and goals. A comparison of them presented in [6], motivated the definition of the G2SPL (Goals to Software Product Lines) approach [1]. It relies on i*-c (i* with cardinality) language, which is used to (i) structure requirements according to stakeholders' intentions for the SPL, (ii) facilitate the gathering of features that define the SPL and (iii) aid the configuration of an individual product.

However, none of the approaches compared in [6], nor G2SPL captures dynamic or behavioral aspects of the SPL. This could be done using a scenario specification technique. Scenarios describe the behavior of the system functionality and are widely

used in requirements engineering because stakeholders easily understand them [7]. PLUSS (Product Line Use case modeling for Systems and Software engineering) [8] is an SPL approach that combines feature models and use case scenarios. It captures both common and variable behavior of the SPL. In PLUSS, both use cases and scenario steps are annotated with the features to which they are related.

In this paper, we present a requirements engineering approach for SPL that combines goal models, feature models and use case scenarios. The combination of these three models should provide a more complete requirement specification of the SPL, modelling stakeholders' goals, the SPL's functionality and its behavior.

2 Objectives of the Research

The main goal of our study was to define a requirements engineering approach for SPL that integrates i* models, features models and use case scenarios. Moreover, this approach should provide guidelines to derivate feature models and use case scenarios with variability from i* models.

Our goal was achieved by extending G2SPL [1], an approach where the feature model of an SPL is generated from i*-c (i* with cardinality) models. We have added new activities to guide the generation of use case scenarios with variability from i*-c models. We have also added a sub-process for configuring the SPL's artifacts for a specific product. This new approach was called GS2SPL [9] (Goals and Scenarios for Software Product Lines).

3 Scientific Contributions

The GS2SPL process consists of eight activities, most of them are part of the Domain Engineering process and only the last one is part of the Application Engineering process. The first four activities were inherited from G2SPL [6], the rest of the process consists in the addition of new activities or adaptation of G2SPL activities. The GS2SPL process is explained below:

1- Creation of SR (Strategic Rationale) Model: This activity consists of modeling stakeholders' goals using i* framework and it is optional if the SR model is already available. The output of this activity is a SR Model.

2- Identification of Candidate Elements To Be Features: The Domain Engineer identifies the elements of the SR Model that could represent features. According to Silva et al. [6], features are extracted from Tasks and Resources. Therefore, all internal tasks and resources of the actor that represents the SPL should be highlighted, as well as task and resource dependencies connected to this actor.

3- Reengineering the SR Model: in this activity, we add cardinality to the SR model. Cardinality may be added to intentional elements and to means-end relationships in which the root element (*end*) has more than one sub-element (*means*). The output is a SR model with cardinality. Fig. 1 presents part of the SR model with cardinality for Mobile Media [10], an SPL that will be used in this paper as a running example. The main purpose of Mobile Media is to manage media files in mobile devices.

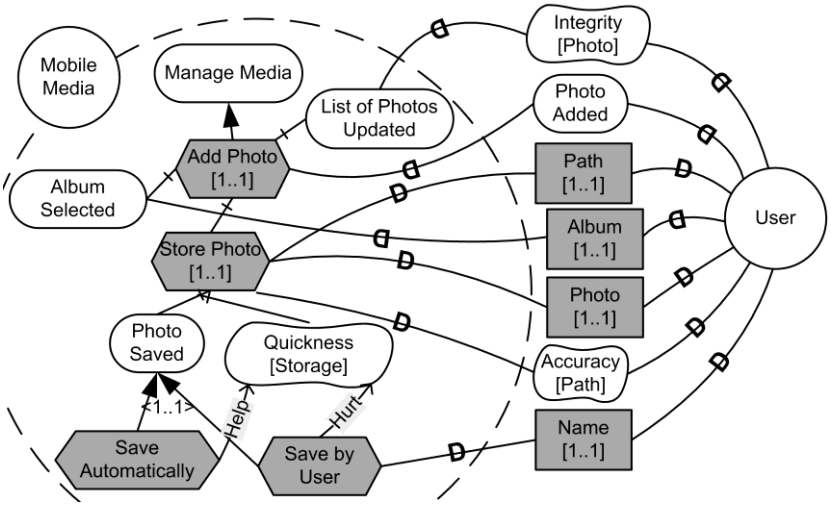


Fig. 1 i*-c model of Mobile Media

4- Elaboration of the Feature Model: This activity is concerned with the derivation of the SPL's feature model, this derivation uses the SR model with cardinality and is guided by the application of some heuristics. According to the heuristics defined in this activity, optional features are obtained from elements with cardinality [0..1], while mandatory features are obtained from elements with cardinality [1..1]. Elements involved in a means-end relationship with cardinality become alternative features with equivalent cardinality. Fig. 2 depicts the FM obtained for Mobile Media.

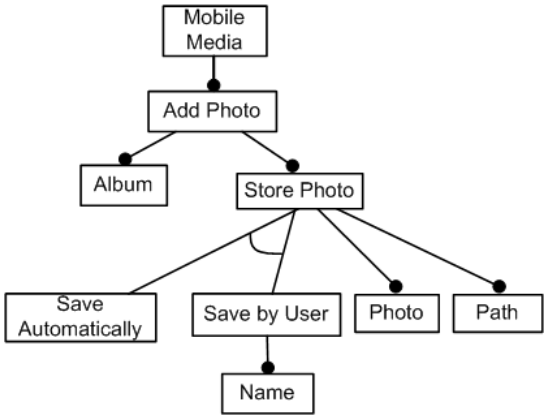


Fig. 2 Feature Model of Mobile Media

5- Feature Model Refinement: This is an optional activity and it is executed if the feature model needs to be reorganized or if there are that were not captured in the SR

model. If the feature model has repeated features, sub-features with more than one parent or different features with the same meaning, reorganization is required. This activity can be performed as many times as the domain engineer believes it is necessary. Our running example is quite simple and did not require the execution of this activity.

6- Elaboration of Use Case Scenarios: The SPL use case scenarios are specified according to an adaptation of the guidelines defined by Castro et al. [11]. This activity uses the SR Model with cardinality and the feature model as input to generate the PLUSS [8] scenarios description of an SPL. The guidelines proposed by Castro et al. in [11] are a mapping between i* models and use case scenarios that are not specific for dealing with SPL variability. We propose guidelines to map i*-c models to PLUSS use case scenarios.

The guidelines are divided in three steps. Step 1 (Discovering Actors) is composed by Guidelines 1 to 5, that determine which i* actors should be mapped to use case actors. Basically, i* actors that have dependencies with the SPL actor should be mapped to use case actors, unless all dependencies between them are softgoal dependencies. In our example, there is only one external actor, "User", and, according to the presented guidelines, it can be mapped to a use case actor.

Step 2 (Discovering use cases for actors) is composed by Guideline 6, that analyzes dependencies between the actor that represents the SPL and those i* actors that were mapped to use case actors in order to determine which dependencies should be mapped to use cases. In summary, we map goal dependencies to use cases; task and resource dependencies are mapped to use cases if they require many steps; and, finally, softgoal dependencies cannot be mapped to use cases, because they represent non-functional requirements. Applying Step 2 to the example, we discovered that only the "Photo Added" goal dependency can be mapped to a use case.

Step 3 (Discovering and Describing Use Case Scenarios) is composed by Guidelines 7 to 12, that guide the elaboration of scenarios descriptions through the analysis of the intentional elements and their relationships inside actors' boundary. In summary, sub-elements of task decomposition links are mapped to primary scenario steps, while sub-elements of means-end links are mapped to alternative steps (creating alternative scenarios). The cardinality of intentional elements must be analyzed to determine if the step derived from the element is mandatory or optional. Applying Step 3 to the Mobile Media example, we obtained the description for the "Add Photo" use case (Table 1).

7- Use Case Scenarios Refinement: Scenarios obtained on the previous activity may be succinct or written on a very high level; it will depend on the level of refinement achieved in the SR model. Hence, we suggest the refinement of scenarios descriptions until they reach the desired level of details.

8- Product Configuration: This is the configuration sub-process and it will be executed every time a new product of the SPL has to be derived. It represents the Application Engineering process of GS2SPL and consists of three activities:

Choice of Specific Configuration: Here the client chooses the goals for the new product. Depending on the client's choices, there may be more than one possible product configuration. In our running example, there are two alternatives: one with "Save Automatically" task (A1) and another with "Save by User" task (A2).

Table 1 "Add Photo" use case scenario

Use Case 1: Add Photo

CHARACTERISTIC INFORMATION

Primary Actor: User
Feature: Add Photo
Scope: MobileMedia
Pre-condition: -
Success Condition: Photo added to album

PRIMARY SCENARIO

ID	User Action	System Response
1	Select "Add Photo" option [Add Photo]	
2	Select album [Album]	
3	Provide path of photo [Path]	
4	Select photo to be added [Photo]	
5	-	Photo is automatically saved [Save Automatically]
5	Choose for photo [Name] [Save by User]	Photo is saved with the chosen name
6	-	List of photos is updated

SECONDARY SCENARIOS

RELATED INFORMATION

Non-functional requirements: Integrity [Photo], Accuracy [Path], Quickness [Storage]

Prioritization of Variants: Alternatives obtained according to the client's choices are ranked based on the priority the client gave to the softgoals (modelled in the SR model). The alternative with the highest priority value represents the most suitable configuration for the client's desires. The function to calculate the priority of each variant will not be presented in this paper due to the lack of space.

Product Artifacts Configuration: First, the configuration model is generated by eliminating, from the FM, all features that are not related to elements in the SR model of the chosen alternative. Then, all cardinality indications must be removed from the SR model, thus the i* model of the product is obtained. Finally, only use cases that are related to selected features will be present on the product's artifacts. Scenario descriptions must also be configured by eliminating the steps that are annotated with features that were not chosen.

4 Conclusions

In this paper we presented GS2SPL (Goals and Scenarios to Software Product Lines), a GORE approach for the requirements engineering phase of SPL development.

GS2SPL guides the creation of an i*-c model for a software product line, which is used to systematically generate the SPL's feature model and then its use case scenarios.

The advantage of using GS2SPL is that the most relevant features and use cases for satisfying the stakeholders' goals are obtained in a systematic way from the i*-c model. GS2SPL also provides a sub-process that guides the configuration of the SPL's requirements models for a specific product, that is based on the softgoals' priority. Unfortunately, our approach does not have a supporting tool yet, making it difficult for its adoption in an industrial context.

5 Ongoing and Future Work

As future work, we plan to: (i) perform an empirical validation of GS2SPL to evaluate its strengths and weaknesses; (ii) develop tool support for our approach; (iii) investigate how to identify feature model constraints from i* models; and (iv) investigate how to take feature interactions into account when generating use case scenarios.

References

1. Silva, C., Borba, C., Castro, J.: A Goal Oriented Approach to Identify and Configure Feature Models for Software Product Lines. In: Proc. of the WER'11, Rio de Janeiro, Brazil (2011)
2. Yu, Y., Leite, J.C.S.P., Lapouchnian, A., Mylopoulos, J.: Configuring features with stakeholder goals. In: Proc. of the ACM SAC'08, Fortaleza, Brazil, pp. 645-649 (2008)
3. Mussbacher, G., Amyot, D., Araújo, J., Moreira, A. Modeling Software Product Lines With AoURN. In: Ws on Early Aspects at AOSD'08, Brussels, Belgium. ACM (2008)
4. Silva, C., Alencar, F., Araújo, J., Moreira, A., Castro, J.: Tailoring an Aspectual Goal-Oriented Approach to Model Features. In: Proc. of the 20th Intl. Conf. on Software Engineering and Knowledge Engineering (SEKE'08), San Francisco Bay, USA (2008)
5. Silva, L., Batista, T., Soares, S., Santos, L.: On the Role of Features and Goals Models in the Development of a Software Product Line. In: Ws on Early Aspects at 9th Annual Aspect-Oriented Software Development Conference (AOSD'10), Rennes, France (2010)
6. Borba, C., Silva, C.: A comparison of goal-oriented approaches to model software product lines variability. In: LNCS, Vol. 5833, pp. 244-253, Springer-Verlag (2009)
7. Maiden, N, Alexander, I.: Scenarios, Stories, Use Cases: Through the Systems Development Life-Cycle. 1 ed., Wiley (2004)
8. Eriksson, M., Börstler, J., Borg, K.: Managing requirements specifications for product lines – an approach and industry case study. In: Journal of Systems and Software, 82(3), 435-447 (2009)
9. Guedes, G., Silva, C., Castro, J., Soares, M., Dermeval, D., Souza, C.: GS2SPL: Goals and Scenarios to Software Product Lines. In: Proc. of the SEKE'12, Redwood City, USA, pp. 651-656 (2012)
10. Figueiredo, E. et al.: Evolving software product lines with aspects: an empirical study on design stability. In: Proc. of the 30th International Conference on Software Software Engineering (ICSE'08), Leipzig, Germany, pp. 261-270 (2008)
11. Castro, J., Alencar, F., Santander, V., Silva, C.: Integration of i* and Object-Oriented Models. In: Yu, E., Giorgini, P., Maiden, N., Mylopoulos, J. (eds). Social Modeling for Requirements Engineering. 1st Ed., MIT Press, pp. 457-483 (2011)

Requirements Engineering with a Perspective of Software Evolution

Anticipating requirements based on organizational change

Marília Guterres Ferreira[1], Julio Cesar Sampaio do Prado Leite[1]

[1]Pontifícia Universidade Católica do Rio de Janeiro, PUC - Rio, Brasil

{mferreira, julio}@inf.puc-rio.br

Abstract. Software Evolution is a fact in industrial life. More than that, maintenance is one of the most expensive phases on software life cycle. Additionally, social and organizational aspects are increasingly gaining greater importance for information systems development. In this context, Organizational Semiotics has been considered a promising tool, providing the framework MEASUR for requirements gathering. In this work, it has been studied how to apply Organizational Semiotics in Requirements Engineering with the purpose of Software Evolution.

Keywords: Requirements Engineering, Software Evolution, Organizational Semiotics, Anticipating Requirements, Organizational Change.

1 Introduction

Software evolution is a fact. Most of the changes in software-based systems are caused by changes in organizations from the need to adapt to a more and more competitive and dynamic environment. Those changes in software are far higher than the initial development costs. If when we develop new software systems, we try to anticipate the ways in which they might change, then the software can be modified easily to implement the new requirements. As most changes in software come from organizational change, to be able to predict how the organization will evolve, we should first define what its current state is and then determine for which state it will go. The objective of this research is to understand the way organizational changes impacts system requirements in order to anticipate requirements that come from software evolution.

The authors are members of the Requirements Engineering Group (ER Group), a group of academic research in Requirements Engineering, and also of the Software Engineering Laboratory of PUC-Rio (LES PUC-Rio), a laboratory of academic research applied to the development of software solutions for industry. Most projects developed in LES deals with software evolution and in some cases the software has to be developed from scratch because the maintenance on the running system is very

difficult and consequently expensive and time-consuming. This research comes from the realization that some of the problems could be avoided if some requirements could be elicited in advance. Observations in the cases have found that the requirements that evolved could be classified in *technological* (related to the evolution of the technology used: programming language, hardware, peripheral systems) or in *organizational* (related to the evolution of the business domain). The system barely can be previously prepared to technological evolution, but when it comes to organizational evolution, it may be not only prepared but also cause and anticipate the changes. The software evolution projects showed a need of requirements engineering more geared to social, political, cultural and ethical aspects.

In this context Organizational Semiotics seems to be a promising theory. Organizational Semiotics is a discipline that studies the use of signs and their effects on social practices. On this subject, there is Stamper's School [1] which proposes a set of methods, MEASUR research program, to the design of information systems based on the social-technical paradigm, considering social, political, cultural and ethical issues involved in understanding the problem in the process of requirements engineering. In Brazil, following this school, Baranauskas and colleagues [2], [3] propose a semioticbased method for stakeholders identification and requirements elicitation. Study cases showed that activities from Organizational Semiotics carried out deal with information not captured by other techniques, involving cultural, behavioral, ethical and political aspects [2], [3]. These points make Semiotics relevant to Requirements Engineering and to the problem addressed in this work.

The purpose of this work is to develop a strategy for Requirements Engineering with a perspective of Software Evolution based on the Organizational Changes. Other recent researches are carried on anticipation of requirements and are been studied and the grounding of this study. Pimentel and colleagues [4], [5] presented foresight techniques to predict requirements for autonomic systems. Rolland and colleagues [6] propose to model changes as a set of gaps between the requirements specification of the current and of the future system. This work aims is to unify the aforementioned researches with a perspective of Organizational Change in order to identify future requirements to make Software Evolution process less arduous. This is an incipient work and its references are its most important related works.

2 Objectives of the Research

The general objective of this research is defining a strategy to anticipate requirements change and consequently make the software evolution less traumatic, less expensive and in less time.

The specific objectives are:

1. Identify organizational concepts influenced by software;
2. Identify organizational concepts that influence software;
3. Define trends of requirements change in a software evolution process;
4. Merge all aforementioned and define a conceptual framework of organizational characteristics most likely to change to be considered in software evolution.

For this, we propose the method showed in Figure 1. This is a preliminary method since it is still on study.

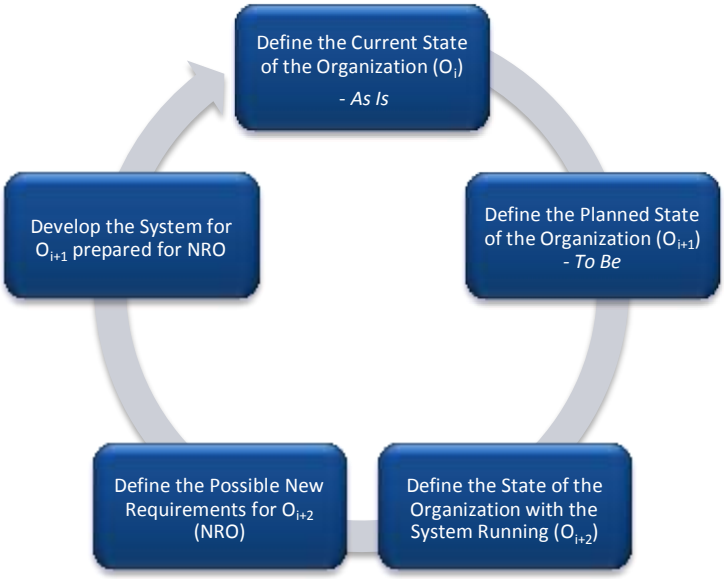


Figure 1. Preliminary Method for Requirements Engineering with a Perspective of Software Evolution

This Preliminary Method is explained as follows:

1. **Define the Current State of the Organization (O_i) –*As Is*:** In this phase, the focus of the requirements engineers is on model the current characteristics of the organization. Here, the problems to be addressed by the software are defined. This step is supported by the method PAM (*Problem Articulation Method*) of Organizational Semiotics.
2. **Define the Planned State of the Organization (O_{i+1}) – *To Be*:** In this phase, the organization is modeled as planned to be. It is time to elicit the goals, the functions and the constraints that the software must to address. This step is supported by the methods SAM (*Semantic Analysis Method*) and NAM (*Norm Analysis Method*) of Organizational Semiotics.
3. **Define the State of the Organization with the System Running (O_{i+2}):** Once the system is running, it implies in some changes in the organization's culture. This phase concentrates on model the changes and differences between **O_i** and **O_{i+1}**. Theories from Organizational Change Management help this step. This step is also supported by the method PAM of Organizational Semiotics.
4. **Define the Possible New Requirements for O_{i+2} (NRO):** With the model **O_{i+2}** in mind, define what would be the new requirements for this organization. This step is also supported by theories from Organizational Change Management and by the methods SAM and NAM of Organizational Semiotics.

5. **Develop the System for O_{i+1} prepared for NRO:** the system to be developed will address the current requirements, elicited in steps 1 and 2, but it will also be prepared for the requirements anticipated by steps 3 and 4 for the evolution process to be less traumatic, less costly and in a shorter time.

Figure 2 shows the disciplines underlying the suggested method and the main points of each step.

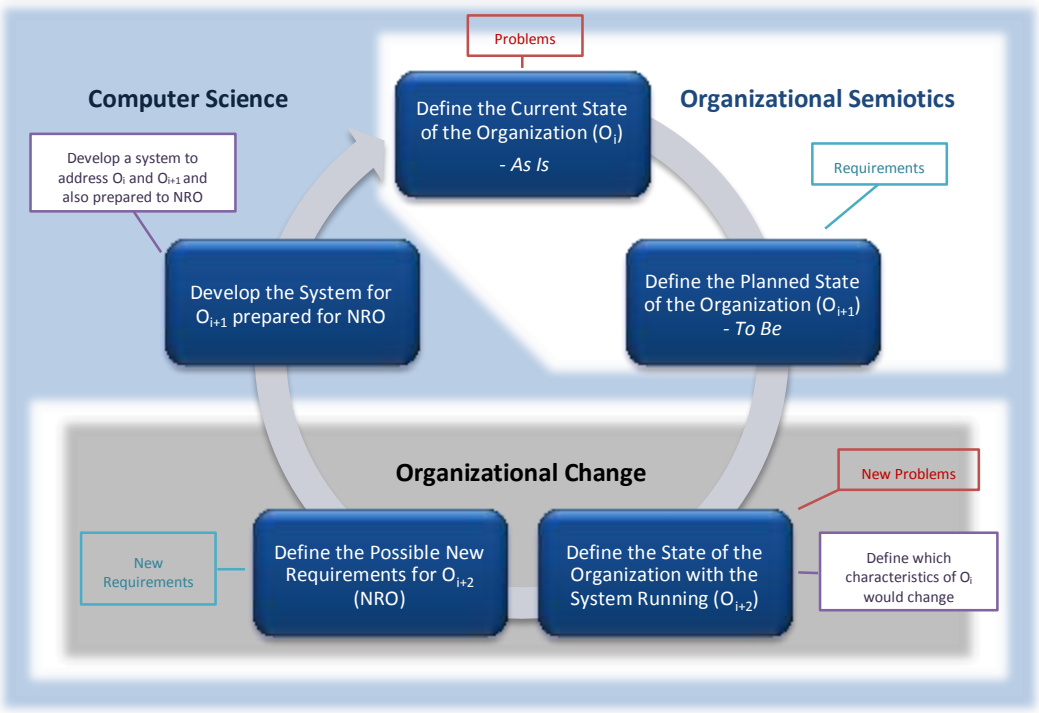


Figure 2. Disciplines that orient the Preliminary Method for Requirements E n-gineering with a Perspective of Software Evolution

3 Scientific contributions

The main contribution of this work is the definition of a strategy to requirements engineering with a perspective of software evolution. This strategy aims to anticipate requirements change based on organizational change. For this, first we define the current state of the organization in two steps, first the problems and then the semantic and norms, based in methods from Organizational Semiotics, PAM, SAM and NAM respectively [1], [2]. And then, based on the changes that the system might cause in the organization, the steps 3 and 4 define the future state of the organization, i.e., the new problems and the new requirements, again supported by Organizational Semiotics. The software-based system to be develop should address the current needs and be prepared for the ones anticipated, thus the software evolution will be less difficult.

4 Conclusions

Software evolution is a recurrent fact in industry and usually under intense time pressure. The anticipation of requirements might support this process. The strategy presented in this study is incremental, once the current model is defined, the next steps determine what is needed to be added, excluded or modified. It is an epistemic tool that aims to help in the design of the problems and requirements of the organization, either in the present and in the future. The expected results are to make the software evolution process less difficult and in a shorter time. It is an ongoing work and further research is required to improve and validate the proposals.

5 Ongoing and future work

Currently, we are investigating real industry cases of software evolution. The preliminary method will be applied in each organization in a *retrospective way*, i. e. looking into the past and into the present, in order to verify if we could have predicted the new requirements with the method. The steps of the analysis to be made will be explained as follows:

1. **Model the past state of the organization:** through conversations with the requirements engineers of the project and research on artifacts and documents, we will elicit the business polices and rules and the problems in the organization that were addressed with the system. It is modeled based on the past, based on PAM. This will be the O_1.
2. **Model the desired stated of the organization:** in this point, we will model what was desired in that moment for the organization to be, what were the requirements. Also modeled based on the past, based on SAM and NAM. This will be the O_2.
3. **Define the current state of the organization:** then, we will model the organization in the present, its new processes, business policies and rules and, mainly, its new problems. We will define how the organization is with the system running. It is modeled based on the present and again in PAM. This will be the O_3.
4. **Define the current requirements for O_3:** now, we will elicit the new requirements for the organization with the system already running. It is modeled based on the present and ever again in SAM and NAM. This will be the NRO.
5. **Analyze the system evolution:** In this point, we will compare the "old" and the current (NRO) requirements and analyze if the NRO requirements could be previously identified if the preliminary method was applied on that moment.

Through the study of the cases, we intend to verify the applicability of the preliminary method with current software evolution projects, once the prediction of requirements may demand long time to be validated. This study will help to better define and improve the method.

Moreover, through the analysis of the nature of change in requirements, we also intend to identify if there is a pattern on requirements evolution, that is, if we could define a framework for requirements evolution. Once the framework is defined, we have to verify if we can point the hotspots on it that should be taking into account in software evolution processes.

As a future work also to be tackled in this thesis research, we pretend to verify if we can also model a conceptual framework of what are the points in an organization that can evolve and at the same time are related to software, points that can either influence the software and be influenced by it. This framework can point out which are the conceptual hotspots that usually evolve and should be considered in any elicitation process. Then, merge the two aforementioned, i. e., make a framework that points at the same time, the organizational characteristics that influences the software and are more likely to change in the process of software evolution. For this we intend to base our researches on theories arising from Organizational Change Management.

References

1. Liu, K.: Semiotics in Information Systems Engineering. Cambridge, England: Cambridge University Press (2000).
2. Baranauskas, M. C., Schimiguel, J., Simoni, C. A., & Medeiros, C. B.: Guiding the Process of Requirements Elicitation with a Semiotic-based Approach – A Case Study. Proceedings of the 11th International Conference on Human-Computer, 3, 100-110 (2005).
3. Simoni, C. C.: A Prática de Desenvolvimento de Software e a Abordagem da Semiótica Organizacional. (Dissertação de Mestrado) Instituto de Computação, Universidade Estadual de Campinas (UNICAMP), Campinas (2003).
4. Pimentel, J., Santos, E., Castro, J., & Franch, X.: Anticipating Requirements ChangesUsing Futurology in Requirements Elicitation. International Journal of Information System Modeling and Design (IJISMD), 3, pp. 89-111 (2012, April).
5. Pimentel, J., Castro, J., Perrelli, H., Santos, E., & Franch, X.: Towards anticipating requirements changes through studies of the future. Fifth International Conference on Research Challenges in Information Science, 1–11. doi:10.1109/RCIS.2011.6006858 (2011)
6. Rolland, C., Salinesi, C., & Etien, A.: Eliciting gaps in requirements change. Requirements Engineering, 9(1), 1–15. doi:10.1007/s00766-003-0168-y (2004).

Uma proposta sobre rastreabilidade de requisitos legais no processo de contratação de soluções de TI na Administração Pública Federal

Lamartine da Silva Barboza[1] , Gilberto A. de A. Cysneiros Filho[2], Ricardo A. C. de Souza[2]

[1] Departamento de Estatística e Informática, Universidade Federal Rural de Pernambuco (UFRPE), Recife, Brasil
lamsilva@gmail.com
[2] Departamento de Estatística e Informática, Universidade Federal Rural de Pernambuco (UFRPE), Recife, Brasil
g.cysneiros@gmail.com, ricardo.andre@deinfo.ufrpe.br

.

Resumo. No âmbito da Administração Pública Federal, o processo de contratação de soluções de TI deve estar de acordo com a Instrução Normativa (IN) nº 04/2010. Essa Instrução Normativa define os requisitos legais (exigências, restrições e orientações) que devem ser atendidos nas diversas etapas desse processo. Nesse contexto, este trabalho apresenta uma proposta de um modelo de referência que aborda um workflow compreendendo os principais subprocessos e artefatos produzidos. Esse modelo de referência tem o objetivo de ajudar na compreensão de todo processo, além de colaborar na proposição de uma abordagem de rastreabilidade entre os requisitos legais e artefatos produzidos, contextualizada a partir de um caso de uso. Do ponto de vista prático, tal abordagem visa auxiliar os gestores públicos e órgãos de controle interno e externo no que se refere ao gerenciamento e fiscalização de todo o processo de contratação de soluções de TI.

Palavras-chave: Rastreabilidade de requisitos, modelos de referência, tecnologia da informação, processo de contratação de soluções de TI.

1 Introdução

A Administração Pública Federal (APF) é um dos maiores contratantes do mercado de soluções de Tecnologia da Informação (TI) no Brasil, movimentando cerca de 12,5 bilhões de reais em 2010, segundo o Tribunal de Contas da União (TCU) [1].

Visando disciplinar o processo de contratação de soluções de TI, a Secretaria de Logística e TI (SLTI) do Ministério do Planejamento, Orçamento e Gestão (MPOG) instituiu a Instrução Normativa (IN) Nº4 de 2010 [2] que deve ser seguida pelos órgãos e entidades integrantes do Sistema de Administração dos Recursos de Informação e Informática (SISP), do qual às Instituições Federais de Ensino Superior (IFES) fazem parte.

Segundo a IN04/2010, solução de Tecnologia da Informação (TI) é "o conjunto de bens e serviços de TI e automação que se integram para o alcance dos resultados pretendidos com a contratação" [2].

A contratação de soluções de TI deve levar em consideração aspectos importantes como a agregação de valor à instituição e gestão de risco, bem como atender às disposições legais e aos princípios básicos da administração pública, como isonomia, economicidade, eficiência, publicidade e legalidade [1].

Na prática, as contratações de soluções de TI devem seguir três fases: (1) planejamento da contratação; (2) seleção do fornecedor; e (3) gerenciamento do contrato [2]. Para auxiliar a execução dessas fases, o TCU publicou o Guia de boas práticas em contratação de soluções de TI [1]. O guia descreve os artefatos que devem ser produzidos e não o processo de contratação como um todo, com ênfase na fase de planejamento da contratação de soluções de TI. Cabendo assim ao gestor público "definir o processo de trabalho de contratação que o órgão seguirá, à luz da legislação existente (e.g. IN 04/2010 [2])". No entanto, problemas podem surgir caso o processo de contratação seguido pelo órgão não esteja "bem definido" ou que não tenha obedecido aos requisitos legais estabelecidos na Normativa. Ocasionando assim inconformidades em auditórias realizadas pelos órgãos de controle.

O processo de contratação de soluções de TI deve atender todas as fases exigidas pela IN04/2010 [2] e possibilitar o rastreamento dos requisitos legais nos artefatos produzidos no processo [3]. Esta última característica é essencial para explicitar a razão e a origem dos itens dos artefatos, bem como para auxiliar o trabalho de controle interno e externo.

Nessa perspectiva, este trabalho apresenta uma proposta de modelo de referência para auxiliar na criação de processos de contratação "bem definidos", bem como procura prover uma compreensão de todo processo. Além disso, busca ajudar nos controles interno e externo por meio de uma proposta de rastreabilidade entre os requisitos legais a serem atendidos e os artefatos produzidos no processo de contratação de soluções de TI.

2 Objetivos da pesquisa

a) elicitar os requisitos legais para contratação de soluções de TI de acordo com a IN04/2010;
b) prover uma abordagem de rastreamento entre os requisitos legais e os artefatos produzidos no processo de contratação de soluções de TI na APF.

3 Contribuições científicas

Elaborar e experimentar um modelo de referência para tratar das relações de rastreabilidade entre os requisitos legais e os artefatos produzidos durante o processo de contratação de soluções de TI.

3.1 Modelos de referência e rastreabilidade de requisitos

O uso de rastreabilidade de requisitos no processo de desenvolvimento de software está relacionado normalmente ao conceito de qualidade. Sommerville [4] relata que um requisito é rastreável se é possível descobrir quem sugeriu o requisito (a fonte), por que o requisito existe (razão), que outros requisitos estão relacionados a ele (dependência entre requisitos) e como o requisito se relaciona com outras informações, tais como desenho do sistema, implementação e documentação do usuário.

Ramesh & Jarke [6] apresentam em estudo os resultados de uma longa pesquisa sobre rastreabilidade de requisitos, com enfoque na visão dos diferente tipos de usuários, os quais foram classificados em usuários sofisticados (*high-end users)* e usuários normais (*low-end users*). Nesse estudo buscou-se também um modelo de referência para implementação dessa rastreabilidade; constatando-se então que os modelos de referência utilizados pelas duas categorias de usuários diferem significativamente.

O meta-modelo desenvolvido no referido estudo possibilitou a captura de informações entre três dimensões: *Source* (fontes), *StakeHolders* (interessados), *Objects* (objetos de produto ou do processo).

3.2 Processo de contratação de TI

Usando o meta-modelo proposto por Ramesh & Jarke [6], podemos, analogamente, estabelecer os relacionamentos (rastreabilidade) das dimensões de um processo específico, diferente do contexto de software. Como por exemplo, no processo de contratação de soluções de TI pela APF. Isto é, relacionar: (i) *Source*: requisitos legais definidos na IN 04/2010 [2]; (ii) *StakeHolders*: pessoas/papeis envolvidos no processo de contratação; e (iii) *Objects*: artefatos produzidos.

Nesse sentido, foi elaborado o modelo de referência (Figura 01, Figura 02, Figura 03), apresentado a seguir, que contempla os principais elementos e Workflow de execução do processo, o qual se baseou no processo de contratação descrito pelo MPOG [5], conhecido como Modelo de Contratação de Soluções de TI (MCTI), tendo como alusão as exigências da Normativa IN 04/2010 [2].

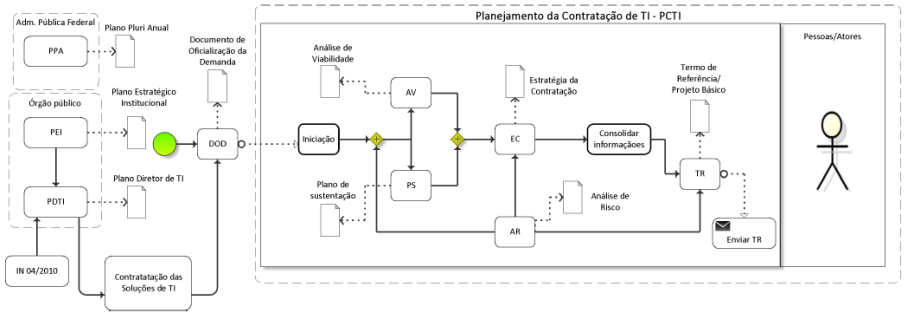


Figura 01: Modelo de referência: fase 01 – adaptação do MPOG[5]

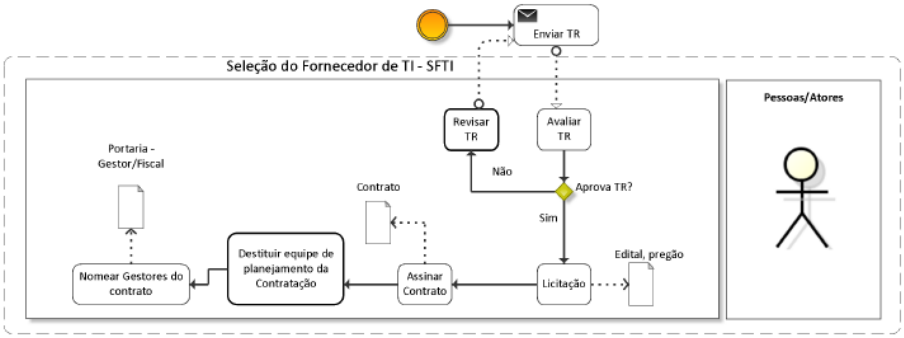


Figura 02: Modelo de referência: fase 02 – adaptação do MPOG[5]

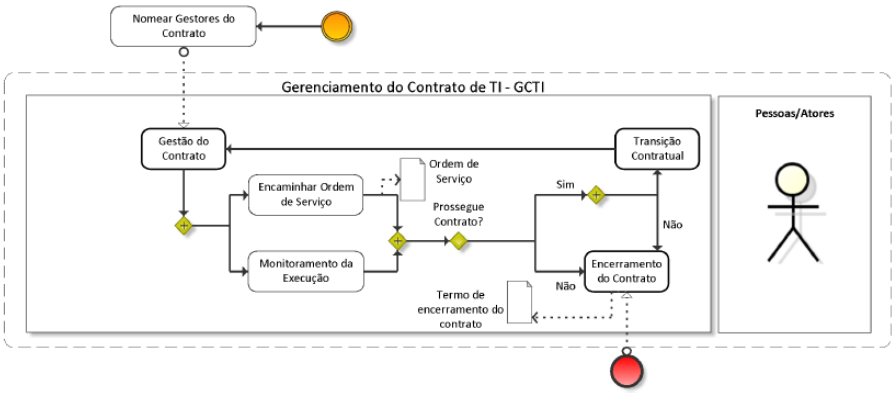


Figura 03: Modelo de referência: fase 03 – adaptação do MPOG[5]

3.3 Caso de uso

Um dos *requisitos legais* extraídos da IN 04/2010 [2], em seu Art. 4º, estabelece que as contratações de TI devem ser precedidas de planejamento, *elaborado em harmonia com o Plano Diretor de Tecnologia da Informação (PDTI).* Neste caso, o desafio é verificar se os *artefatos* produzidos no processo de contratação estão *alinhados com o PDTI* do órgão.

Para experimentar o modelo de referência apresentado vamos avaliar um exemplo real, aplicado ao órgão público Universidade Federal Rural de Pernambuco (UFRPE): suponha uma solicitação formal realizada por um departamento acadêmico sobre aumento e atualização dos computadores do laboratório de informática do departamento. Essa solicitação é registrada no DOD (Documento de Oficialização da Demanda) ponto do qual o processo de contratação é iniciado. No entanto, para se prover a rastreabilidade do requisito legal: “contratações de TI devem ser precedidas de planejamento, *elaborado em harmonia com o Plano Diretor de Tecnologia da Informação (PDTI)”*, indicado na IN 04/2010, é preciso verificar se no documento de planejamento de TI (PDTI) do órgão em questão possui entre seus elementos alguma referência sobre “ampliação e atualização de computadores ...”, na prática, ao se

analisar o PDTI da UFRPE [7], disponível em: http://www.pdti.ufrpe.br/ , percebemos na *seção 7.0 (Plano de Metas e Ações)*, na *subseção 7.1 (Necessidades, Metas e Ações de Infraestrutura)* – o quadro com a necessidade *N07: Ampliar e atualizar o parque computacional nos Laboratórios de Informática e espaços de ensino*, com metas, ações e indicadores dessa necessidade. Portanto, a contratação e implantação dessa solicitação de TI estariam em conformidade com o requisito legal citado, bastando também observar se esse "alinhamento ao PDTI" estaria presente nos artefatos produzidos no referido processo de contratação.

O problema é então pensar em mecanismos para se realizar esse "rastreamento", isto é, da solução de TI contratada, passando por todo processo (em suas três fases) até os requisitos legais elencados na IN 04/2010, conforme ilustra o cenário da Figura 04 a seguir:

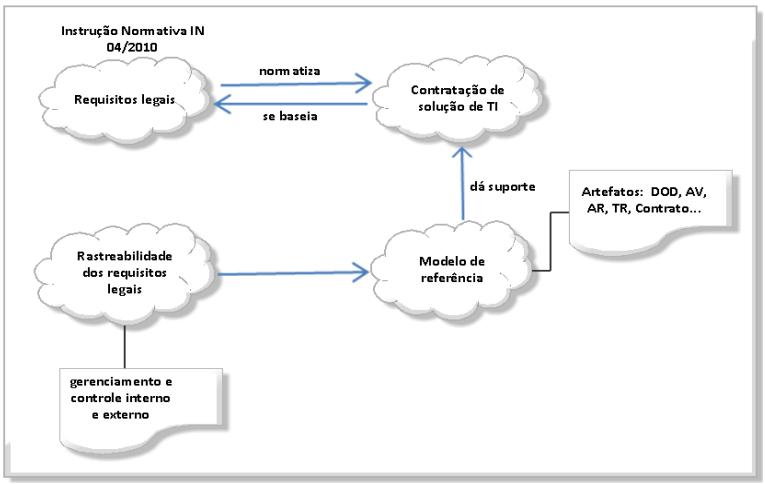


Figura 04: Cenário de rastreabilidade, Fonte: elaborado por Barboza, L.S.

Na prática, o registro desse rastreamento na maioria das vezes não é realizado pelos órgãos públicos em nível de software, ocasionando dificuldades em um processo de auditoria, uma vez que os requisitos legais definidos na Normativa seriam verificados de forma manual, checando documentos da contratação e documentos de planejamento (e.g. PDTI).

Uma proposta inicial de solução é, portanto, realizar o registro desses requisitos legais em cada processo de contratação de solução de TI, por meio de ferramentas já existentes, as quais permitem criar estratégias de rastreabilidade de requisitos no processo de desenvolvimento de software, e assim testar se tal proposta inicial "atende" a rastreabilidade de requisitos legais.

Outra proposta de solução é pensar em uma abordagem automática do registro e mapeamento dos requisitos legais nos diversos artefatos envolvidos na contratação da solução de TI, similar aos estudos desenvolvidos por Caseiros & Cismam [8] que tratam do rastreamento de requisitos de software em sistemas orientados a agentes.

Os trabalhos de investigação no contexto desta pesquisa estão apenas começando, o que pode permitir outras possibilidades de abordagens de solução.

4 Conclusões

Abordagens de rastreamento de requisitos são essenciais ao contexto de processos de negócios, principalmente em casos como o objeto de estudo deste trabalho, pois o processo de contratação de soluções de TI é extenso, possuindo vários relacionamentos e dependências. Nesse sentido, pensar em um modelo de referência que represente uma visão completa do processo, bem como a aplicação de uma técnica e (ou) metodologia que permita verificar e validar se essas contratações estão de acordo com os requisitos legais, nesse caso as normas estabelecidas pela IN 04/2010, facilita a compreensão do contexto e auxilia os controles necessários.

5 Trabalhos futuros e em andamento

A especificação de modelo de referência é só o primeiro passo no trabalho de abordagem de requisitos legais no processo de contratação de TI. O que permite uma experimentação prática inicial. Em um segundo momento é importante realizar o registro dos relacionamentos em ferramentas de software e comparar o resultado em tais ferramentas. Também é extremamente válido pensar em abordagem automática para registro da rastreabilidade, ou seja, verificar por exemplo o "alinhamento" entre IN 04/2010, documentos de planejamentos, processos e artefatos envolvidos, auxiliando assim a verificação e validação "automática" do processo de contratação de TI.

Referências

1. Tribunal de Contas da União. Guia de boas práticas em contratação de soluções de tecnologia da informação: riscos e controles para o planejamento da contratação. -Brasília, Brasil: TCU (2012) 1-527
2. Secretária de Logística e Tecnologia da Informação. Instrução Normativa nº 04 de 12 de novembro de 2010: Processo de contratação de Soluções de Tecnologia da Informação pelos órgãos integrantes do Sistema de Administração dos Recursos de Informação e Informática (SISP) do Poder Executivo Federal. -Brasília, Brasil: MPOG (2010)
3. Winkler, S. and Pilgrim J. V. A.: Survey of traceability in requirements engineering and model-driven development. Software and System Modeling, Vol.9, pp. 529–565 (2010)
4. Sommerville, I. Software Engineering, Addison – Wesley , Reading, MA (1998)
5. Ministério do Planejamento, Orçamento e Gestão. Guia de Boas Práticas para Contratação de Soluções de Tecnologia da Informação. -Brasília, Brasil: MPOG (2011) 1-230
6. Ramesh, B. and Jarke, M.: Towards Reference Models for Requirements Traceability. IEEE Transactions on Software Engineering, Vol.27, pp. 58-93 (2001)
7. Universidade Federal Rural de Pernambuco. Plano Diretor de Tecnologia da Informação 2013-2015. – Recife, Brasil: UFRPE (2013) 1-76
8. Cysneiros, C. and Zisman A.: Traceability for Agent-Oriented Design Models and Code. 19th International Conference on Software Engineering and Knowledge Engineering, USA, (2007)

A bi-directional mapping between i* and BPMN models in the context of business process management

Rebeca Alves, Carla Silva, Jaelson Castro

Centro de Informática, Universidade Federal de Pernambuco - UFPE, Recife, Brazil
{rsa2, ctlls, jbc}@cin.ufpe.br

Abstract. Business Process Management (BPM) involves identifying, modeling, understanding, rethinking and redesigning business processes so that they can achieve the organization's goals. Using a goal oriented approach to model business process, such as i*, can systematically guide the alignment of the business processes with the strategic goals of the organization. In fact, there are more popular techniques to model business process that are focused on capturing the process activities flow, such as BPMN (Business Process Modeling Notation). However, these techniques don't capture the strategic goals of the organization and, therefore, they don't make explicit the alignment (misalignment) between the processes and fulfillment of these goals. We argue in favor of using both models (i* and BPMN) in a complementary way. This paper proposes a model driven approach to obtain BPMN models from i* models and vice-versa. This bi-directional approach can be useful for practitioners of i* and for practitioners of BPMN in the context of BPM.

Keywords: Framework i*, Business Process Management, Goal Orientation.

1 Introduction

Business processes are those that characterize the organization's actions and represent what an organization does in order to achieve a specific purpose or goal. These processes are supported by other internal processes, resulting in a product or a service that is received by an external client. A suitable management of the business processes is important because an organization is as effective and efficient as their business processes. In this context, in business process management (BPM), modeling business process are essential because these models facilitate the understanding and communication about these processes in a organization, as well as, they serve as basis to perform analysis and improvements in the business processes.

BPM has been increasingly used by the industry. In fact, business process modeling techniques focused in capturing process activities flow, such as BPMN (*Business Process Modeling Notation*) [1], are based on concepts such as activities, events, artifacts, control flow, message flow, gateways, among others. However, these techniques lack in capturing the organization's strategic goals and, therefore, they don't make explicit the alignment (misalignment) between the processes and fulfillment of these goals. Moreover, business processes happen in social organization

environments. Organizations are composed of social actors that possess goals and interests that are pursued through a net of relationships with other actors. A richer business process model should include not only how, when and by whom the processes activities are performed, but also the goals of the actors in charge of performing these activities, as well as the dependencies between these actors in order to achieve their goals [2].

Modeling business processes by using a goal oriented approach, such as the i* framework, helps in aligning the business process improvement towards the satisfaction of the organization's strategic goals. i* models capture the organizational level with emphasis on the motivation and intentionality of the actors in the organizational environment. It brings the social analysis that overcomes the usual information and activities flow view and focuses on the goals that the organization wants to achieve [3]. This way, goal oriented models could complement the traditional business process modeling by using flowchart based approaches, such as UML's activity diagram [4] and BPMN. In this paper we argue in favor of using both approaches, goal oriented and flowchart oriented, to model business processes. In particular, we present a model driven approach to obtain BPMN models from i* models and vice-versa, in order to assure that the business processes will be aligned with the strategic organizational goals. This approach is an extension of the proposal presented in [5] in the sense that we refine the existing heuristics for mapping i* models to BPMN models and add new heuristics for this mapping. Moreover, we create heuristics for mapping BPMN models to i* models. The bi-directional mapping between i* and BPMN models aims to encourage the combined use of both modeling techniques in the context of BPM.

This paper is organized as follows. Section 2 describes the research goals and overviews the approaches used in this work. Section 3 presents the contribution of this work applied to a small example to illustrate the mapping from i* model to BPMN model and vice-versa. In section 4, we discuss the obtained results and section 5 presents current and future works.

2 Objectives of the Research

The use of BPM has grown over the years, mainly in the industry. Since business processes should satisfy organizational goals [6], a goal oriented model of business processes can be transformed into a flowchart model of business processes to ensure the fulfillment of the organization's goals [7].

Although the goal oriented approach is not as used as flowchart based approach to model business processes, the integration of both approaches could facilitate the understanding of organizational goals and contribute to obtain a business process aligned with the goals of the organization.

Several methods to transform i* and BPMN models have been proposed as reported and analyzed in [7]. As a result from this analysis, it was discovered practical challenges that need to be addressed for an effective transformation between i* and BPMN models. In particular, the approach proposed in [5] stands out as being one of the methodologies which provides a consistent mapping from i* models to BPMN

models. However, some issues remain still open, such as how to obtain sub-processes in BPMN models from i* models [7].

The objective of this paper is to provide a more complete and systematic set of mapping heuristics to improve the method presented in [5].

3 Scientific Contributions

The contribution of this work is an improvement of the method defined by [5] to obtain BPMN models directly from i* models and vice-versa. The heuristics to obtain a BPMN model from an i* model is described below:

1. In this step, routines are specified and their scope defined. According to [3], a routine is a subgraph of the SR model that represents a particular course among the alternatives. A scope includes the sub-tasks of a routine, dependencies linked to these sub-tasks and actors linked to these dependencies [5]. It will be created a BPMN model for each routine.
2. Each actor present in a scope is transformed into a participant into the BPMN model.
    a. Actors that do not belong to the same organization become different pools.
    b. Actors that do belong to the same organization become different lanes in the same pool.
3. The internal tasks of the actors present in a scope are included as an activity into the lane/pool of the corresponding participant in the BPMN model.
4. If the task within the scope is decomposed, its subtasks must be analyzed:
    a. If the sub-tasks must be performed in parallel, they become parallel activities in the lane/pool of the corresponding actor.
    b. If the sub-tasks must be performed in sequence, they become activities linked through sequence flows.
5. A task dependency is included as an activity in the lane corresponding to the *dependee* actor and a message flow links this activity with the activity present in the lane corresponding to the *depender* actor.
6. A goal become an end event:
    a. If the goal is a dependency, the end event is included in the lane corresponding to the *depender* actor.
    b. If the goal is an internal element of an actor, the end event is included in the lane/pool of the corresponding actor.
7. The root task related to the chosen routine becomes the initial event that triggers the process.
8. A resource dependency becomes an artifact produced by the activity present in the participant representing the *dependee* actor. Two message flows are added between the activities present in the participants mapped from the *depender* and *dependee* actors. These message flows are placed in opposite directions.
9. When the task is decomposed into more than one level, it will be transformed into a sub-process.

The first three heuristics were defined in the original proposal presented in [5]. The last five heuristics are new and were included to make the method more systematic.

The inverse process is applied to the BPMN model to obtain the i* model. An initial version of these heuristics is presented below:

1. Each participant (lane or pool) in the BPMN model is an actor in the i* model.
2. Message flow links become dependencies. When the activity linked by the message flow produces an artifact, the dependency has a resource as *dependum*. The *dependee* actor of this dependency is the actor producing the resource.
3. A non-empty end event may become, depending on the judgment made by the analyst, an internal goal related to the routine being modeled or a dependency goal. In the former case, the goal is internal to the actor that possesses the end event. In the latter case, the *dependee* of the goal dependency is the actor that possesses the end event.
4. A sequence of activities in the BPMN model must be analyzed, and depending on the judgment made by the analyst, they can become sub-tasks of the same decomposed task or tasks without a father and without sons.
5. Softgoals are not modeled in BPMN, but they can be inferred by searching quality attributes associated to the activities performed by the participants.

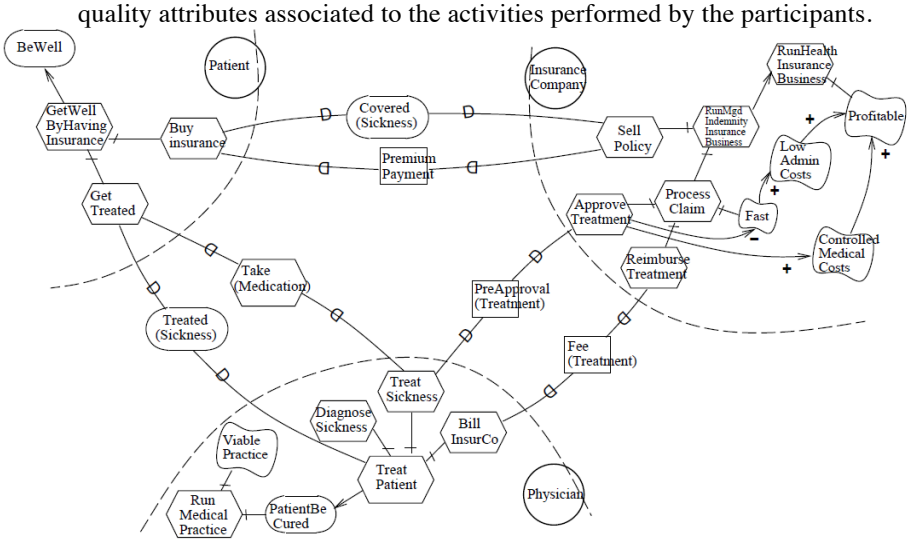


Fig. 1 i* model [3]

3.1 Running Example

The approach was applied to the "Managed Indemnity Insurance" process [3], depicted in Fig. 1. The process is based on management compensation case, where the physician must obtain prior approval to give a treatment to a patient in order to receive payment for the treatment. Fig. 2 presents the BPMN model, obtained after performing the proposed mapping heuristics. We chose the *TreatPatient* routine

present in the Physician actor. Subtasks present in the scope of this routine were included as internal activities in the Physician pool. Patient and Insurance Company actors were also included in the BPMN model as pools, because they are not part of the same organization.

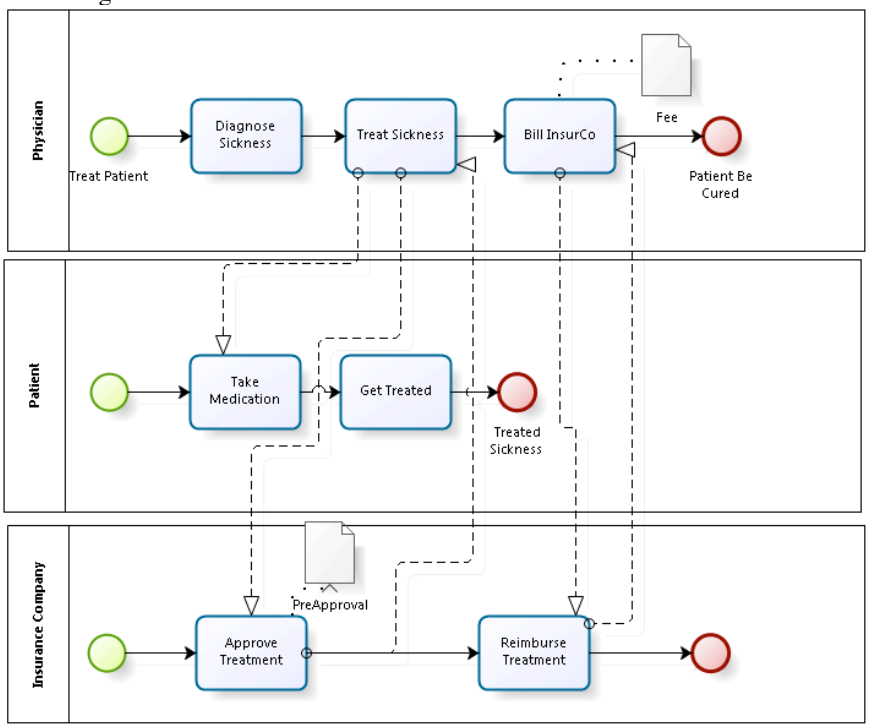


Fig. 2 BPMN model

4 Conclusions

In the previous section, it was presented an initial version of an approach to obtain BPNM models from i* models and vice-versa. The proposed approach aims to make this mapping process more systematic by providing a more complete set of heuristics.

In the example shown, we performed the mapping heuristics and we obtained a BPMN model (Fig. 2) generated from an i* model (Fig. 1) and an i* model (Fig. 3) generated from a BPMN model (Fig. 2). However, the heuristics to perform this last mapping still present some limitations, such as how to obtain goals, softgoals and decomposed tasks, as illustrated by the example.

Although the temporary limitations of our approach, we recognize that obtaining a BPMN model from a goal-oriented model allows understanding the activities of the process within the organization and if these activities are aligned with the strategic goals of the organization. Moreover, using both approaches allows obtaining a richer business process model since BPMN and i* aren't semantically equivalent. In fact,

these approaches capture relevant and complementary business processes information, such as the execution order of the activities (BPMN), goals and softgoals (i*).

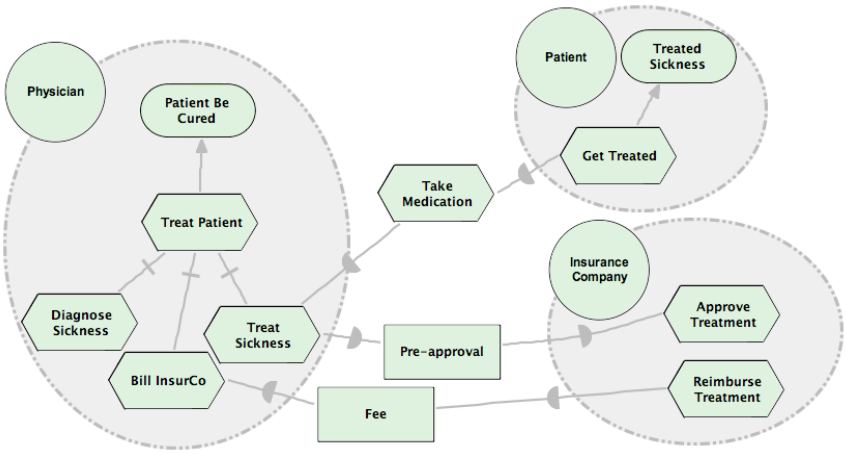


Fig. 3 i* model from BPMN model

5 Ongoing and Future Work

So far, we are working on a more complete set of heuristics to map i* models to BPMN models and vice-versa. We intend to define a more systematic method, reducing human inference during the mapping process, and enabling the automatic generation of one model from another by using model transformations.

References

1. Miers, D. and Stephen, W. BPMN Modeling and Reference Guide. Future Strategies Inc. 2008.
2. Yu, E. and Mylopoulos, J. From E-R to A-R - Modelling Strategic Actor Relationships for Business Process Reengineering. Proc. of the 13th Intl. Conf. on the Entity-Relationship (ER'94) Lecture Notes in Computer Science no. 881, Springer-Verlag. pp. 548-565 1994.
3. Yu, E.: Modeling Strategic Relationships for Process Reengineering. Ph.D. Thesis, Graduate Dept. of Comp. Science, University of Toronto, 1995.
4. Booch, G., J. Rumbaugh and I. Jacobson (1999) The Unified Modeling Language User Guide, Addison Wesley, Reading, MA.
5. Koliadis, G.; Vranesevic, A.; Bhuiyan, M.; Krishna, A. and Ghose, A.: Combining i* and BPMN for Business Process Model Lifecycle Management. In: Business Process Management Workshops, 2006, pp. 416-427.
6. White, S. A.: Introduction to BPMN. IBM Corporation, 2004. Available at: http://www.bptrends.com. Last access: March. 2013.
7. Decreus, K., Snoeck, M. and Poels, G.: Practical Challenges for Methods Transforming i* Goal Models into Business Process Models. In: 17th IEEE International Requirements Engineering Conference (RE'09), Atlanta, GA, USA (2009).

Evoluindo o Catálogo de Transparência: o Estudo do Requisito Não funcional de Entendimento

Priscila Engiel[1], Julio Leite[1]

[1] Programa de Pós-Graduação em Informática – PUC-Rio
R. Marquês de São Vicente, 225 – Gávea, 22451-900, Rio de Janeiro-RJ-Brasil
{pengiel,julio}@inf.puc-rio.br

Resumo. Organizações têm sido cobradas por sua capacidade de fornecer informações sobre seu funcionamento e resultados. Porém tornar-se transparentes não é apenas divulgar informações, é necessário que estas informações sejam tratadas, usadas e compreendidas por seu público alvo. Este artigo descreve uma proposta de evolução de um Catálogo de Transparência, com um estudo mais aprofundado sobre a característica de entendimento. Esta característica foi escolhida para ser detalhada, pois se acredita que para a os sistemas orientados aos cidadãos, é importante que estes entendam a informação que está sendo divulgada

Keywords: NFR-Framework, requisito não funcional, transparência, entendimento

1 Introduction

A demanda por transparência tem se tornado crescente no contexto das organizações, principalmente das públicas. A importância da abertura de informações em organizações públicas é um dos passos necessários para a diminuição da corrupção, mas tendo sido também visualizada como um meio para o estabelecimento de uma sociedade democrática, com cidadãos engajados capazes de acessar e entender as informações disponíveis [1]. Várias leis e acordos têm sido aprovados em busca de garantir a transparência nas organizações. Iniciativas como a Lei Sarbanes-Oxley [2], o acordo da Basiléia - Basel Committee on Banking Supervision [3] e a iniciativa EITI – Extractive Industries Transparency Initiative [4] demonstram a intenção dos governos e das sociedades civis em obter esta transparência. Em setembro de 2011 foi lançada a Parceria para Governo Aberto (Open Government Partnership – OGP), uma iniciativa internacional que pretende difundir e incentivar globalmente práticas governamentais como transparência orçamentária, acesso público à informação e participação social.

No Brasil, existem leis explicitando as intenções do governo quanto à questão de transparência. A Lei da Transparência [5] e a Lei do Acesso [6] obrigam à disponibilização de informações por parte das organizações públicas. Outros exemplos de normatização para transparência são a Carta de Serviços ao Cidadão [7], que tornou obrigatória, para órgãos do Poder Executivo Federal que prestam

atendimento direto ao público, a apresentação de informações sobre os serviços prestados e o Decreto 7724 [6] que dispõe sobre os procedimentos para a garantia do acesso a informação e para a classificação das informações quando ao seu sigilo. No sentido de implementar a transparência, as organizações tem investido na criação de software com informações sobre a organização. Um exemplo são as Páginas de Transparência Pública e os Portais de Transparência, que divulgam as despesas realizadas pelos órgãos e entidades da Administração Pública Federal, informando sobre a execução orçamentária, licitações, contratações, convênios, consumo de passagens, diárias etc. Outro exemplo são os Portais de Serviço, que disponibilizam a explicação e os procedimentos para o serviço ser prestado e cada vez mais com possibilidades de prestação do serviço online, como é o caso do Estado de São Paulo (http://www.cidadao.sp.gov.br).

Entretanto, o requisito transparência é mais que a divulgação das informações, dado que transparência se dá fornecendo informação com qualidade e permitindo seu uso, mas, principalmente, fazendo com que a informação disponibilizada seja compreensível [8][9][10].

Apesar de todos estes impulsos e cobranças pela transparência, as organizações precisam de métodos e/ou técnicas que as ajudem na construção de sistemas de software que viabilizem esta implementação, de forma a garantir que a informação seja compreendida. Atualmente não existem métodos para esse objetivo. As organizações que necessitam divulgar suas informações criam estratégias próprias para cumprir esta demanda.

2 Objetivos da Pesquisa

As organizações atualmente só estão preocupadas em divulgar as informações geradas, sem pensar na qualidade necessária para que estas sejam compreendidas pelo cidadão.

Um exemplo é o Portal de Serviços do Estado de São Paulo. Um dos serviços disponibilizados é o serviço de emissão de certidões (http://www.cidadao.sp.gov.br/servico.php?serv=304059), como apresentado na Figura 1. Nesta página podemos observar a utilização de alguns termos que podem ser de difícil compreensão para o cidadão como, por exemplo, "transcrição paleográfica ". Além disso, a informação não está completa sendo direcionada para outro telefone ou email para mais informações, podendo levar ao usuário a um ciclo de diferentes links e sites até alcançar a informação desejada.

O objetivo deste trabalho é aprofundar o conhecimento sobre a característica Entendimento, um requisito não funcional (meta flexível) 1 que contribui positivamente para a transparência. Para isso um processo como o da Figura 2, precisa existir: a organização deve gerar a informação, tratá-la pensando nas características de seu público alvo e por fim, divulgar essa informação tratada para que ela possa ser compreendida e assimilada. No caso do site citado anteriormente, a organização ao

1 Aqui se usa a contextualização de requisitos não-funcionais na proposta de Chung et. al (CHUNG et al 2000)

invés de usar o termo “transcrição paleográfica” por uma linguagem mais próxima do cidadão como por exemplo cópia de textos antigos.

Figure 1- Portal de Serviços de São Paulo

Figure 2 - Divulgação de informações organizacionais inteligíveis

Para realizar a transformação das informações em informações que sejam compreendidas pelo cidadão, primeiramente, será necessário definir o conceito de entendimento e as operacionalizações necessárias para implementá-lo em um determinado contexto. Depois de definido esse conceito, é necessário estabelecer como implementá-lo. Para isso será necessário elaborar uma método para esta implantação.

3 Contribuições Científicas

Como resultado deste trabalho será realizado uma definição sistemática de Entendimento (uma meta-flexível), de modo que software de apoio organizacional possam ser avaliados no sentido de que existe mais transparência, medida por um melhor entendimento. Esta definição estará organizada em um catálogo possibilitando o uso deste conhecimento em diversas organizações e a evolução deste conhecimento.

Através desse projeto, estaremos validando os resultados ate aqui obtidos na representação de catálogos de transparência, contribuindo também com retroalimentação no sentido de aprimora-los, principalmente quanto ao seu Entendimento.

Também será estudo os processos de elicitação focados para a explicitar o RNF (Requisito Não Funcional) Entendimento em Organizações de forma que uma organização seja mais transparente. A intenção é de que esta tecnologia possa ser

utilizada por organizações que desejam disponibilizar suas informações para os cidadãos, por meio de software, de modo que estes compreendam o que está sendo apresentado.

Será realizada também uma validação dos instrumentos propostos, verificando assim a sua usabilidade e potenciais melhorias. A maneira como esta validação será realizada ainda está sendo estudada pelos pesquisadores.

Este trabalho apresenta uma proposta para definir a característica de Entendimento, de modo a tornar as informações disponibilizadas, por meio de software mais transparentes. O trabalho ainda está em seu estágio inicial, havendo apenas a proposta de elaboração do trabalho e uma ideia inicial de como será constituído o catálogo que organiza as características de entendimento.

4 Trabalhos futuros e em andamento

Nosso objetivo de aprofundar o conhecimento sobre a meta-flexível entendimento num contexto de informações organizacionais fundamenta-se na infraestrutura proposta por Chung[11], mas que carece de uma maior detalhamento, não apenas na instanciação, mas também na investigação de até que ponto a infraestrutura de Chung precisa ser adaptada para nosso caso. Isso implica, por exemplo, um estudo dos efeitos (conflitos) da operacionalização de entendimento em outras qualidades da transparência.

Além disso estaremos também estudando como armazenar esse conhecimento de forma mais apropriada e nesse caso estaremos estudando a ideia de padrões de NFR (Requisitos Não Funcionais), como também estaremos investigando possibilidades de automação e métodos experimentais para análise da qualidade do futuro catálogo.

Também será necessário definir uma estratégia para validação do catálogo e do método que serão propostos As pesquisas de definição de SIGs (Softgoal Interdependency Graph) já existentes podem ser uma fonte de apoio para a definição da forma de validação. Os resultados podem ser oriundos de dados coletados através de observações, experimentos, estudos de casos e outras formas de coleta de informações definidas como adequadas para a pesquisa.

Referencias

1. Holzner B, Holzner L (2006) “Transparency in global change: the vanguard of the open society”. University of Pittsburgh Press, Pittsburgh
2. SOX - Sarbanes-Oxley Act of 2002, Pub. L. No. 107-204, 116 Stat. 745 (codifiedas amended in scattered sections of 15 U.S.C.), 2002
3. BASEL - Basel Committee on Banking Supervision - http://www.bis.org/.Acessado em 20/03/20012
4. EITI – Extractive Industries Transparency Initiative - <http://eitransparency.org/>. Acessado em 20/03/2012.
5. LEI 131 – Disponibilização em tempo real de informações - https://www.planalto.gov.br/ccivil_03/Leis/LCP/Lcp131.htm. Acessado em 30/04/2011

6. Lei do Acesso - Regula o acesso a informação - http://www.planalto.gov.br/ccivil_03/_Ato2011-2014/2011/Lei/L12527.htm
7. DECRETO Nº 6.932, Provê a simplificação dos processos de prestação de serviço ao cidadãos <http://www.planalto.gov.br/ccivil_03/_Ato2007-2010/2009/Decreto/D6932.htm> . Acessado em19/02/2012.
8. CAPPELLI, C. Uma Abordagem para Transparência em Processos Organizacionais Utilizando Aspectos. D.S.c Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, 2009.
9. FUNG, A.; GRAHAM, M.; Weil D. Full Disclosure: the Perils and Promise of Transparency. Cambridge University Press, Londres, UK 2007
10. CAPPELLI, C. Leite, J. C. S. P. "Transparência de Processos Organizacionais" In: II Simpósio Internacional de Transparência Nos Negócios,. v. II. pp. 1-13, Niterói, Rio de Janeiro.2008
11. FUNG, A.; GRAHAM, M.; Weil D. Full Disclosure: the Perils and Promise of Transparency. Cambridge University Press, Londres, UK 2007

Usando Modelos Para Apoiar a Especificação e Verificação de Requisitos de Ubiquidade

Leonardo Mota, Jobson Massollar, Guilherme Horta Travassos

Federal University of Rio de Janeiro/COPPE/PESC
Caixa Postal 68.511, CEP 21945-970, Rio de Janeiro - Brasil
{lsm, jobson, ght}@cos.ufrj.br

Abstract. A computação ubíqua é caracterizada como um novo paradigma onde o poder de processamento está disponível de forma onipresente e imperceptível no ambiente do usuário. Explorar este paradigma em projetos de software permite tratar soluções para problemas até então inviáveis devido a suas características gerais de utilização e acesso. Assim, essa pesquisa visa evoluir uma abordagem de apoio à definição e verificação de requisitos de ubiquidade ao definir estruturas (através de um metamodelo chamado *UbiModel*) para guiar a geração da especificação. Nesse contexto foi desenvolvida uma ferramenta que usa *UbiModel* para apoiar a descrição dos requisitos de ubiquidade de forma mais robusta. Considerando que os requisitos são usualmente definidos sob o ponto de vista das demandas e necessidades do usuário, a ferramenta permite a integração das características de ubiquidade com o ponto de vista do comportamento esperado do sistema, propiciando a descrição dos requisitos de forma abrangente.

Keywords. Requisitos, Ubiquidade Computacional, Ferramenta CASE.

1 Introdução

Atualmente, sistemas de software são importantes ativos estratégicos para as organizações e fazem parte do dia a dia dos indivíduos de forma cada vez mais intensa, tornando fundamental que o seu funcionamento esteja de acordo com os requisitos estabelecidos. Nesse contexto, torna-se cada vez mais essencial identificar, compreender, documentar e avaliar os requisitos que um sistema em particular vai apoiar [1]. Os requisitos possuem papel central no desenvolvimento de software, pois são base para estimativas, modelagem, projeto, implementação e testes, estando presentes ao longo de todo o ciclo de desenvolvimento do sistema. Dada a importância dos requisitos, atividades de controle da qualidade devem ser realizadas para verificar, validar e garantir a qualidade dos requisitos. De acordo com [2], uma das principais formas de minimizar impactos negativos em fases avançadas do desenvolvimento de software é através da redução da inserção de defeitos nas fases iniciais do projeto e criação de mecanismos que possam identificar os defeitos nas fases em que são inseridos a fim de não permitir sua propagação para fases posteriores do desenvolvimento, o que

aumentaria o custo de correção, retrabalho e os riscos de falha do software. Ou seja, quanto mais cedo os defeitos são corrigidos ou minimizados, menores serão os custos de sua correção.

Na construção de softwares ubíquos existem características específicas que normalmente não são consideradas pelas abordagens de desenvolvimento de software tradicionais disponíveis no corpo de conhecimento da engenharia de software, tais como sensibilidade ao contexto e heterogeneidade de dispositivos. Desta forma, ao utilizar estas tecnologias no desenvolvimento de software ubíquo em diferentes domínios de problema, é possível que sua eficiência e/ou eficácia não atinjam o desempenho esperado [3]. Por isso, observa-se a importância de entender como características de ubiquidade podem influenciar no desenvolvimento do software e como podem ser consideradas quando aplicando as abordagens correntes de apoio ao desenvolvimento. Atualmente, ainda é possível observar a existência de poucas investigações sobre como a engenharia de software pode apoiar o desenvolvimento de software ubíquo, o que aumenta a dificuldade e os riscos associados ao desenvolvimento desta categoria de software [4].

A partir desse cenário, Spínola [4] apresenta uma abordagem para apoiar a definição de requisitos de ubiquidade em conjunto com preocupações relacionadas à garantia da qualidade da especificação. Esta abordagem foi elaborada a partir de um conjunto de 6 características e 92 fatores da computação ubíqua [5], com a finalidade de se obter uma especificação de software mais robusta ao considerar as especificidades desse domínio frente a ubiquidade computacional.

2 Objetivos da pesquisa

O trabalho de Spínola [4] apresenta um conjunto baseado em evidência de características e fatores de ubiquidade e suas respectivas definições, obtido a partir de revisões sistemáticas da literatura [5] e de pesquisas de opinião realizadas com especialistas nesse domínio. A partir da construção desse corpo de conhecimento, Spínola [4] propôs um *checklist*, denominado *UbiCheck* [6], que tem como objetivo apoiar e guiar através de um conjunto de perguntas a captura e a elaboração da especificação de requisitos de ubiquidade. Estudos experimentais realizados indicaram que *UbiCheck* ajuda o engenheiro de software a direcionar sua atenção para as informações importantes na definição dos requisitos de ubiquidade. Entretanto, por se tratar simplesmente de uma lista de perguntas abertas e não de um modelo formalizado, sua estrutura não permite a exploração de cenários mais proeminentes, principalmente aqueles relacionados à organização automatizada de apoio adequado à especificação e avaliação da qualidade das especificações de requisitos. Além disso, as perguntas presentes no guia são abertas e possuem um nível de abstração muito elevado, o que as torna demasiadamente interpretativas, ou seja, a definição dos requisitos é muito dependente da interpretação e experiência do engenheiro de software.

Nesse contexto, o objetivo dessa pesquisa é aprimorar a abordagem apresentada por Spínola [4] através da definição de um metamodelo, chamado *UbiModel*, que procura organizar as características e fatores de ubiquidade e suas interrelações de forma mais concreta e detalhada, minimizando o caráter interpretativo sobre essas

características e fatores presente na abordagem original. Assim, ao invés de ser guiado por um conjunto de perguntas, o especificador constrói a especificação a partir da instanciação do metamodelo *UbiModel*, ou seja, a partir da um conjunto rígido e bem definido de elementos, seus relacionamentos e definições. Nesse sentido, *UbiModel* se propõe a prover um direcionamento mais concreto acerca da especificação dos requisitos de ubiquidade através da construção de um modelo com os elementos necessários para a especificação desse tipo de requisito. É importante destacar que *UbiModel* foi totalmente definido a partir das características e fatores de ubiquidade apresentados por Spínola [4]. A Figura 1 mostra os elementos de *UbiModel* referentes a característica *Sensibilidade ao Contexto* que representa o comportamento (*behavior*) esperado do sistema frente a um contexto que se apresenta. Um contexto (*context*) é um cenário de utilização do sistema que considera usuários (*user*) participando de alguma atividade (*activity*) em alguma localização (*location*), sob determinadas condições (*condition*) do ambiente.

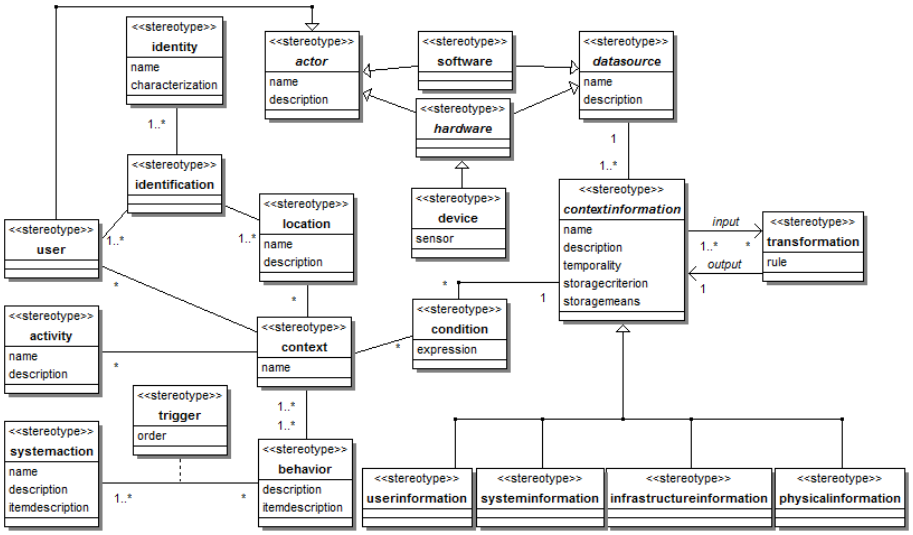


Fig. 1. Visão parcial do *UbiModel*

Outro aspecto explorado na pesquisa está relacionado às perspectivas nas quais os requisitos se apresentam. Os requisitos de ubiquidade descritos com *UbiModel* intencionam representar o ponto de vista do usuário, ou seja, representam suas demandas, necessidades e restrições. Sendo assim, observou-se a possibilidade de tornar a especificação mais abrangente caso fosse possível incluir também a descrição dos requisitos sob a perspectiva do sistema, ou seja, como o sistema irá se comportar a fim de atender às demandas, necessidades e restrições dos usuários. Nesse contexto, a abordagem proposta por Massollar [7] também define um metamodelo, chamado *UCModel*, que apoia a especificação funcional sob a ótica do comportamento esperado do sistema. Assim, o metamodelo *UbiModel* foi ampliado para que os modelos contendo os requisitos de ubiquidade também pudessem referenciar os modelos definidos com

o metamodelo *UCModel*, criando um rastro entre as necessidades/restrições de ubiquidade e o comportamento sistêmico associado à essas necessidades/restrições.

3 Contribuições da Pesquisa

As contribuições desta pesquisa estão relacionadas à definição de uma abordagem para especificação de requisitos de ubiquidade alinhada com os conceitos da computação ubíqua e as características funcionais, visando fornecer uma estrutura que garanta a qualidade da especificação, com expectativa de oferecer apoio à:

- Atualização das descrições das características de ubiquidade de acordo com a interpretação dada na elaboração do metamodelo *UbiModel* em função de novos elementos originados ao detalhar os conceitos relacionados a abordagem proposta em [4];
- Elaboração de um glossário de termos sobre computação ubíqua para apoiar o especificador;
- Estruturação do metamodelo *UbiModel*, que permite a organização da especificação dos requisitos segundo um conjunto de critérios e restrições bem definido e oferece a estruturação a partir do qual é possível tratar questões relacionadas à garantia da qualidade;
- Elaboração de um conjunto de orientações voltadas para a redação das descrições dos requisitos de ubiquidade;
- Obtenção de especificação mais robusta, ao permitir a definição dos requisitos de ubiquidade tanto sob a perspectiva do usuário quanto do sistema.

3.1 Infraestrutura de Apoio à Especificação de Requisitos de Ubiquidade

A partir da definição do metamodelo *UbiModel* foi possível, também, definir uma infraestrutura computacional de apoio à especificação de requisitos de ubiquidade. Como os requisitos estão estruturados em um modelo (*UbiModel*) é possível realizar automaticamente a verificação sintática do mesmo. As principais funcionalidades construídas e oferecidas pela infraestrutura são:

- Caracterização de projetos de software ubíquos de acordo com um modelo de características e fatores de ubiquidade;
- Criação das especificações dos requisitos de acordo com o metamodelo *UbiModel*;
- Associação dos requisitos de ubiquidade definidos com elementos de *UCModel* [7];
- Verificação automática das restrições sintáticas previstas no metamodelo *UbiModel* da especificação do requisito;
- Geração da especificação de requisitos em formato texto (padrão RTF – *Rich Text Format*).

O uso dessa infraestrutura computacional se propõe a reduzir o esforço dedicado à especificação de requisitos de ubiquidade, pois grande parte das atividades são auto-

matizadas. Adicionalmente, a automação proporcionada pode evitar erros durante a execução das tarefas de especificação, visto que o número de interações manuais também é reduzido. A seguir são descritas as ferramentas que fazem parte desta infraestrutura.

UbiProject: tem o objetivo de apoiar a caracterização do projeto de software ubíquo, considerando-se que nem todos os projetos apresentam todas as características de ubiquidade. Utilizar todo o corpo de conhecimento estruturado em [4] para apoiar o desenvolvimento de projetos de software ubíquo sem levar em consideração as restrições e particularidades de cada projeto, pode reduzir a efetividade do uso deste conhecimento. Assim, é importante caracterizar o projeto no que diz respeito às características de ubiquidade, a fim de especializar o apoio à especificação dos requisitos de ubiquidade de acordo com as necessidades do projeto.

UbiSpecification: tem o objetivo de apoiar a especificação de requisitos de ubiquidade propriamente dita usando o metamodelo *UbiModel*, que funciona como um roteiro de especificação ao estruturar os elementos e relações importantes na descrição dos requisitos de ubiquidade. Além disso, permite a associação dos requisitos definidos com elementos de *UCModel* [7], modelo de especificação integrado a *UbiModel* que permite detalhar os requisitos por meio de casos de uso/diagramas de atividade.

UbiDocument: tem o objetivo de gerar um documento estruturado com a especificação de requisitos em formato textual a partir dos modelos gerados na ferramenta *UbiSpecification*.

4 Conclusão

Essa pesquisa tem por objetivo a elaboração de um arcabouço que forneça um contexto mais concreto e robusto para especificação de requisitos de ubiquidade, ao substituir o conjunto de perguntas abertas proposto por Spínola [4] por um metamodelo UML (UbiModel) a partir do qual essas especificações podem ser elaboradas.. Como o *UbiModel* tem o propósito de guiar a elaboração dos requisitos, espera-se que o especificador possa definir requisitos menos ambíguos e de forma menos dependente da experiência e de interpretações individuais do especificador. Além disso, espera-se também que a infraestrutura proposta reduza o esforço necessário para especificação dos requisitos, haja vista que ela oferece apoio para grande parte das atividades previstas. Por fim, também é uma meta do presente trabalho a redução de defeitos inseridos durante o processo de especificação dos requisitos, principalmente aqueles relacionados a omissões e inconsistências, visto que *UbiModel* possui uma estrutura bem definida, capaz de direcionar o especificador para as informações realmente relevantes. Ainda nesse contexto, o *UbiModel* oferece a possibilidade de explorar a verificação automática dos modelos de requisitos, principalmente na questão sintática.

5 Trabalhos Futuros e Em Andamento

Atualmente os modelos estão formalizados, a infraestrutura computacional construída e a pesquisa encontra-se em fase de planejamento de estudo (prova de conceito)

para avaliar a adequação do metamodelo *UbiModel* por meio da utilização da infraestrutura de apoio computacional desenvolvida.

Por fim, a abordagem proposta nesta pesquisa oferece oportunidades de pesquisa no que diz respeito a: (1) definição ou adaptação de técnicas de inspeção, alinhadas aos conceitos definidos na abordagem, e que possam explorar os elementos que a compõem: modelos conceituais, regras, descrições, dentre outras; (2) mapeamento dos requisitos definidos através de *UbiModel* para os possíveis itens de especificação que representam o detalhamento desses requisitos, e; (3) realização de estudos experimentais com o objetivo de verificar a redução de defeitos em documentos de especificação.

6 Referências

1. Aurum, A., Wohlin, C.: Engineering and Managing Software Requirements. Springer-Verlag, New York (2005)
2. Boehm, B. W.M Basili, V.R.: Software Defect Reduction Top 10 List. In: IEEE Computer, vol. 34 pp. 135-137. (2001)
3. Ducatel, K., Bogdanowicz, M., Scapolo, F., Leijten, J., Burgelman, J. C.: Ambient Intelligence: From Vision to Reality. In: IST Advisory Group Draft Report, pp. 45-48 (2003)
4. Spínola, R.O.: Apoio à Especificação e Verificação de Requisitos Funcionais de Ubiquidade em Projetos de Software. Tese de Doutorado. COPPE/UFRJ. Rio de Janeiro (2010)
5. Spínola, R.O., Massollar, J.L., Travassos, G.H.: Towards a Conceptual Framework to Classify Ubiquitous Software Projects. In: Proceedings of the 8th International Conference on Software Engineering and Knowledge Engineering (SEKE), San Francisco (2006)
6. Pinto, F.: Uma Abordagem para Apoiar Especificação de Requisitos para Projetos de Software Ubíquo. Dissertação de Mestrado. COPPE/UFRJ. (2009)
7. Massollar, J.L.: Uma Abordagem para Especificação de Requisitos Dirigida por Modelos Integrada ao Controle da Qualidade de Aplicações Web. Tese de Doutorado. COPPE/UFRJ. Rio de Janeiro (2011)
8. Spínola, R. O., Travassos, G.H.: Arcabouço para Apoiar a Definição e a Garantia de Qualidade de Requisitos de Ubiqüidade em Projetos de Software. In: II Workshop on Pervasive and Ubiquitous Computing. Campo Grande (2008)
9. Spínola, R., Pinto, F.C.R, Travassos, G. H.: Ubicheck: An Approach to Support Requirements Definition in the Ubicomp Domain. In: 25th Symposium On Applied Computing, pp 306-310. Sierre (2008)
10. Massollar, J. L., Mello, R. M. de, Travassos, G. H.: Structuring and Verifying Requirements Specifications through Activity Diagrams to Support the Semi-automated Generation of Functional Test Procedures. In: QUATIC. Lisboa (2012)
11. Massollar, J L., Mello, R. M.de, Travassos, G. H.: Investigating the Feasibility of a Specification and Quality Assessment Approach Suitable for Web Functional Requirements. In: 30th International Conference of the Chilean, pp.108,117, 9-11. Computer Science Society (SCCC), Curico (2011)

Evolução de Requisitos na Metodologia Ágil

Thiago Cabral[1,*], Rafael Soares[1], Fernanda Alencar[1, 2]

[1] Programa de Pós Graduação em Engenharia da Computação, Universidade de Pernambuco, Rua Benfica, 455 – Madalena – Recife/PE, Brasil
{tclm,rhas, fernandaalenc}@ecomp.poli.br
[2] Universidade Federal de Pernambuco, DES-CTG, Av. de Arquitetura, s/n., CDU - Recife/PE, Brasil.
fernanda.ralencar@ufpe.br

Abstract. Nowadays, agile metodologias are widely accepted mainly by industry, by being iterative approaches that keep the dynamics of software development environments. Numerous researches indicate that these methodologies are useful and effective for successful development. However, very little has been about efforts made in the requirements phase in agile methodologies. The evolution of requirements, in particular, brings risks to the software project. However, when well managed can improve software quality and satisfaction of the stakeholders. The evolution of requirements is a themed deserving further examination of these methodologies. In the engineering community requirements, the i * approach is a very rich ontology and has been intensively studied and utilized. Thus, the paper proposes the use of the i * approach to the analysis of the factors necessary for the management of the evolution of requirements in agile methodologies.

Resumo. Atualmente, as metodologias ágeis são amplamente aceitas, principalmente pela indústria, por serem abordagens iterativas que mantêm a dinâmica dos ambientes de desenvolvimento de software. Inúmeras pesquisas apontam que essas metodologias são úteis e eficazes para o sucesso do desenvolvimento. No entanto, muito pouco se tem sobre esforços empreendidos na fase de requisitos nas metodologias ágeis. A evolução de requisitos, em particular, traz riscos ao projeto de software, mas quando bem gerenciada pode melhorar a qualidade do software e a satisfação dos stakeholders. A evolução de requisitos é uma temática que merece ser aprofundada nessas metodologias. Na comunidade de engenharia de requisitos, a abordagem i* é uma ontologia muito rica e tem se mostrado bastante estudada e utilizada. Assim, o artigo propõe o uso da abordagem i* para a análise dos fatores necessários ao gerenciamento da evolução dos requisitos em metodologias ágeis.

1 Introdução

A engenharia de requisitos é o processo de analisar e documentar a funcionalidade pretendida de um sistema de software. Muitas vezes considerada como uma atividade inicial na construção de um sistema [1], acaba por ter amplo ciclo de vida. Dentre as

atividades do processo de engenharia de requisitos merece destaque a evolução dos requisitos ao logo do processo de desenvolvimento do sistema. A maioria das empresas que utilizam os métodos tradicionais de engenharia de requisitos, no tocante à evolução dos requisitos, não sabe lidar com essa questão. Essas empresas se utilizam de estratégias *ad-hoc* para tratar essa questão.

Em se tratando de pequenos projetos de sistemas, a evolução dos requisitos pode não levar a grandes problemas, uma vez que uma pessoa pode coordenar o sistema em sua totalidade. Porém, em grandes projetos, o mau gerenciamento da evolução de requisitos leva a problemas indesejáveis, tais como, retrabalho, atraso para entrega do sistema, baixa qualidade e aumento dos custos do sistema de software. Em [6] é apresentado o estado da arte da evolução de requisitos. Segundo essa pesquisa existe muitos sinônimos para o termo "evolução" usados nas literaturas dentre os cerca de 120 trabalhos identificados com relação à temática.

Outros pontos relacionados a mudanças nos requisitos dizem respeito à volatilidade dos mesmos e às alterações no ambiente organizacional. A volatilidade dos requisitos leva a que funcionalidades descritas na especificação do sistema sejam também instáveis, dificultando a evolução natural do sistema pretendido. As mudanças no ambiente organizacional levam, não só, a problemas com a documentação, mas também com o gerenciamento e a evolução do sistema como um todo. Fato é que as mudanças serão constantes e os stakeholders devem estar preparados para isso [2].

Os métodos ágeis surgem com o objetivo de melhorar a comunicação entre os indivíduos no ambiente de desenvolvimento (comunicação face a face) e de minimizar as falhas no gerenciamento das mudanças. Assim, quatro valores fundamentais foram propostos: os indivíduos e suas interações devem ser considerados acima de procedimentos e ferramentas; o funcionamento do software deve estar acima de documentação abrangente; a colaboração dos clientes deve estar acima da negociação de contratos; e, a capacidade de resposta imediata às mudanças em detrimento de um plano pré-estabelecido. Não se trata de um desprezo aos métodos e ferramentas tradicionais do desenvolvimento de software, mas sim do estabelecimento de uma escala de valores, na qual a flexibilidade e a colaboração são mais relevantes do que a rigidez de processos e planejamentos clássicos.

O desenvolvimento ágil quebrou o paradigma de que não se pode planejar o desenvolvimento de um software como se planeja a construção de um prédio. Nos métodos ágeis os requisitos são desenvolvidos de forma incremental, de acordo com as prioridades do cliente. Com isso, como o setor de projetos de desenvolvimento de software está sujeito a constantes mudanças. Os métodos ágeis objetivam atender às rápidas mudanças[3].

Práticas de software ágeis obtêm uma maior agilidade através da auto-organização de equipes, colaboração do cliente, maior qualidade da documentação, menos tempo reduzido para o mercado. Suportam iterações que podem integrar qualquer mudança nos requisitos do usuário, durante a fase de implementação [4]. Todavia, após a entrega do produto o trato da evolução dos requisitos não é algo trivial. Faz-se necessário monitoramento de todo o conjunto de requisitos que permanece consistente, bem como a relação entre os requisitos, com as decisões tomadas junto à equipe de teste. Nos métodos ágeis, a evolução dos requisitos começa na fase de monitoramento, após a entrega do produto.

Apesar de tudo, alguns pesquisadores afirmam que os métodos ágeis se aplicam melhor a contextos específicos [2]. Alguns trabalhos apontam as diversas fragilidades desses métodos [5], sobretudo no tocante a engenharia de requisitos. Dentre os problemas apontados, destacamos a gerência de requisitos e a evolução desses.

Na Seção 2, são apresentados os objetivos da pesquisa. Na Seção 3, são descritas as contribuições científicas deste artigo. As conclusões são apresentadas na Seção 4. Os trabalhos futuros são apresentados na Seção 5 e por último, na Seção 6, as referências.

2 Objetivos da Pesquisa

A evolução de requisitos continua sendo um grande problema quando nos referirmos a uma metodologia ágil. Ainda não se sabe qual o impacto da evolução de requisitos é na prática nos métodos ágeis. Procuramos entender melhor as abordagens atuais utilizadas no gerenciamento de requisitos e os problemas que surgem na evolução de requisitos em uma metodologia ágil.

De acordo com Jaqueira et al.[5], Espinoza e Garbajosa [9] a diferença entre o desenvolvimento de software no modelo ágil e tradicional não se encontra apenas na forma como os processos são realizados, mas também nos artefatos produzidos como saídas de cada processo.

Para tentar solucionar a fragilidade dos métodos ágeis em tratarem a gerência de requisitos e da evolução desses, faz-se necessário identificar mecanismos de como tratar da evolução dos requisitos nas metodologias ágeis sem ferir, fundamentalmente, seus princípios.

Desta forma, este artigo propõe a utilização da técnica i* [8], para tratar a análise do processo da evolução dos requisitos em metodologias ágeis, tais como, o Scrum.

3 Contribuições Científicas

Evolução de requisitos é um processo de mudança contínua de requisitos em uma determinada direção. No entanto, pesquisas mais atuais se concentram em como lidar com a evolução depois que acontece. Evolução de requisitos acontece depois que um produto é entregue [6]. Este processo continua a ser um fenômeno pouco compreendido a partir das perspectivas de evolução.

No gerenciamento dos requisitos, o acompanhamento da evolução é necessário com o objetivo de se evitar problemas inesperados durante todo o ciclo de vida do sistema. O monitoramento de requisitos geralmente é realizado através da captura de informações específicas sobre os requisitos e é, especialmente, importante quando em sistemas de software de grande porte, com longo ciclo de vida.

Os sistemas de grande porte são muitas vezes baseados em produtos previamente lançados e muitos de seus componentes são reutilizados e /ou prolongados [1] fazendo com que a qualidade não fique em um nível desejado. Além disso, esses sistemas

estão sendo desenvolvidos por muitas equipes diferentes em diferentes localizações físicas o que deve levar a um maior cuidado não só em seu desenvolvimento, mas também em sua manutenção e na evolução de seus requisitos.

Esse assim chamado desenvolvimento global de software é introduzido por muitas empresas e essa tendência deverá crescer cada vez mais. Essa forma de desenvolver softwares com a terceirização globalizada faz com que o gerenciamento de requisitos se torne ainda mais difícil de organizar e implementar em um projeto [1]. Contudo, sabe-se que, ao gerenciar a evolução dos requisitos, evita-se que problemas futuros como custo elevado, falhas no gerenciamento de mudanças e baixa qualidade dos artefatos produzidos possam ocorrer no desenvolvimento. Nos métodos ágeis, a gerência de requisitos e a evolução não são tratadas como parte do desenvolvimento de software. Isso representa uma fragilidade desse tipo de metodologia.

Com isso, em foco, a utilização da técnica i* pode proporcionar um mecanismo de visualização dos impactos na evolução do sistema pretendido. Metas podem deixar de ser satisfeitas, impactando diretamente nas mudanças dos requisitos. Da mesma forma que metas flexíveis, que representam requisitos não funcionais, podem também serem prejudicadas e impactarem na qualidade do sistema que esteja sendo evoluído.

A técnica i* é uma ontologia rica que trata das relações entre atores e que vem sendo bastante utilizada nos dias atuais [8] [12]. Salienta-se ainda que a técnica i* captura requisitos organizacionais e metas modelando sistemas e seus ambientes em termos dos relacionamentos intencionais entre atores estratégicos [10]. Da mesma forma, procura-se saber o que o ator quer, como ele consegue o que quer e de quem depende para conseguir o que quer. Contudo, esta técnica tem por objetivo articular a noção de intencionalidade distribuída. O foco nos stakeholders e seus relacionamentos para descrição de requisitos é uma característica da técnica i*, onde os atores dependem uns dos outros para atingir suas metas. Os métodos ágeis também se concentram em fatores humanos e apostam na entrega de valor para o cliente. No grupo outras experiências com Scrum e i* já tiveram êxito [5], [11].

A partir da avaliação do uso da técnica i* em conjunto com métodos ágeis, em particular com o Scrum, através de casos exemplos será possível: (i) verificar como o uso dos modelos i* enriquecem a análise da evolução dos requisitos em métodos ágeis, contribuindo para a melhoria dos projetos de desenvolvimento ágil de software e (ii) avaliar a viabilidade de seu uso como forma de documentação da evolução dos requisitos.

4 Conclusões

A gerência de requisitos engloba todas as partes que contribuem para a geração de um documento de requisitos e sua manutenção ao longo do tempo. A evolução é considerada uma das questões mais críticas no desenvolvimento de sistemas.

Por outro lado, têm-se os métodos ágeis inseridos em ambientes de desenvolvimento de software dinâmicos, onde os requisitos estão em constante modificação e são construídos a partir do *feedback* das partes interessadas (stakeholders) no decorrer do desenvolvimento. O não gerenciamento dos requisitos e o trato com sua evolução

afeta não só a confiabilidade do sistema, mas também as atividades de engenharia e do ambiente.

Em particular, a técnica ágil Scrum apresenta dificuldades não só em capturar requisitos, mas, sobretudo em gerenciá-los e, portanto, na evolução dos sistemas. Em se tratando de requisitos organizacionais de dependência e intencionalidade, que promovem o reuso e agilidade, poucas são as técnicas que os consideram. Em Scrum a dependência não é explicitamente considerada.

A técnica i* por sua vez, é utilizada para o mapeamento organizacional, sendo amplamente pesquisada na academia. Possui uma série de vantagens, dentre elas a derivação sistemática de requisitos a partir de objetivos, o alcance da completude das especificações de requisitos, além de terem os objetivos mais estáveis do que os requisitos. Sua notação gráfica propicia a visibilidade dos relacionamentos de dependência e intencionalidade entre entidades que se relacionam (atores). Essa característica pode vir a contemplar às técnicas ágeis de recursos gráficos, pois o foco, tanto na técnica i* e quanto nos métodos ágeis, está em todos os agentes envolvidos no processo. Assim, estamos investigando como usar o modelo i*, para tratar a evolução dos requisitos nos métodos ágeis.

5 Trabalhos Futuros e em Andamento

Apesar do reconhecido papel da engenharia de requisitos, a evolução de requisitos continua sendo um fenômeno pouco compreendido [7]. Dessa forma, abordagens que tratam dessa questão são sempre importantes, pois da evolução depende o sucesso de um projeto.

Atualmente, está sendo realizado um trabalho de revisão sistemática acerca das principais fontes de discussão sobre como a evolução de requisitos está sendo abordada pela comunidade acadêmica e na indústria. O foco será com relação, aos métodos ágeis, em particular o Scrum, e aos desafios associados a esses. Será analisado se as relações de dependências dos atores da organização podem promover o reuso do conhecimento da organização, para os times, em vários projetos. Para tanto, será considerada a inclusão de conceitos avançados da técnica i*, tais como papéis, agentes e posições. Outro ponto a ser investigado diz respeito ao uso do mapeamento organizacional muito além dos projetos de desenvolvimento e manutenção de software, mas com relação à análise de riscos desses projetos. Precisar-se-á avaliar, contudo, a compatibilidade com a característica de agilidade do Scrum.

Esta proposta surgiu para proporcionar aos stakeholders e toda equipe que faz parte do desenvolvimento a importância da evolução dos requisitos nos dias de hoje e um mecanismo gráfico de análise da evolução dos requisitos. Para validar a proposta, serão conduzidos exemplos com a participação da academia e da indústria.

6 Referências

1. Lormans, M.: Managing Requirements Evolution using Reconstructed Traceability and Requirements Views. Department of Software Technology. Delft University of Technology, (2009).
2. Puntel, M. and Prass, F.: Um Método Ágil Híbrido. Sistemas de Informação. Universidade Luterana do Brasil, Cachoeira do Sul, (2010).
3. VIANA, Antônio Geraldo Gonçalves.: Gerenciamento de projetos em processo ágil de desenvolvimento de software. Instituto de Educação Tecnológica – IETEC, Belo Horizonte.[20-] Disponível em: <www.techoje.com.br/site/techoje/categoria/abrirPDF/393>, Acesso em: abr. 2013.
4. Ahmed, A., Ahmad, S., Ehsan, Dr. N., Mirza, E. and Sarwar, S. Z.: Agile Software Development: Impact on Productivity and Quality, *Proceedings of the* 2010 *IEEE ICMIT*, Singapore, 2-5 June (2010), pp. 287-291.
5. Jaqueira, A., Andreotti, E., Lucena, M., Aranha E.: Desafios de Requisitos em Métodos Ágeis: Uma Revisão Sistemática. 3rd Brazilian Workshop on Agile Methods, São Paulo (2012).
6. Li, J., Zhang, H., Zhu, L., Jeffery, R., Wang, Q. and Li, M.: Preliminary Results of A Systematic Review on Requirements Evolution, In: 16th International Conference on Evaluation & Assessment in Software Engineering (EASE 2012), Institute of Software, Chinese Academy of Sciences, China, (2012), pp. 12–21
7. Anderson, S., Felici, M.: Quantitative aspects of requirements evolution, In: Computer Software and Applications Conference, 2002. COMPSAC 2002. Proceedings. 26th Annual International, (2002), pp. 27- 32.
8. Yu, E.: Modeling Strategic Relationships for Process Reengineering. In: Social Modeling for Requirements Engineering. Cambridge, Massachusetts, London, England: MIT Press, (2011) p. 11-152.
9. Espinoza, A. and Garbajosa, J.: Study to Support Agile Methods More Effectively through Traceability, Computer Science Innovations in Systems and Software Engineering, Vol. 7, No. 1, (2011), pp. 53-69.
10. Alencar, Fernanda M. R. de: Mapeando a Modelagem Organizacional em Especificações Precisas. 302 fls. Tese (Doutorado em Ciência da Computação). Universidade Federal de Pernambuco, Recife, Centro de Informática, Programa de Pós-Graduação em Ciência da Computação, (1999).
11. Scheidegger, Ma. E. S. de: Integrando Scrum e a Modelagem de Requisitos Orientada a Objetivos por meio do SCRUM i*. Dissertação (em Ciência da Computação). Universidade Federal de Pernambuco, Recife, Centro de Informática, Programa de Pós-Graduação em Ciência da Computação, (2011).
12. i*Wiki "IstarWiki community". Disponível em: <http://istar.rwth-aachen.de/tiki-index.php?page=i%2A+Wiki+Home>. Acesso em: abr. 2013.

On the Impact of Software Ecosystems in Requirements Communication and Management

Rodrigo Pereira dos Santos, Cláudia Maria Lima Werner

System Engineering and Computer Science Department – PESC/COPPE
Federal University of Rio de Janeiro – Zip Code 21945-970 – Rio de Janeiro, Brazil

{rps, werner}@cos.ufrj.br

Abstract. The treatment of economic and social issues in Software Engineering was pointed out as a challenge for the next years. Companies and organizations have directly (or not) opened up their software platforms and assets to others, including partners and third-party developers around the world, generating software ecosystems (SECOs). This changes the traditional software industry because it requires mature research in software requirements and architecture in an environment where business models and socio-technical networks can impact the management of the platforms' needs and demands overtime. However, one strong inhibitor is the complexity in defining and modeling SECOs elements to improve their comprehension and analysis and their impacts on requirements engineering. So, this paper introduces an approach to support SECO definition and modeling based on the SECOs domain. The goal consists in dealing with the stakeholders' value proposition and realization in SECOs, as well as treating nontechnical issues in components and social repositories.

Keywords: Software Ecosystems, Conceptual Modeling, Domain, Sociotechnical Networks, Requirements Management, Repositories, Reuse.

1 Introduction

Software Engineering (SE) field has directly supported software industry through methods, techniques and tools to develop interconnected and large-scale software systems in a rapid speed of deployment and evolution [5]. According to Boehm [4], the main goal of SE is to create products, services and processes to earn value to society considering its different facets and perspectives. Software industry exists since it produces value realized by its stakeholders [3]. So, the way their interests and expectations are communicated is critical for the manner they are heard and effectively influence future solutions to meet their needs [6]. In parallel, in the software vendor's point of view, large-scale software development process is complicated, expensive, slow, and unpredictable [9]. This remaining challenge motivates research and practice communities to understand the economic and social issues in SE [18]. In this context, some explanations can be highlighted:

- software development process requires to carefully think about the platforms which will support it as well as the networks among its artifacts and stakeholders (socio-technical networks), i.e., connectivity and dependency among products and organizations, for example, suppliers, distributors, third-parties, developers, consultants and other organizations and clients that affect (and are affected by) this scenario [2];
- innovations no longer arise from an organization, i.e., they result from a synergy of different agents of software industry called co-innovation, in such a way organizations collaboratively join and focus on supporting new products to satisfy clients' needs and requirements and to incorporate new innovation cycles [19].

As stated by Bosch [5], software engineers should have abilities to abstract the complexity of a system as a whole, which is composed by software, hardware and peopleware joint around a common environment (i.e., platform). Additionally, the traditional perspective of SE has been deconstructed in order to consider the birth, development, maturation, and "death" or transformation of platforms, where collaboration and interoperability among actors and artifacts are crucial [16]. It means that the development of software products and systems generally requires collaboration of many individuals, groups, and organizations that form an ecosystem of interdependent artifacts and stakeholders [21]. The underlying structures were discussed in SE research as software ecosystems (SECOs) in the early 2000s [8]. Some of the biggest software organizations are heading SECOs development such as Amazon, Microsoft, Nokia, SAP, Google, and Apple. For this reason, SECO is reaching a status of research topic basically conceived from the movements of software industry and its related services [12]. However, the first researches about SECOs were done by Business Schools in the 90s, as discussed by Bosch [5] and Santos & Werner [15].

Based on previous studies from literature, in this research we define SECOs as *sociotechnical networks*[1] for developing software products and services, that are composed by technical, transactional and social components that relate to each other in order to engineer and manage one or more platforms, generating value and innovation in software industry. Some examples are Eclipse SECO, Microsoft SECO, and iPhone SECO [8]. SECOs studies in SE field were initially motivated by the software product line approach aiming at allowing external developers to contribute to hitherto closed platforms through a global software industry [5][13]. SECOs community has discussed the research directions at literature and industrial cases that reinforce important SECO perspectives, such as requirements, architecture, mobile platforms, global SE, social and sociotechnical networks, modeling, business considerations, organizational-based management, and multidisciplinary studies [2].

Since software vendors resort to virtual integration through alliances to create and keep networks of influence and interoperability in SECOs worldwide [15], different requirements communication and management networks are produced and should be

[1] *Socio-technical* networks are graphs of nodes (actors and artifacts) and edges (their dependencies). In turn, *sociotechnical* networks extend them to contemplate a multidisciplinary view, including other elements to analyze SECOs facts and artifacts based on the actor-network theory [7].

maintained [9]. According to Fricker [6], it means that large-scale organizations need to consider the interplay of a considerable number of stakeholders for defining requirements of their commercial and technical products and platforms. For Paech *et al.* [9], different specifications are used to negotiate and document agreements that match stakeholders' propositions and realizations, e.g., marketing requirements specifications to define the product related offering by product management; use case specifications to align product management and users; technical specifications to align development and product managements; and system specification to align team leaders and development management.

Based on this discussion, requirements communication and management in SECOs is a challenge, especially in the distributed software development scenario [17]. Fricker [6] points out (i) *tactics and methodology problems* as responsible for difficulties in understanding the matching between requirements and solutions, and (ii) *strategic problems* as responsible for mismatching between interests and expectations that is critical to prepare an organization and its markets to accept new software products, systems, services, and SECO platforms. Moreover, a transition from conventional structures and relationships in industry to a SECO will likely have impacts on business and technical specification and design choices [15]. Thus, the communication and management of needs and requirements can be affected by SECOs definition and modeling since they depend on strategic goals, intentions and relationships of each actor in a network of both actors and artifacts [2].

2 Objectives of the Research

The novelty and complexity of SECO research in SE produce many issues especially related to a vague and diverse concept of SECOs and to the lack of results and contributions from empirical studies [8]. So, it is important to provide a conceptual and technological support for defining and modeling SECOs, as well as the impacts on requirements communication and management. In order to contribute in this direction, the current research aims at exploring the concept of domain for SECOs to develop an approach to improve the comprehension of SECOs in a globalized software development environment, named *ReuseSEEM* (**Reuse**-based **S**oftware **E**cosystems **E**ngineering and **M**anagement). The concept of domain is inspired in that one used in Software Reuse [20].

The focus of the research is to consider business and social elements to understand both the internal view of a SECO (i.e., organizational) and the external view (i.e., software supply network and related ecosystems). In this case, business and social elements should enrich the definition and modeling of SECOs, such as pricing, marketing, negotiation, and evaluation, from the business side [12][14][15], and interaction, utility, reputation, promotion, contribution, and recommendation, from the social side [11][16][19]. On the other hand, an application of *ReuseSEEM* consists in mapping the knowledge of networks of actors and artifacts to SECOs' needs and requirements. Thus, platform, products and services' requirements can be communicated and managed in a SECO environment, which is usually distributed, interactive and dynamic.

3 Scientific Contributions

Four steps were established for the *ReuseSEEM* approach, as shown in Fig. 1: (1) *definition*: create and validate a body of knowledge for the SECOs domain through a conceptual model enriched with variability (e.g., actors with different roles in distinct SECOs); (2) *modeling*: map the conceptual modeling to sociotechnical and software supply networks in order to visualize and browse through the SECOs; (3) *analysis*: select a SECO (or part of it) from a stakeholder's point of view in order to analyze its different perspectives and levels based on business and social elements, e.g., identify needs and requirements for a SECO platform, or suggest new niches or SECOs of interest for a specific organization; and (4) *maintenance*: provide a research strategy to support empirical studies aiming at generating and feeding a historical data and experience reports repository since there are many research and practice targets to be analyzed in SECOs from the SE field. The link between the modeling and analysis of SECOs is created through a repository of SECO components (e.g., registered SECOs, actors, platforms, artifacts, previous software supply networks etc.) and a social network site (e.g., actors and artifacts exposed to the global software industry).

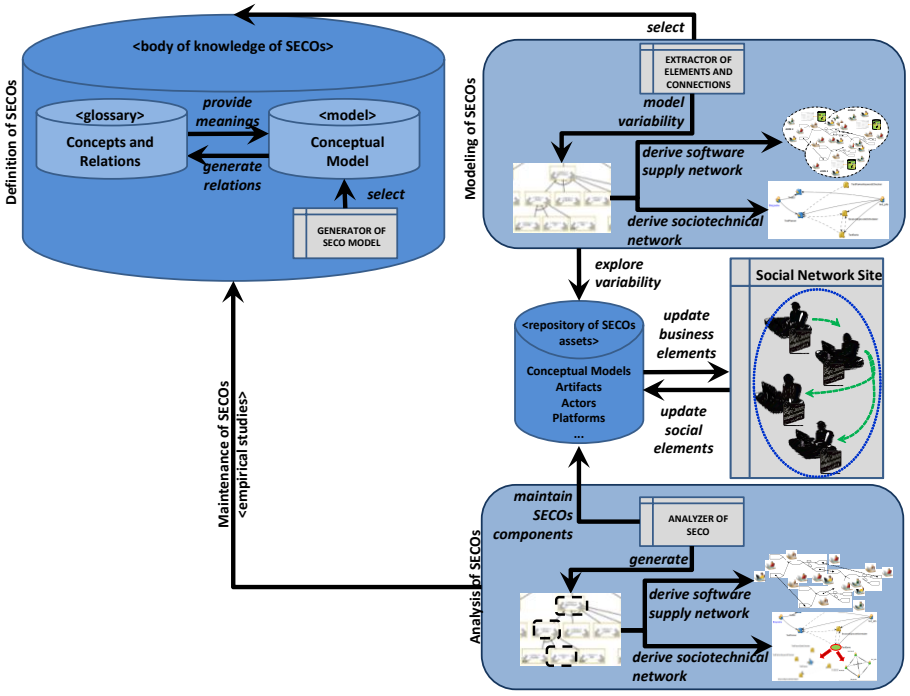


Fig. 1. *ReuseSEEM* approach.

The first two steps are the initial focus of this research, i.e., definition and modeling of SECOs. As mentioned in Section 2, the main contribution of this research consists in improving the comprehension of SECOs from the SE point of view. As such, *ReuseSEEM* tries to use the SECOs metaphor to understand the economic- and social-based SE view as well as to better identify, visualize and

manage opportunities, requirements, market niches, clients' needs and systems evolutions. So, modeling of requirements, goals and domains can be treated by *step 3* through the development of a tool to support requirements communication and management based on a community participation to suggest and solve client's needs in an extended social network site. Finally, as empirical contributions, we intend to plan and execute at least two studies to verify *ReuseSEEM*: (1) a *survey with experts* in SECOs in order to evaluate the conceptual model; and (2) a *case study* in order to evaluate the definition, modeling and analysis of a SECO (in this case, using the mentioned tool focused on requirements communication and management).

4 Conclusions

Since SE community is dealing with both technical issues and other kinds of concerns in its evolution as a research field, SECOs emerge as a topic to investigate the different and integrated perspectives of the global and dynamic software industry, i.e., technical [10][13], transactional [15] and social [16]. Despite the efforts performed by researchers and practitioners, the SECO domain is still vague and divergent [2][8]. At the same time, its comprehension is becoming very important to the SE point of view because SECOs metaphor allows understanding its activities, such as requirements communication and management [6][11]. This research intends to contribute in defining and modeling the SECOs domain as well as use this knowledge to analyze cases of SECOs in the SE point of view [17]. Apart from applying requirements engineering (RE) concepts (i.e., requirements, goals and domains) to develop the first two steps of *ReuseSEEM*, this approach can contribute to RE research as an instrument for identifying, visualizing and managing opportunities, requirements, market niches, clients' needs and systems evolutions in the context of SECOs (*step 3*).

5 Ongoing and Future Work

Nowadays, we are working in three tracks: (1) developing *step 1* of *ReuseSEEM*, i.e., mapping the SECO domain and creating a conceptual model through a map of concepts (e.g., actors, artifacts, relationships, roles and responsibilities) [11] in order to execute an empirical evaluation with experts in SECOs; (2) developing an application for *step 3* of *ReuseSEEM* in *architecture*, i.e., defining a technology recommender to support a SECO governance approach for enabling an information technology (IT) architecture based on software asset management [1]; and (3) developing another application for *step 3* of *ReuseSEEM* in *RE*, i.e., mapping sociotechnical networks through the extension of social network sites to communicate and manage needs and requirements in SECOs [19]. Although *steps 1* and *2* are not finished, tracks (2) and (3) can be independently built and initially evaluated based on the common sense on SECO concepts and models identified and studied by the Software Reuse Lab at COPPE/UFRJ since 2009.

Acknowledgments. The authors thank to CNPq/FAPERJ for the financial support.

References

1. Albert, B., Santos, R., Werner, C.: A Study on Software Components Governance Based on SOA Governance Elements. In: 6th SBCARS, pp. 120-129, Natal, Brazil (2012)
2. Barbosa, O., Santos, R.P., Alves, C., Werner, C., Jansen, S.: A Systematic Mapping Study on Software Ecosystems through a Three-dimensional Perspective. In: Jansen, S., Cusumano, M. & Brinkkemper, S. (eds.) Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry, pp. 59-81. Edward Elgar Publishing (2013)
3. Biffl, S., Aurum, A., Boehm, B., Erdogmus, H., Grünbacher, P.: Value-Based Software Engineering. Springer-Verlag (2005)
4. Boehm, B.: A View of 20th and 21st Century Software Engineering. In: 28th International Conference on Software Engineering, pp. 12-29, Shanghai, China (2006)
5. Bosch, J.: From Software Product Lines to Software Ecosystem. In: Proceedings of the 13th Software Product Line Conference, pp. 1-10, San Francisco, USA (2009)
6. Fricker, S.: Specification and Analysis of Requirements Negotiation Strategy in Software Ecosystems. In: 1st IWSECO, pp. 19-33, Falls Church, USA (2009)
7. Latour, B.: Science in Action: How to Follow Scientists and Engineers Through Society. Harvard University Press (1988)
8. Manikas, K., Hansen, K.: Software Ecosystems – A Systematic Literature Review, Journal of Systems and Software, v. 86, n. 5, pp. 1294-1306 (2013)
9. Paech, B., Dorr, J., Koehler, M.: Improving Requirements Engineering Communication in Multiproject Environments, IEEE Software, v. 22, n. 1 (Jan/Feb), pp. 40-47 (2005)
10. Santana, F., Werner, C.: Towards the Analysis of Software Projects Dependencies: An exploratory visual study of software ecosystems. 5th IWSECO, Potsdam, Germany (2013)
11. Santos, R., Sampaio, J.: Análise e Aplicações de Redes Sociais em Ecossistemas de Software. In: IX Brazilian Symposium on Information Systems, Short courses, v. 2, pp. 19-24, João Pessoa, Brazil (2013) (in Portuguese)
12. Santos, R., Werner, C.: Revisiting the Concept of Components in Software Engineering from a Software Ecosystem Perspective. In: 4th European Conference on Software Architecture Workshops (IWSECO), pp. 135-142, Copenhagen, Denmark (2010)
13. Santos, R., Werner, C.: A Proposal for Software Ecosystems Engineering. In: 3rd IWSECO, pp. 40-51, Brussels, Belgium (2011)
14. Santos, R., Werner, C.: Brechó-EcoSys: From a Component Library to a Software Ecosystems Platform. In: 12th ICSR, Tool Demonstrations, Pohang, South Korea (2011)
15. Santos, R., Werner, C.: Treating Business Dimension in Software Ecosystems. In: 3rd ACM/IFIP International Conference on Management of Emergent Digital EcoSystems, pp. 197-201, San Francisco, USA (2011)
16. Santos, R., Werner, C.: Treating Social Dimension in Software Ecosystems through ReuseECOS Approach. In: 6th IEEE International Conference on Digital Ecosystem Technologies, pp. 1-6, Campione d'Italia, Italy (2012)
17. Santos, R., Werner, C.: ReuseECOS: An Approach to Support Global Software Development through Software Ecosystems, In: 7th IEEE International Conference on Global Software Engineering Workshops (WDDS), pp. 60-65, Porto Alegre, Brazil (2012)
18. Santos, R., Werner, C., Barbosa, O., Alves, C.: Software Ecosystems: Trends and Impacts on Software Engineering. In: XXVI Brazilian Symposium on Software Engineering – 'Grand Challenges in Software/System Engineering, pp. 206-210, Natal, Brazil (2012)
19. Santos, R., Esteves, M.G., Freitas, G., Souza, J.: Using Social Networks to Support Software Ecosystems Comprehension and Evolution, Social Networking (2013) (accepted)
20. Werner, C.: Building Software Ecosystems from a Reuse Perspective. In: 1st IWSECO, p. 3, Falls Church, USA (2009)
21. Werner, C., Santos, R.: Software Ecosystems: Status, Research Directions and the Practice in Software Industry. In: XV CIbSE, Tutorials, Buenos Aires, Argentina (2012)

Representando Características Autonômicas nos Processos de Negócio

Karolyne Oliveira, Tarcísio Pereira, Emanuel Santos, Jaelson Castro

Universidade Federal de Pernambuco—UFPE, Recife, PE 50 740-560, Brazil
{kmao, tcp, ebs, jbc}@cin.ufpe.br

Abstract. Em consequência das novas demandas de negócios e avanços tecnológicos os modelos processos de negócio estão se tornando cada vez mais complexos e heterogêneos. Isto remete a processos de negócios que devem ser gerenciados de forma autônoma em resposta a mudanças no seu contexto ambiental. Processos de Negócios que são capazes de ser auto geridos são denominados processos de negócios autonômicos (do inglês *Autonomic Business Process* - ABP). No entanto, o principal desafio é fornecer variabilidade, compreensibilidade e escalabilidade em modelos de processos de negócios que estão cada vez mais complexos. Este trabalho propõe a representação explícita de (i) variabilidade, através de um ambiente sensível ao contexto; (ii) atributos de qualidade, a fim de representar adequadamente os parâmetros do sistema; e, (iii) modularidade, através da utilização de uma abordagem multi camada para ABP. Esta nova abordagem, chamada MABUP, oferece quatro níveis bem definidos de abstração para modelagem: Nível Organizacional, Nível Tecnológico, Nível Operacional e Nível de Serviço. Para prover gerencia aos processos, um engenho autonômico utiliza as informações contidas nos modelos de processo de negócio e realiza adaptações no sistema.

Keywords: Modelagem de Processo de Negócio, Computação Autonômica, Auto Adaptação.

1 Introdução

A crescente demanda por sistemas cada vez mais complexos fez com que analistas e pesquisadores previssem, há cerca de uma ou duas décadas atrás, que haveria uma grande dificuldade em gerir as Tecnologias de Informação e Comunicação (TICs) em curto ou médio prazo. De fato, os atuais serviços disponibilizados têm o desafio de conviver com milhões de usuários, com acesso simultâneo e absolutamente estocástico. Outra demanda, não necessariamente nascida de corporações de magnitude global, é o interesse de focar os esforços e recursos dos gerentes para os assuntos específicos do negócio [1]. É preciso se construir uma infra-estrutura de TICs capaz de absorver, sem grande impacto, as mudanças advindas das decisões daqueles que regem o negócio. É necessário que não haja preocupação em como implementar tecnologicamente a solução ou mudar drasticamente o suporte de informática envolvido na mudança.

Como resposta a essas dimensões intangíveis de complexidade e a busca por foco apenas no negócio para o processo decisório, a nossa proposta é buscar menos

dependência da ação humana. A Computação Autonômica (do inglês *Autonomic Computing* – AC) é um paradigma computacional com fortes características de autossuficiência, onde a concepção de novos sistemas já parta, desde a fase de análise e projeto, de elementos que possibilitem o crescimento e pleno funcionamento dos sistemas "por si só" auto gerenciados e independente de intervenção de gerentes extremamente qualificados tecnicamente.

Por esta razão, a conveniência de modelagem organizacional durante o processo de engenharia de requisitos e o papel do analista de negócio se faz cada vez mais necessário [2, 3]. O modelo de processo de negócio (do inglês *Business Process Model* - BPM) reflete as atividades de uma organização e facilita o entendimento do domínio da aplicação. Para tal lança-se mão de métodos, técnicas e ferramentas para analisar, modelar, publicar, otimizar e controlar processos envolvendo diversos recursos da organização, entre estes podemos citar humanos, aplicações, documentos e outras fontes de informação [4]. Dentro da computação autonômica, muitas pesquisas apontam o uso atrelado à análise de variabilidade para indicar mudanças no contexto de uso da aplicação [5] tal como de requisitos não-funcionais (RNF) para representar quais indicadores devem ser garantidos.

Uma parte do sistema autonômico, denominado gerente autonômico, interage constantemente com elemento gerenciado, a fim de manter o equilíbrio diante das perturbações de entrada. O ciclo MAPE (Monitorar-Analisar-Planejar-Executar) é uma implementação da técnica clássica de *feedback control* [6]. Quando um sistema de controle fechado está em ação, o sistema é gerenciado enquanto interage com o ambiente externo. Assim, no momento em que o ambiente provoca distorções no comportamento do sistema, o monitor (também chamado "sensor") irá detectar, por meio de comparação constante entre um conjunto de métricas de saída emitidas pelo sistema e as métricas de qualidade definidas. Se a distorção entre os parâmetros constantemente em comparação excede os limites pré-definidos, o componente analisador alerta o planejador passando a informação da métrica coletada. O planjador, por sua vez, irá planejar intervenções sobre o sistema, a fim de ajustá-lo de volta ao seu estado ótimo. O plano é colocado em prática através de atuadores que irão interagir com o sistema, alterando alguns dos seus parâmetros de execução. O gerente autonômico visa dar suporte as quatro características autonômicas conhecidas: auto configuração, auto recuperação, auto otimização e auto proteção.

No entanto, abordagens atuais não levam em consideração aspectos relativos à variabilidade, compreensibilidade e escalabilidade em modelos de processos de negócios que estão cada vez mais complexos. Em sua maioria, os trabalhos relativos a gestão autonômica de processos apresentam aspectos sobre composição automática de serviços, falhas e outros aspectos da definição e execução de processos compostos. Nesta pesquisa está sendo investigado como incorporar características autonômicas aos processos de negócio. Pretendemos propor extensões as linguagens atuais de modelagem (eg. BPMN, UML, EPC, CED, etc) para permitir que os processos possam ser monitorados, os desvios de comportamento sejam diagnosticados e, se necessário, correções ou processos alternativos sejam ativados.

2 Objetivos da Pesquisa

O objetivo geral da pesquisa é investigar e propor uma extensão aos modelos de processo de negócio de modo a incorporar as características autonômicas. Esta extensão deverá permitir uma maior expressividade desse modelo para indicar

características que afetam e modificam diretamente os fluxos de negócio e consequentemente seus sistemas de suporte.

Os objetivos específicos são:

- **Objetivo 1** – Investigar sobre modelagem de processos de negócio e suas extensões para expressar variabilidade, RNF e características autonômicas;
- **Objetivo 2** – Propor uma linguagem de modelagem de processo de negócio que expresse adequadamente variabilidade e as características autonômicas;
- **Objetivo 3** – Desenvolver uma ferramenta de suporte à modelagem e gestão de processo de negócio autonômico de acordo com o ciclo MAPE;
- **Objetivo 4** – Validação dos mecanismos propostos para expressão de variabilidade no contexto de sistemas autonômicos;

3 Contribuições Científicas

Para alcançar o objetivo de prover uma abordagem para processos de negócio autonômicos, considerou-se a utilização de modularização e separação de interesses para representar as diferentes características. Esta abordagem, denominada MABUP, está retratada na Fig. 1 e apresenta uma visão geral da gestão de modelo de processo de negócio em multi camada e inclui duas fases principais: (i) a *fase de modelagem* que explora a modelagem de quatro níveis de abstração: nível organizacional, nível tecnológico, nível operacional e nível de serviço, (ii) a *fase de gestão* que inclui um engenho autonômico o qual considera os processos de negócio modelados e as métricas fornecidas pelos sistemas para orientar a adaptação realizada com um ciclo MAPE.

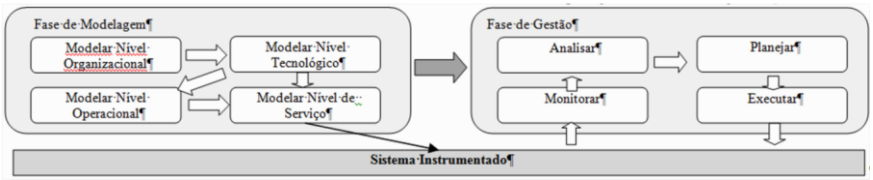


Fig. 1. Visão geral das fases de modelagem e gestão de MABUP

Este trabalho apresenta três contribuições principais: (i) um processo para orientar a aplicação da abordagem MABUP e promover o uso de diferentes níveis de abstrações em modelos de processo de negócio autonômicos; (ii) um meta-modelo para integrar os conceitos utilizados neste trabalho; e, (iii) uma arquitetura autonômica baseada no ciclo MAPE para apoiar adaptações autonômicas.

O processo para aplicação do MABUP e metamodelo, com base em [7], explora a alta variabilidade no ambiente operacional de sistemas orientados a serviços pelo uso de contextos e adaptações autonômicas por operacionalizações dos requisitos nãofuncionais (RNF). Este foi adaptado para suportar diferentes tipos de notação e é divido em sete passos, a saber:

1. Definir o nível organizacional do BPM.
2. Definir os processos de negócios autonômicos (nível tecnológico e operacional)
3. Conectar os processos de negócios autonômicos e serviços (nível de serviço)
4. Definir contextualização dos processos de negócio autonômicos

5. Definir qualidade dos serviços (QoS)
6. Monitorar em tempo de execução
7. Planejar adaptação com base no processo e nos indicadores dos serviços

A Fig. 2 apresenta o meta-modelo do MABUP e define os conceitos nos diferentes níveis de abstração utilizados na abordagem. O nível organizacional reflete um nível bem definido de modelagem no qual se assume que os processos são imune às mudanças tecnológicas, assim, é o nível apropriado para definir as atividades críticas do negócio. No nível tecnológico as atividades críticas são refinadas e descritas assim como são executadas e suportadas por alguma tecnologia, desta forma, é o nível adequado para se representar as tarefas que necessitam ser monitoradas por um engenho autonômico. No nível operacional são descritas as tarefas que devem ser executadas caso haja algum desvio nos atributos de qualidade definidos e a depender do contexto, assim, é o nível adequado para expressar tarefas operacionais e contextualização. Por fim, no nível de serviço são descritos os serviços de suporte aos processos monitorados e operacionais. Serviços monitorados devem ter explicitamente representados os atributos de qualidade que são aferidos, já os serviços operacionais são responsáveis por realizar ajustes no ambiente operacional de suporte aos processos.

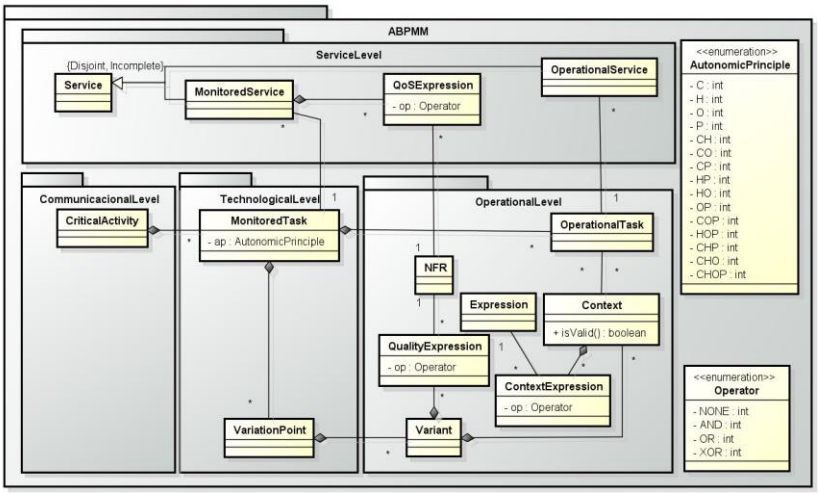


Fig. 2. Meta modelo da abordagemMABUP (MABUPMM),

A arquitetura para dar suporte à gestão autonômica dos processos de negócio, descrita em [8] considera o monitoramento de RNF e contextos para guiar ajustes no ambiente operacional. Assim, neste trabalho o ciclo MAPE foi adaptado de modo a considerar um processo de negócio instrumentado de acordo com nossa abordagem. A partir desta adaptação os módulos monitor, analisador, planejador e atuador passam a considerar os indicadores e realizar ações de ajustes guiados pelo modelo. A Fig. 3 apresenta a visão geral da arquitetura autonômica da abordagem MABUP.

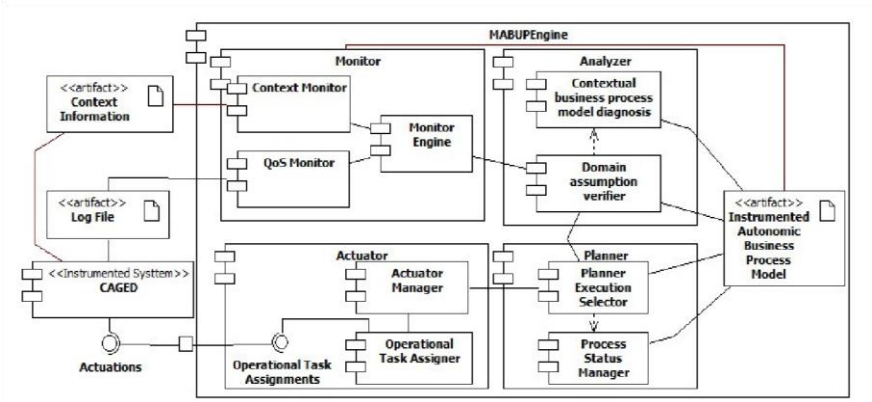


Fig. 3. ArquiteturaMABUP

4 Conclusões

Neste trabalho, foi apresentada uma abordagem, denominada MABUP, para o gerenciamento de processos de negócios autonômicos seguindo uma estratégia multi camada. Esta proposta foi dividida em duas partes: as fases de modelagem e gestão. Como resultado, três contribuições foram apresentadas: (i) processo para aplicação do MABUP; (ii) metamodelo com os conceitos utilizados na abordagem sob o ponto de vista organizacional, tecnológico, operacional e de serviço; e (iii) uma arquitetura que pretende representar o comportamento correto dos sistemas de suporte aos processos. Os benefícios são múltiplos e incluem:

- *Modularidade em ABP:* Uso de diferentes níveis bem definidos para representar as características autonômicas;
- *Escalabilidade:* A modularidade auxilia na abstração dos processos reduzindo assim a complexidade do modelo ABP, e aumenta assim sua escalabilidade;
- *Separação de interesses*: Em nosso modelo MABUP a relação entre o conhecimento de negócio, tecnológico e operacional é explicitamente expressa em diferentes níveis de abstração e interligados do modelo.
- *Expressividade de recursos autonômicos em BPM:* Em contraste com outras abordagens que necessitam de uma base de conhecimento para expressar as métricas que afetam a adaptação, a nossa abordagem fornece os conceitos de evento crítico, a tarefa monitorada, variabilidade, contexto e atributos de qualidade expressos em modelos de processo (BPM) que guia a auto gestão em tempo de execução. Todos estes conceitos estão interligados para indicar como as métricas impactam diretamente cada processo de negócio autonômico.
- *Orientar a adaptação com base em métricas, como contexto e atributos de qualidade, presentes no BPM:* Considerando o conhecimento fornecido em nosso modelo MABUP, o módulo de gerenciamento autonômico guia suas ações de acordo com os atributos de contexto e de qualidade que afetam as tarefas operacionais.

5 Trabalhos futuros e em andamento

A abordagem apresentada neste artigo é parte de um trabalho em andamento. Como tal, muitos aspectos ainda necessitam ser desenvolvidos. Atualmente, estamos desenvolvendo um arcabouço ferramental para modelagem de processo de negócio autonômico com base na abordagem MABUP. Para isto, estamos realizando uma extensão à notação BPMN de modo a inserir os conceitos relacionados ao nosso trabalho.

Como trabalho futuro, nós planejamos realizar um experimento controlado de modo a avaliar de maneira empírica nossa proposta.

6 Referências

1. Horn, P.: Autonomic Computing: IBM's Perspective on the State of Information Technology, http://www-1.ibm.com/industries/government/doc/content/resource/thought/278606109.html, (2001).
2. Bridgeland, D. and Zahavi, R.: Business Modeling: A Practical Guide to Realizing Business Value. (2009).
3. Iiba: A Guide to the Business Analysis Body of Knowledge ® (BABOK ® Guide). International Institute of Business Analysis (2009).
4. Omg: Business Process Modeling Notation (BPMN) Version 1.2. Wirtschaftsinformatik. 50, 504-507 (2009).
5. Santos, E., Castro, J., N, C.D.V.S.: A Goal-Oriented Approach for Variability in BPMN. In: Hadad, G., Dieste, O., and Carvallo, J.P. (eds.) WER10. pp. 17-28 (2010).
6. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. IEEE (2003).
7. Oliveira, K., Castro, J., Santos, E., Fidalgo, R., Espana, S., Pastor, O.: A Multi Level approach to Autonomic Business Process. 26th Brazilian Symposium on Software Engineering (SBES 2012). , Natal, RN (2012).
8. Oliveira, K., Castro, J., España, S., Pastor, O.: Multi-level Autonomic Business Process Management. International Conference on Business Process Modeling, Development and Support (BPMDS 2013). pp. 184-198 (2013).
9. Reijers, H.: Modularity in process models: Review and effects. Business Process Management. 20-35 (2008).

Developing IT Systems to Leverage Information and Knowledge Management

José Viterbo, Maria Luiza d'Almeida Sanchez and Carlos Alberto Malcher

Laboratory of Management in Communication and Information Technology,
Fluminense Federal University, Niterói, Brazil
viterbo@ic.uff.br, mluiza@gteccom.uff.br
malcher@gteccom.uff.br

Abstract. *Information and knowledge management are fundamental components for helping organizations in efficiently acquiring, storing and utilizing knowledge for problem solving, dynamic learning, strategic planning and decision making. In a previous work we presented the Information and Knowledge Management Framework (MGIC), a methodology for building a model to serve as basis for the implementation of integrated information and knowledge systems that support business intelligence in public organizations. The first step of this methodology consists on building an integrated model, based in well-known modeling techniques, such as Business Use Case models and class models, which focus on the representation of the main business requirements and services provided by the organization. In this paper we discuss how such model is used to guide the requirements specification and development of new IT systems or the evolution of legacy systems.*

Keywords: Business Intelligence, Knowledge Management, Software Development

1 Introduction

The term Business Intelligence (BI), coined in the early 1990s, is widely used today to describe analytic applications, i.e., systems that combine data gathering, data storage, and knowledge management with analytical tools to present complex internal and competitive information to planners and decision makers. BI allows managers to make informed and intelligent decisions regarding the functioning of their organizations. Informed decisions lead to better, more efficient processes in the actual work environment, and help create a powerful competitive advantage.

As organizations become more automated and data-driven, the industry is evolving toward BI systems that support operational decision making at all levels in the organization. There is a growing realization that BI must be integrated into the business operations of the enterprise to enable the many knowledge workers engaged in business processes to make better and quicker decisions. This scenario imposes several new requirements on the architecture of software

systems and on the back end data integration processes, such as handling a much larger number and diversity of data sources and data types, more complex analytic and reporting tools, a larger number of data mart connections, 24x7 availability, etc.

Besides data architecture, the knowlegde management is also an integral component of BI and hence decision making in the organization. Knowledge management and BI need to be considered together as necessarily integrated and mutually critical components in the management of intellectual capital. Knowledge management, which may be described as a systematic process of finding, selecting, organizing, distilling and presenting information in a way that improves the comprehension of the organization's employees in specific areas of interest, helps an organization to gain insight and understanding from its own experience. Specific knowledge management activities help focus the organization on acquiring, storing and utilizing knowledge for such things as problem solving, dynamic learning, strategic planning and decision making.

As such, information and knowledge management are fundamental components for helping organizations in efficiently acquiring, storing and utilizing knowledge for problem solving, dynamic learning, strategic planning and decision making. In a previous work we presented the Information and Knowledge Management Framework (MGIC), a methodology for building a set of models to serve as basis for the implementation of integrated information and knowledge systems that support business intelligence in public organizations [1].

The MGIC aims at identifying the whole information flow in a organization, since it is collected from internal and external sources, stored, processed and output. The first step of this methodology consists on building an integrated model, based in well-known modeling techniques, such as business use cases and class models, which focus on representing the main business requirements and services provided by the organization. In this paper we discuss how such model is used to guide the requirements specification and development of new IT systems or the evolution of legacy systems.

2 Objectives of the research

The first step of the MGIC methodology consists on building an integrated model that comprises four modules: an information flow model, an ontology model, a knowledge model and a requirements model. Each model is built based in well-known modeling techniques, such as BPMN, OntoUML [2], business use cases, etc. When the final model is ready, the MGIC comprises a set of guidelines that governs the creation, the validation, the dissemination, the storage, the use and the recovery of information and knowledge to allow the organization to fully reach its goals. In our present research we are interested in defining how this set of models may be used to guide the software development process in the organization. In particular, software development is mainly influenced by the requirements model, which focus on the representation of the main business requirements and services provided by the organization.

The RUP (Rational Unified Process) methodology, which suits the development of large projects, is partly employed to build the requirements model, in an iterative approach. Each iteration includes one or more disciplines of development, has a set of well-defined objectives and produces a partial model of the final system, using as input the partial model of the previous iteration to evolve and refine the system to get to the final product, complying with all rules and requirements. To deal with the iterative development, RUP provides a structured approach, dividing the project into four distinct phases: conception, design, construction and transition. Each of aggregate activities required to achieve a certain goal.

In the MGIC methodology, efforts are limited to the design phase. The main RUP disciplines approached in this phase are the identification of business requirements and the development of the information model. During the identification of requirements, the techniques used consist of interviews, questionnaires, group dynamics, study of documentation (legislation, existing systems, standards and business rules, etc ...) and on-site observation. The main model developed during this discipline is the set of functional requirements expressed in the form of user needs through the use case business models. Moreover, during the course of the following discipline, the information model is built. It consists of a class diagram, identifying relevant information and their associations to comply with the provisions of the use case model, thus proposing an information architecture. This model is supported by textual descriptions that facilitate the understanding of systems developers. It is important to stress that this model is a conceptual data model, i.e., it does not incorporate characteristics of a particular implementation. All models built are validated by the users who provided the information for their preparation.

In the next section, we discuss how such model must be used to guide the requirements specification and development of new IT systems or the evolution of legacy systems.

3 Scientific contributions

As a further step to MGIC, we propose a software development process (SDP) as a methodology for developing software coherently with the model previously described. This methodolgy is also based on RUP and aims at standardizing the life cycle of a software development project. One of the benefits of adopting a well-defined process is to increase the level of productivity of the technical teams involved in projects, because it formalizes the distribution of activities and assignments for each role. Besides being a mechanism for obtaining a high quality product, the SDP also aims at supporting the definition of agreements on future contracts for service software development. Our proposed SDP comprises six stages: (a) preliminary study, (b) initiation, (c) elaboration, (d) construction, (e) transition and (f) acceptance. It involves the Information Management team (IM), Knowledge Management team (KM), the Information Technology

team (IT), the client (any unit or member in the organization) and the software development company contracted (contractor).

3.1 Preliminary study

In the preliminary study, the feasibility of implementing a functional demand is analyzed in the following steps:

1. IM receives from the Client a request for implementing a particular functionality.
2. Based on the MGIC models, IM analyzes the request in order to verify:
   - If this functionality or other one similar is already implemented in any subsystem or if it fits in an existing subsystem. In this case, such subsystem will undergo an evolutive maintenance.
   - If this functionality must be implemented in a new subsystem. In this case, the development of such subsystem will follow this process.
3. If necessary, IM and KM update the MGIC models. Any new business use case model must be ready before the start of the system analysis (next stage). IM prepares (or updates) a requirements model, defining business use cases, information model and business rules.
4. IM prepares the preliminary study, a document that defines to which services specified in MGIC these features are associated, if new services/business requirements will have to be updated in the MGIC model and gguidelines for the system development (system that will be changed, requirements priorities, requirements to be met).
5. IT prepares the preliminary project containing a brief description of the project, a technical feasibility study considering the costs-benefits relation associated, project estimation, macro schedule and technology options/software architecture.
6. IT prepares the Project Opening Document, validated by IM.

In this stage, steps 2 to 4 are additions to the traditional software development process, in order to provide for the use of the MGIC model.

3.2 Initiation

The initiation stage aims at reaching consensus on the project goals among all stakeholders, and absorb and refine the information presented by the preliminary study. It comprises the following steps:

1. IT prepares (or reviews, in case of maintenance) the vision document for development projects.
2. IT prepares (or reviews, in case of maintenance) the project glossary, maintaining consistency with the ontology model.
3. IM validates the vision document and the glossary and determines whether it is necessary to maintain the ontology model.

4. IT details the requirements model, defining system use cases.
5. IT prepares the detailed schedule of the project and a list of implementation risks.

In this stage, step 3 is an addition to the traditional software development process to garantee that MGIC model is consistently updated.

3.3 Elaboration

The goal of the elaboration stage is to define a baseline for the system architecture, aiming at providing a stable base for the effort of the construction stage. It comprises the following steps, each validated with IM and the Client, and, when a software development company is contracted, with IT, ensuring the quality of each product:

1. IT refines the use cases with sequence diagrams and defines the class model, consistently with the logic information model.
2. IT refines the software architecture, considering non-functional requirements.
3. IT elaborates the interaction model and storyboard.
4. IT prepares preliminary user manual and training plan.
5. IT prepares the data model, which must be consistent with the information model (classes that access persistent data and their access methods).
6. IT prepares the acceptance tests plan (and homologation).
7. IT develops a map of function points.

3.4 Construction

The purpose of the construction stage is to complete the system development based on the architecture previously defined. It comprises the following steps:

1. The contractor implements the use cases.
2. The contractor prepares the system interaction manual, which complements the user manual.
3. The contractor prepares a maintenance test plan and test programs.

3.5 Transition

The transition stage focus on ensuring that the software is made available for its end users. This includes testing the product in preparation, comprising the following steps:

1. The contractor prepares the implementation plan.
2. The contractor prepares the installation manual.
3. The contractor and IT perform acceptance testing and generate an approval report. If there are outstanding issues, a plan is established to solve the problems raised.
4. The contractor deliveries the product.

3.6 Acceptance

In the acceptance stage, the main goal is to assess the results and service levels, based on the user's feedback. In this stage, steps 1, 4 and 5, that would tipically be executed by IT, are due to IM instead. Besides, KM is in charge of step 3, adhering to MGIC methodology, as follows:

1. IM and the client perform tests in the homologation environment to verify compliance with the specified requirements.
2. IT deploys the system in a production environment (client).
3. KM trains the users of the system for deployment.
4. IM conducts a satisfaction survey with the client.
5. IM issues the project closure term.

4 Conclusions

MGIC comprises a framework that governs the creation, the validation, the dissemination, the storage, the use and the recovery of information and knowledge to allow an organization to fully reach its goals. In our present research we are interested in defining how this set of models guide the development of new software functionalities to give support to some organizational business requirement. We proposed a software development process (SDP) to guide the development of software coherently with the MGIC models. One of the benefits of adopting this well-defined process is to increase the level of productivity in the organization. In particular, this SDP assures that new software functionalities will meet the general business requirements previously modeled. Moreover, the model will be maintained in order to incorporate any new requirements that may be identified.

5 Ongoing and future work

.

As an ongoing work, MGIC framework is being applied to create a model for the Brazilian public agency responsible for the regulation of terrestrial transpost services. While the model is being built, legacy systems are being mapped to check which business requirements they satisfy. The SDP has just being proposed and the implementation of new software functionalities is being discussed to serve as a case study.

References

1. Rezende, L., Malcher, C., Lobo, A., Castro, J., Burmann, C., Merino, L.: A Gesto do Conhecimento na Cadeia de Valor Colaborativa para o Desenvolvimento Sustentado do Transporte Terrestre no Brasil. In: Proc. of Knowledge Mangement Brasil. (2012)
2. Benevides, A., Guizzardi, G.: A model-based tool for conceptual modeling and domain ontology engineering in ontouml. In Filipe, J., Cordeiro, J., eds.: Enterprise Information Systems. Volume 24 of Lecture Notes in Business Information Processing. Springer Berlin Heidelberg (2009) 528–538

Uma Estensão do STREAM para Escolha de Padrões Arquiteturais baseada em Requisitos Não-Funcionais

Fábio Silva[1,2], Marcia Lucena[1], Leonardo Lucena[2], Roniceli Moura[1]

[1]Universidade Federal do Rio Grande do Norte (UFRN), Natal, RN, Brasil
marciaj@dimap.ufrn.br, roniceli.moura@yahoo.com.br
[2]Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte (IFRN), Natal, RN, Brasil
{fabio.silva,leonardo.lucena}@ifrn.edu.br

Resumo. Cada vez mais os requisitos não funcionais são considerados primordiais aos sistemas computacionais. Satisfazer esses requisitos exige uma atenção especial com a arquitetura já que uma arquitetura inadequada introduz uma complexidade maior, além da complexidade intrínseca do sistema. Estudos mostram que apesar das atividades de engenharia de requisitos e de arquitetura de software atuarem em diferentes aspectos do desenvolvimento, é preciso executá-las de forma iterativa e entrelaçada para produzir sistemas computacionais satisfatórios. O processo STREAM apresenta uma abordagem sistemática para diminuir a lacuna entre requisitos e o desenvolvimento da arquitetura, enfatizando os requisitos funcionais, enquanto que os requisitos não-funcionais são usados de forma *ad hoc*. No entanto, alguns estudos mostram que requisitos não-funcionais impactam o sistema como um todo. Este artigo apresenta uma proposta para auxiliar o processo STREAM na realização da escolha de padrões arquiteturais a partir de requisitos não-funcionais, com propósito de guiar o refinamento da solução arquitetural.

Palavras Chave: *Requisitos Não-Funcionais, Arquitetura de Software, Padrão Arquitetural.*

1 *Introdução*

O relacionamento entre requisitos e arquitetura é considerado a parte central de um Processo de Desenvolvimento de Software (PDS) [5]. Enquanto a atividade de Engenharia de Requisitos (ER) é responsável por tratar os Requisitos Funcionais (RFs) e Não-Funcionais (RNFs) de um software, a Arquitetura de Software (AS) está relacionada com os componentes do software e a comunicação entre eles. No processo de construção de um sistema de software, a AS é considerada uma questão crítica, já que a qualidade da arquitetura está relacionada com a satisfação ou não dos requisitos chaves requeridos pelo software [4]. O processo STREAM (Strategy for Transition between REquirements models and Architectural Models) [8] surgiu com o intuito de contribuir para a diminuição da lacuna existente entre a AS e a ER. Esta estratégia,

baseada em MDD (Model-Driven Development) [11], tem como objetivo principal a geração de modelos arquiteturais a partir de modelos de requisitos i*. O STREAM é constituído por: (i) um conjunto de regras de transformação para preparar os modelos de requisitos i*, chamadas de regras horizontais; (ii) um conjunto de regras de transformação para gerar modelos arquiteturais, chamadas de regras verticais; e (iii) um processo sistemático para auxiliar os desenvolvedores. O processo sistemático para auxiliar os desenvolvedores é dividido em 4 atividades: (i) refatorar modelos de requisitos; (ii) gerar modelos arquiteturais; (iii) selecionar uma Solução Arquitetural; e (iv) refinar uma Solução Arquitetural. As regras horizontais são utilizadas na Atividade (i) e as verticais na Atividade (ii). Para executar a Atividade (iii) os RNFs são usados para a seleção de Solução Arquitetural, quando ocorrer mais de uma. E, de acordo com os RNFs utilizados, alguns Padrões Arquiteturais (PAs) podem ser aplicados na atividade (iv). Os PAs a serem utilizados na Atividade (iv) são selecionados a partir de RNFs. O problema é que o STREAM apresenta uma limitação na escolha desses PAs. Não existe uma sistematização nessa escolha. Ela é realizada de forma ad hoc.

Em um processo de construção de AS são levados em consideração tanto os requisitos funcionais quanto os não-funcionais [2]. No entanto, os RNFs normalmente exercem influência sobre todo o sistema [6]. Eles são fundamentais em dois momentos do projeto de AS [1],[6]: (i) seleção de uma AS dentre várias; e (ii) escolha de um ou mais PAs que serão usados no refinamento da AS.

No processo de projeto e construção de um sistema complexo de software, a AS se apresenta como uma questão crítica, já que a qualidade da arquitetura está relacionada com a satisfação ou não dos requisitos chaves requeridos pelo software [4]. Segundo [12], decisões tomadas durante a fase de AS são particularmente significativas uma vez que tem implicações em todo o sistema, especialmente sobre os RNFs. Se de um lado temos os PAs e do outro os RNFs a serem atendidos por uma AS, como relacioná-los de forma a guiar a uma AS de qualidade de acordo com os RNFs? Uma vez que os RNFs foram elicitados e representados no modelo de requisitos, eles precisam ser analisados com o objetivo de se chegar a conclusão de: (i) quais impactam na AS; e (ii) dentre eles, quais são os prioritários.

Neste contexto, este trabalho visa levar em consideração RNFs, que sejam arquiteturalmente significantes, para guiar o refinamento de uma AS. Para isso um processo sistemático foi proposto tendo como entrada modelos de requisitos com RNFs elicitados, e como saída PAs selecionados para o refinamento da AS. Este processo será incorporado ao STREAM com o objetivo de contribuir para solucionar a deficiência existente nesta abordagem.

O artigo está organizado da seguinte forma: na seção 2 serão apresentados os objetivos da pesquisa. Na seção 3 são apresentadas as contribuições científicas realizadas. Na seção 4 são apresentadas as conclusões encontradas. Na seção 5 são apresentados os trabalhos futuros e em andamento.

2 *Objetivos da Pesquisa*

A pesquisa deseja responder a seguinte pergunta: quais PAs são os mais adequados para refinar uma determinada AS afim de que ela atenda a RNFs específicos? A AS é a Solução Arquitetural gerada e escolhida na aplicação do processo STREAM. O seu refinamento é a atividade realizada com o objetivo de fazer com que ela atenda a RNFs específicos. Neste contexto, a pesquisa assumiu como objetivo geral a proposição de um processo, o STREAM-AP, que sistematize a escolha de PAs, a partir de RNFs, para que possam ser usadas no refinamento de uma AS.

No processo de sistematização serão utilizadas contribuições científicas disponíveis nas áreas de RNFs e de PAs. Assim, a ideia principal do trabalho foi reunir em uma única abordagem o necessário para a realização do processo de escolha de PAs e inserí-lo no contexto do STREAM.

Os objetivos específicos deste trabalho são: (i) pesquisar bases de dados de RNFs; (ii) pesquisar bases de dados de PAs; (iii) disponibilizar e utilizar essas bases de dados de forma integrada; (iv) definir as atividades e sub-atividades do processo de sistematização proposto; (iv) aplicar o processo de sistematização proposto em um exemplo.

3 *Contribuições Científicas*

O trabalho proposto surgiu com o objetivo de suprir a deficiência apresentada pela abordagem STREAM no que diz respeito a escolha de PAs a partir de RNFs. Neste sentido, bases de dados de RNFs e de PAs são reunidas e disponibilizadas em um único ambiente. Tanto os RNFs quanto os PAs serão relacionados com tipos de sistemas. E, para contribuir na direção do objetivo principal da nossa proposta, as bases de dados serão relacionadas entre si, de forma a demonstrar o impacto que os PAs exercem sobre os RNFs, ou seja, se um PA impacta positivamente, de forma neutra ou negativamente em um determinado RNF. Essas bases de dados são resultados dos seguintes trabalhos científicos: (i) RNFs e seus atributos [9]; (ii) RNFs por tipos de sistemas [9]; (iii) conflitos entre os RNFs [10]; (iv) PAs por tipos de sistemas [7]; (v) impacto de PAs em RNFs [6]; (vi) pares de PAs de sistemas que utilizam até dois PAs [7]. Elas serão consultadas e/ou atualizadas no decorrer da aplicação da abordagem proposta. A partir dos dados consultados, decisões poderão ser tomadas no que diz respeito a utilização ou não de determinados PAs com o objetivo de se alcançar RNFs específicos.

A partir da incorporação do STREAM-AP no processo STREAM pretende-se tornar a escolha de PAs a partir de RNFs menos empírica e mais científica. A Figura 1 apresenta uma visão geral da abordagem STREAM com o STREAM-AP incorporado a ela. O significado da sigla STREAM-Ap é: STREAM (*Strategy for Transition between REquirements models and Architectural Models*) - AP *(Architectural Pattern*).

A Figura 2 apresenta as principais atividades e artefatos utilizados e gerados pelo STREAM-AP.

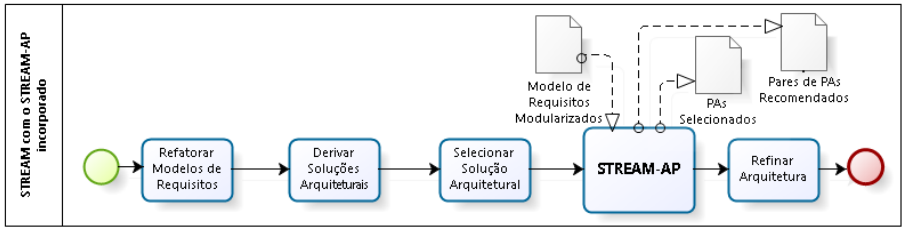


Fig. 1. Visão geral do STREAM com o STREAM-AP incorporado

É importante acrescentarmos que o processo proposto não se restringe e não é dependente da abordagem STREAM. Para que o STREAM-AP seja utilizado é necessário apenas que se tenha um modelo de requisitos no qual esteja disponível os RNFs a serem atendidos.

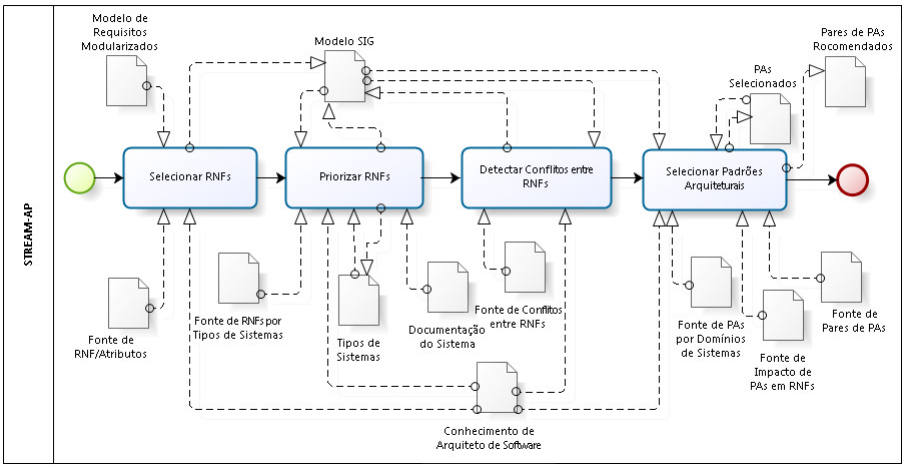


Fig. 2. Visão geral do STREAM-AP

4 *Conclusões*

Neste trabalho nós propomos o STREAM-AP. Trata-se de um processo que objetiva sistematizar a escolha de PAs a partir de RNFs. Esses PAs serão utilizados no refinamento da AS de forma que ela atenda a RNFs específicos.

Algumas abordagens foram propostas com o objetivo de reduzir a lacuna existente ente a ER e a AS, por exemplo [3],[8]. Entretanto, nas nossas pesquisas, percebemos uma deficiência no aprofundamento da escolha de PAs a partir de RNFs. Em específico, a abordagem STREAM [8] realiza a seleção de RNFs e PAs de forma ad hoc.

Com a utilização do STREAM-AP pretende-se aumentar a qualidade da AS e auxiliar o arquiteto de software na difícil tarefa de decisão arquitetural. A ajuda oferecida

a ele/ela é de fundamental importância uma vez que tira dele/dela a responsabilidade única da tomada de decisão baseada apenas em seus conhecimentos, situação em que em muitos momentos ele/ela não tem conhecimento suficiente para sozinho realizar a tomada de decisão.

5 *Trabalhos Futuros e Em Andamento*

O processo de escolha de PAs a partir de RNFs, proposto pelo STREAM-AP, precisa ser aprimorado no que diz respeito a dois aspectos: (i) semi-automatização do processo; e (ii) bases de dados de apoio ao processo.

No que diz respeito a semi-automatização do processo, ela se faz necessária pelo fato da manipulação das bases de dados de forma manual ter se mostrado trabalhosa, uma vez que o volume de dados a ser analisado é considerável e existem vários cruzamentos de dados que necessitam ser realizados. Tal procedimento contribui para a diminuição da carga de trabalho e também da responsabilidade única imposta sobre o arquiteto de software.

A escolha pela semi-automatização ao invés da automatização total foi considerada levando-se em consideração a natureza do processo em questão. Neste tipo de processo é de fundamental importância a intervenção manual do arquiteto de software com o objetivo de ajustes nas decisões realizadas.

A semi-automatização deverá ser realizada através da construção de um software que a partir dos parâmetros de entrada e das informações armazenadas em sua base de dados, gere para o arquiteto de software as informações que ele necessita para a tomada de decisão. No que diz respeito as bases de dados, o arquiteto de software terá a possibilidade de usar os dados armazenados e/ou armazenar novos dados.

O software será constituído de dois módulos: (i) gerenciamento da base de dados; e (ii) execução do processo de seleção dos PAs. Quanto a tecnologia utilizada no desenvolvimento do software, será utilizada a tecnologia Java, mais especificamente do framework para desenvolvimento web JSF. Já no que diz respeito a tecnologia de banco de dados, será usado o SGBD MySql e o framework de persistência de dados Hibernate. A escolha dessas tecnologias são justificadas pela sua aceitação no meio acadêmico e industrial, e por serem adequadas ao tipo de software que se deseja construir. Além disso, todas as tecnologias são *Open Source* e são utilizadas no ambiente de ensino e pesquisa da UFRN. O processo de construção desse software está em fase de análise e em breve será iniciado o processo de desenvolvimento da ferramenta.

Como segundo aspecto temos a questão das bases de dados que auxiliam todo o processo. Elas precisam ser mais enriquecidas no que diz respeito a quantidade e aos tipos de dados disponibilizados: (i) disponibilizar maior quantidade de RNFs e seus respectivos atributos; (ii) disponibilizar a relação entre os atributos de RNFs e os tipos de sistemas, e não apenas por RNFs; (iii) disponibilizar a relação entre os atributos de RNFs e os PAs, e não apenas por RNFs. O objetivo do enriquecimento das bases de dados é proporcionar uma análise/escolha mais refinada dos PAs. Para que isso ocorra, é de fundamental importância que outras fontes de informações, da academia e/ou da indústria, relacionadas a RNFs, conflitos entre RNFs, PAs, impacto de PAs em

RNFs e a relação entre eles, sejam consultadas de forma a enriquecer e dar um maior embasamento a este trabalho de pesquisa.

Referências

1. Castro, J., Lucena, M., Silva, C. T., Alencar, F. M., Santos, E., & Pimentel, J.: Changing attitudes towards the generation of architectural models. Journal of Systems and Software , 85 (3), 463-479 (2012)
2. Chung, L., Gross, D., & Yu, E. S.: Architectural Design to Meet Stakeholder Requirements. WICSA, (pp. 545-564) (1999)
3. Chung, L., Supakkul, S., Subramanian, N., Garrido, J. L., Noguera, M., Hurtado, M. V., et al.: Goal-Oriented Software Architecting. In: Relating Software Requirements and Architectures (pp. 91-109) (2011)
4. Garlan, D.: Software architecture: a roadmap. Proceedings of the Conference on The Future of Software Engineering (pp. 91-101). New York, NY, USA: ACM (2000)
5. Hall, J. G., Grundy, J., Mistrik, I., Lago, P., & Avgeriou, P.: Introduction: Relating Requirements and Architectures. In: Relating Software Requirements and Architectures (pp. 1-9) (2011)
6. Harrison, N. B., Avgeriou, P.: Leveraging Architecture Patterns to Satisfy Quality Attributes. ECSA, (pp. 263-270) (2007)
7. Harrison, Neil B.; Avgeriou, Paris. Analysis of architecture pattern usage in legacy system architecture documentation. In: Software Architecture, 2008. WICSA 2008. Seventh Working IEEE/IFIP Conference on. IEEE, 2008. p. 147-156. (2008)
8. Lucena, M., Castro, J., Silva, C. T., Alencar, F. M., & Santos, E.: Stream: a strategy for transition between requirements models and architectural models. SAC, (pp. 699-704) (2011)
9. Mairiza, D., Zowghi, D., & Nurmuliani, N.: An investigation into the notion of non-functional requirements. SAC, (pp. 311-317) (2010)
10. Mairiza, D., Zowghi, D., & Nurmuliani, N: Towards a Catalogue of Conflicts Among Non-functional Requirements. ENASE 2010: 20-29 (2010)
11. Kleppe, A., Warmer, J., & Bast, W. MDA Explained: The Model Driven Architecture. Prentice Hall. (2003).

Harrison, N. B., Avgeriou, P., & Zdun, U. Using Patterns to Capture Architectural Decisions. IEEE Software , 24 (4), 38-45. (2007)

Automatic Models Transformation for the STREAM process

Monique Soares[1], João Pimentel[1], Carla Silva[1], Gabriela Guedes[1], Cleice Talitha[1], Fernanda Alencar[2], Jaelson Castro[1], Jéssyka Vilela[1]
[1]Universidade Federal de Pernambuco – UFPE, Centro de Informática, Recife, Brasil
{mcs4, jhp, ctlls, ggs, ctsn, jbc, jffv}@cin.ufpe.br
[2]Universidade Federal de Pernambuco – UFPE, Departamento de Eletrônica e Sistemas, Recife, Brasil fernandaalenc@gmail.com

Abstract. The STREAM (Strategy for Transition Between Requirements and Architectural Models) process allows systematic generation of initial architectural design models from oriented goals requirements models through an informal definition of model transformation rules. It was observed that the first two activities in this process are time-consuming and error-prone because they involve model transformation rules that are not automated. This article advocates the automation of such transformation rules, with the intention of improving the productivity of the process and the quality of the models produced.

Keyword: Requirements Engineering, Software Architecture, Transformation Rules, Automation.

1 Introduction

The STREAM process (Strategy for Transition Between Requirements and Architectural Models) [2] is a systematic approach to integrate the activities of requirements engineering and architectural design, based on model transformations, to generate architectural models from requirements models.

To transform the requirements models in architecture models the STREAM uses i* (iStar) [3] as source language and Acme [4] as target language. This approach has four activities, namely: Prepare Requirements Models, Generate Architectural Solutions, Select Architectural Solution and Refine Architecture. The first two activities need attention and take a lot of time to be done. These activities define horizontal and vertical transformation rules (HTRs and VTRs), respectively, which are amenable to automation are implemented. Once automated, the first two STREAM activities would not require as much time and attention to be performed.

Currently, the transformation rules are made manually, demanding attention from the analyst. Our proposal is to use a transformation language to describe these rules and execute them with a supporting tool.

The i* language allows goal-oriented modeling [3], it is able to represent the organization and characteristics of the system being developed. Stakeholders and

systems are represented as actors. To achieve its goals, the actors depend on each other. The i* language is composed of two models: a SD (Strategic Dependency) model, that represents dependency relationships between actors, and a SR (Strategic Rationale) model, that details how actors achieve their goals and dependencies.

In a dependency, a *depender* actor depends on another actor (*dependee*) to achieve some objective (*dependum*). A *dependum* can be a goal, a softgoal, resource or task.

Acme is an architectural description language [4] designed to describe the view of the components and connectors of the system architecture. It has six main types of entities for representing architecture: Components, Connectors, Systems, Ports, Roles and Representations.

Components represent the primary computational elements and storage of data from one system. Connectors characterize interactions between the components. Systems denote configurations of components and connectors. Each port identifies a point of interaction between the component and its environment. Roles define the connectors' interfaces. Representation describes support hierarchical architectures. Among these six types, the most basic elements of architectural description are components, connectors and systems.

2 Research Objectives

This paper aims to answer the following research question:

Is it possible to automate the transformation rules defined in the first two STREAM activities? And, if possible, how to automate these rules?

The main objective of this work is to automate the transformation rules defined in STREAM. In order to accomplish this, we have established the following strategy:

- Define the transformation rules in a transformation language;
- Make the vertical and horizontal transformation rules compatible with iStarTool [6];
- Make the vertical transformation rules consistent with AcmeStudio.

3 Scientific Contributions

The automation of the horizontal and vertical transformation rules proposed by STREAM requires a language that allows mapping elements and processing them. To do this, we have chosen the QVTO (Query / View / Transformation Operational) [5] language, which is a transformation language that has integration with Eclipse and whose community gives constant maintenance and support.

To facilitate the development of artifacts used as input in the first activity of STREAM, we have chosen iStarTool tool [6], which allows i* modeling. It is developed under a model-driven technology, the GMF (Graphical Modeling Framework)
[7] Eclipse. The iStarTool uses XMI files to save its models, based on its own metamodel. The XMI file generated by iStarTool is compatible with the QVTO plugin in Eclipse.

The input artifacts of the first STREAM activity are created in iStarTool and the transformation of these artifacts is executed by the QVTO Eclipse plugin. This first activity is concerned with improving the modularity of the expanded system actor and it is subdivided into three activities: analysis of internal elements, application of horizontal transformation rules and evaluation of i* models.

In order to develop these activities it is necessary to use:

- Heuristics to guide the decomposition of the system actor;
- A set of rules for transforming i* models;
- Metrics to measure the modularization degree of initial and final i* models.

There are four HTRs (Horizontal Transformation Rules). **Table 1** presents application examples of each horizontal rule, showing the original model and the resulting model after applying the rule, in order to representate how each rule works.

HTR1 moves a previously selected sub-graph, according to the heuristics. The analyst may choose to move this sub-graph for a new actor or an existent actor. HTR1 was not automated because it depends on the choice of the analyst, so it is necessary to apply this rule manually. The analyst uses the heuristics and performs the HTR1 rule.

After applying HTR1, the resulting model may not be correct according to the i* language syntax. Therefore, we must check the preconditions of the others rules, HTR2, HTR3 and HTR4.

The HTR2 moves a means-end relationship across the actor boundary. HTR2 considers the situation where the sub-graph to be moved has the root element as a "means" of a means-end relationship.

The HTR3 moves a contribution relationship across the actor boundary. The HTR3 considers the situation in which the sub-graph to be moved has a contribution relationship with other elements that cannot be moved.

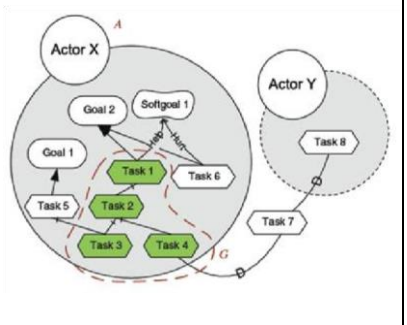
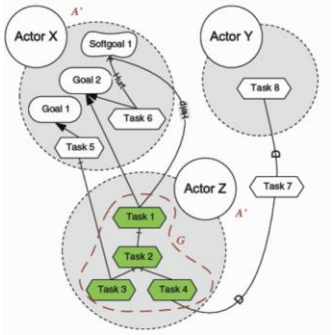
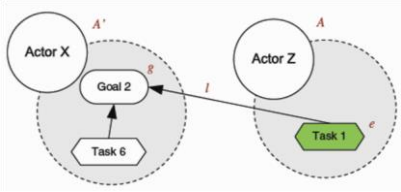
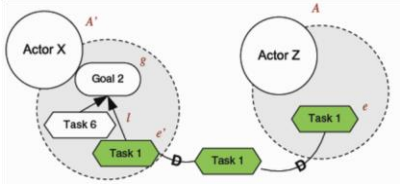
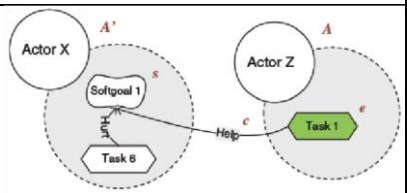
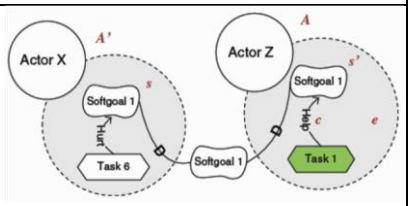
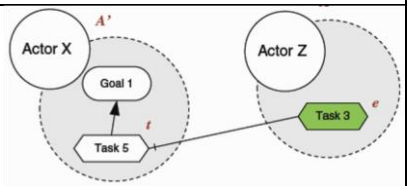
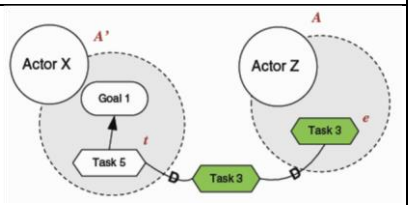
The HTR4 moves a task decomposition relationship across the actor boundary. HTR4 considers the situation where the sub-graph to be moved has a task decomposition relationship with other elements that cannot be moved.

The transformation rules are intended to delegate internal elements of software actor to other actors. This delegation must ensure that new actors and the original actor has a dependency relationship. Thus, the original model and the final model are supposed to be semantically equivalent.

Upon completion of this activity, the actors representing the software are easier to understand and maintain, since there are less internal elements. The original requirements model is decomposed into a more modularized one.

Table 1. Application example of HTRs adapted from [8]

Rule	Original Model	Resulting model after applying the rule

HTR1		
HTR2		
HTR3		
HTR4		

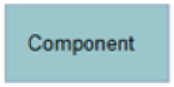
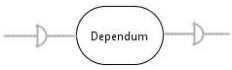
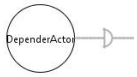
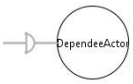
The second activity of STREAM process (Derive Architectural Solutions) is concerned with the mapping of i* actors and dependencies into Acme elements through the application of Vertical Transformation Rules(VTRs).

As the vertical transformation rules do not consider the internal elements of the actors, first we create a SD model from the modularized i* model, produced on the first activity.

The first rule (VTR1) maps i* actors into Acme components, while VTR2 transforms i* dependencies in Acme connectors. The VTR3 maps *depender* actors as required port of Acme connector. Finally, the VTR4 maps *dependee* actors as provided port of Acme connector.

Table 2 shows the VTRs.

Table 2. Vertical Transformation Rules

	Fonte (*i**)	Alvo (Acme)
VTR1		
VTR2		
VTR3		
VTR4		

For the construction of i* artifacts we used iStarTool and the rules were described in QVT. It was necessary to make the vertical rules consistent with iStarTool and AcmeStudio tools, to make it possible, we used the i* metamodel present iStarTool and the Acme metamodel designed according to the AcmeStudio sintax to describe and run the rules.

To apply automated rules you need to perform these steps [9]:

1. Create the i* model of the system using iStarTool;
2. Use three heuristics for selecting the subset of candidate elements for refactoring (modularization);
3. Apply HTR1 manually, i.e., move the subset of candidate elements to another actors, in order to modularize the i* model;
4. Use the SR model obtained with HTR1 as input to the application of other horizontal transformations rules, which are already automated;
5. Transform the modularized i* SR model, resulting from the application of horizontal transformation rules, into an SD model;
6. Use the SD model as input to the application of vertical transformation rules, which results in an initial architectural model in Acme.

Of the four horizontal transformation rules, three were automated (HTR2, HTR3 and HTR4). The vertical transformation rules were divided into four rules for better understanding and all of them were automated.

4 Conclusions

The main objective of this work was to automate the horizontal and vertical transformation rules proposed by the activities 1) Prepare Requirements Models and 2) Generate Architectural Solutions of the STREAM process.

We used the iStarTool tool to create the input artifact for performing the horizontal rules. This artifact is the i* SR model obtained with manual execution of HTR1 applied according to the choice of the analyst.

The transformation rules proposed by STREAM were described in QVT, a specification language for processing models, for the automation and execution of the

transformation rules. The Eclipse environment was used to make possible the implementation of the rules in QVT language.

We applied the automated rules in three software projects to exemplify their use. More details are presented in [1].

5 Ongoing and Future Works

Currently, the output of our process is a XMI file with an architectural model in Acme. However, the AcmeStudio tool is able to read only files described using the Acme textual language. Therefore, it is not possible to display the architectural model graphically. So, our plan is to provide another set of transformation rules for generating the textual file also, thus facilitating the graphical visualization in AcmeStudio tool. It is also necessary to include the automated rules on iStarTool, to facilitate the execution of the rules in the same tool, the modeling and the transformation will occurs in the iStarTool. And last, execute the rules on various application examples with the intention of evaluating the real benefits and limitations of approach.

References

1. Soares, M. C. Automatization of the Transformation Rules on the STREAM process (In Portuguese: Automatização das Regras de Transformação do Processo STREAM). Dissertation (M.Sc.). Center of Informatics: UFPE, Brazil, 2012, available in: http://www.cin.ufpe.br/~mcs4/dissertation_mcs4.pdf.
2. Lucena, M. STREAM: A Systematic Process to Derive Architectural Models from Requirements Models. Thesis (PhD). Center of Informatics: UFPE, Brazil, 2010
3. Yu, E. Modelling Strategic Relationships for Process Reengineering. Thesis (PhD). University of Toronto: Department of Computer Science, 1995.
4. Acme. Acme - The Acme Architectural Description Language and Design Environment., 2011. Available in: <http://www.cs.cmu.edu/~acme/index.html>. Acessed: March 2013.
5. OMG. QVT 1.1. Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification, January 2011. Available in: <http://www.omg.org/spec/QVT/1.1/>. Acessed: March 2013.
6. Malta, A., Soares, M., Santos, E., Paes, J., Alencar, F., Castro, J. (2011). iStarTool: Modeling requirements using the i* framework. IStar 11.
7. ECLIPSE GMF. GMF - Graphical Modelling Framework. Available in: <http://www.eclipse.org/modeling/gmf/ >. Acessed: March 2013.
8. Pimentel, J., Lucena, M., Castro, J., Silva, C., Santos, E., Alencar, F. (2011). Deriving software architectural models from requirements models for adaptative systems: the STREAM-A approach. Requirements Engineering Journal.
9. Soares, M. C., Pimentel, J., Castro, J., Silva, C., Souza, C. T., Guedes, G. S., Dermeval, D. (2012). Automatic Generation of Architectural Models From Goals Models. SEKE 2012: 444-447.

Integrando Requisitos Ágeis com Modelos i*

Aline Jaqueira, Bernardo Gurgel, Márcia Lucena

Departamento de Informática e Matemática Aplicada – UFRN
alinejaqueira@ppgsc.ufrn.br, bernardogfilho@gmail.com,
marciaj@dimap.ufrn.br

Resumo. Histórias de usuário são um artefato de uso limitado. O contexto geral do sistema, seus objetivos, as relações de dependência dos requisitos são muitas vezes omitidos ou mal entendidos com o uso somente desse artefato. Este trabalho apresenta uma abordagem para complementar os requisitos ágeis (histórias de usuário) através de sua integração com modelos i*. As histórias de usuário são mapeadas para modelos i* e, por meio da visualização dos modelos, busca-se contribuir dando suporte aos *stakeholders* através da representação visual e mais abrangente dos requisitos. Além disso, as histórias de usuário poderão ser analisadas de acordo com a carga semântica da técnica i*. Ao utilizar os modelos, a característica de agilidade dos projetos poderá ser mantida pelo fato da visualização através de modelos ser mais simples e rápida.

1 Introdução

Nos métodos ágeis os requisitos são desenvolvidos de forma incremental, de acordo com as prioridades do cliente. A elicitação é realizada com clientes que fazem parte da equipe de desenvolvimento. Os clientes escrevem histórias do que o sistema precisa fazer (história de usuário) e priorizam as mesmas de acordo com o valor do seu negócio. Essas histórias são os artefatos de saída da atividade de elicitação dos requisitos nos métodos ágeis.

Uma história de usuário descreve a funcionalidade que tem valor para o cliente do software e é usada para planejamento do projeto, funcionando como um lembrete para a equipe já que conversas posteriores sobre a história são essenciais para transmitir os detalhes das mesmas [3]. Nos métodos ágeis as histórias de usuário são usadas por serem consideradas uma abordagem flexível, de baixa sobrecarga e centrada no usuário [1]. São amplamente utilizadas pelo desenvolvimento ágil [2] e são consideradas neste trabalho como um artefato de requisito ágil.

As histórias de usuário geralmente são escritas em linguagem natural. Um formato para a escrita das histórias vem sendo bastante utilizado: "como <papel>, eu quero <ação> para <meta>" [3]. Portanto, para aplicação das histórias de usuário neste trabalho, parte-se do princípio que as mesmas utilizam esse formato.

As histórias de usuário constituem artefatos muito restritos para contemplar determinadas questões como o entendimento global do sistema e as relações de dependências entre as funcionalidades [4]. Algumas limitações desse artefato [5]: (i) dependências entre as histórias são geralmente "escondidas"; (ii) o entendimento do contex-

to social no qual o sistema a ser desenvolvido está inserido não está, nem pode ser contido numa história de usuário; (iii) interpretar alguns detalhes e implicações do sistema torna-se um processo difícil a partir somente das histórias de usuário. Essas limitações sobrecarregam a equipe ao exigirem rigorosas operações mentais.

Além disso, contar com a participação do cliente comunicando os requisitos através das histórias, das conversas e dos testes de aceitação é um desafio e algumas decisões se tornam arriscadas principalmente para sistemas complexos. Para a equipe, torna-se muito difícil controlar o sistema unicamente com histórias de usuário [6].

2 Objetivos da Pesquisa

Este trabalho propõe uma abordagem que visa melhorar alguns desafios citados para as histórias de usuário. Busca-se dar suporte aos *stakeholders* fornecendo uma visão gráfica e abrangente das histórias de usuário e dos seus relacionamentos com o sistema através de modelos i*.

Tanto os métodos ágeis quanto a técnica i* ressaltam os *stakeholders* no ambiente de desenvolvimento, o que justifica a escolha da técnica i* para representar as histórias de usuário[13]. O foco nos *stakeholders* e seus relacionamentos para descrição de requisitos é uma característica da técnica i*, onde os atores dependem uns dos outros para atingir suas metas. Os métodos ágeis também se concentram em fatores humanos e reconhecem as pessoas como peças fundamentais e como principais impulsionadores do sucesso do projeto [7].

Portanto, neste trabalho, os conceitos das histórias de usuário são empregados juntamente com os conceitos da técnica i* [8]. O papel das histórias de usuário são mapeados como atores nos modelos i*, as metas das histórias de usuário são mapeadas como metas nos modelos i* e as ações das histórias de usuário são mapeadas como tarefas do Ator Sistema, pois as mesmas serão operacionalizadas pelo sistema. Foram utilizados conceitos e notações da técnica i* de sua versão simplificada [9] e somente alguns elementos da técnica i* são utilizados de acordo com a necessidade de mapeamento deste trabalho. Dessa forma, na técnica i*, para construir o modelo SD (*Strategic Dependency*), os elementos utilizados são o *ator*, as *metas* e a *associação IS_A*. Para o modelo SR (*Strategic Rationale*) são utilizados os elementos *tarefas*, *recursos* e a ligação de *decomposição*.

Ao utilizar os modelos da técnica i* a partir das histórias de usuário, a característica de agilidade dos projetos poderá ser mantida pelo fato da visualização através dos modelos ser mais simples e rápida. Modelos proporcionam agrupamentos visuais que permitem analisar rapidamente grandes quantidades de informações, além de ajudarem a organizar e apresentar as informações e darem contexto aos detalhes [6]. Uma vez que os modelos i* são representações gráficas dos requisitos, ter-se-á uma maneira abrangente e visual de enxergar as histórias de usuário, melhorando a visualização do contexto do sistema, facilitando o acesso aos requisitos, contribuindo para a tomada de decisão no ambiente de desenvolvimento.

Para simplificar o entendimento, foram estabelecidas heurísticas como recurso para se chegar à solução do mapeamento [13]. Vale ressaltar que as heurísticas esta-

belecidas devem ser executadas na ordem em que foram expressas para que o mapeamento ocorra de maneira mais correta e objetiva.

Heurísticas para mapear as histórias de usuário para o modelo SD:

SD-H1: Criar o Ator Sistema;

SD-H2: Criar um Ator no modelo i* para cada diferente papel das histórias de usuário;

SD-H3: Criar uma meta no modelo i* para cada meta das histórias de usuário. Se houver metas repetidas as mesmas serão definidas uma única vez no modelo;

SD-H4: Se houver metas repetidas para atores diferentes, criar um Ator genérico;

SD-H4.1: Criar um relacionamento IS_A do Ator genérico para os demais atores específicos que compartilham a mesma meta;

SD-H5: Relacionar as dependências de cada ator com suas metas.

Heurísticas para mapear as histórias de usuário para o modelo SR:

SR-H1: Criar uma Tarefa dentro do Ator Sistema para cada ação das histórias de usuário;

SR-H2: Se houver ações diferentes para a mesma meta, criar uma tarefa genérica;

SR-H2.1: Decompor a tarefa genérica em sub tarefas que representem as ações associadas à mesma meta;

SR-H3: Relacionar as dependências de cada meta com as tarefas correspondentes de acordo com as histórias de usuário.

SR-H4: Se houver Tarefas que dependem do próprio Ator a que estão relacionadas, gerar um recurso com o nome da tarefa;

SR-H5: Relacionar o recurso criado dependendo do Ator.

Para demonstrar uma aplicação da abordagem, utilizou-se como exemplo um sistema de *login* simples [10] considerando a perspectiva de um usuário e de um administrador. A tabela 1 apresenta as histórias de usuário do sistema de *login*.

De acordo com as heurísticas propostas, o modelo SD será mapeado primeiramente criando o Ator Sistema. Posteriormente, cria-se os atores para os papéis das histórias de usuário, que, nesse caso, são ator Usuário e ator Administrador. As metas das histórias de usuário são criadas como metas no modelo SD e ligadas como dependências saindo dos atores a que estão associadas e chegando ao Ator Sistema.

Table 1. Histórias de Usuário do Sistema de Login (Fonte: IBM [10])

	Papel	**Ação**	**Meta**
1	Usuário	Ter nome de usuário	Acessar conteúdo seguro
2	Usuário	Ter senha	Acessar conteúdo seguro
3	Usuário	Escolher nome de usuário	Personalizar a conta
4	Usuário	Alterar senha padrão	Personalizar senha
5	Administrador	Atribuir senha ao usuário	Registro ser automatizado
6	Administrador	Enviar email de registro	Confirmar ativação da conta no email
7	Administrador	Solicitar login ao usuário	Garantir segurança do

			conteúdo
8	Usuário	Cadastrar lembrete de senha	Lembrar a senha
9	Administrador	Solicitar lembrete de senha	Confirmar usuário

Para gerar o modelo SR, cada ação das histórias de usuário foi gerada como tarefa dentro do Ator Sistema, pois o mesmo é que irá operacionalizá-la, realizando a tarefa de maneira particular para atender às metas dos atores. Como existem, nesse exemplo, ações diferentes para a mesma meta, foi criada uma tarefa genérica no modelo SR que foi decomposta nas ações em forma de sub-tarefas. As tarefas que dependem do próprio ator geraram um recurso que depende do ator com o nome da tarefa. A figura 1 mostra o resultado do mapeamento.

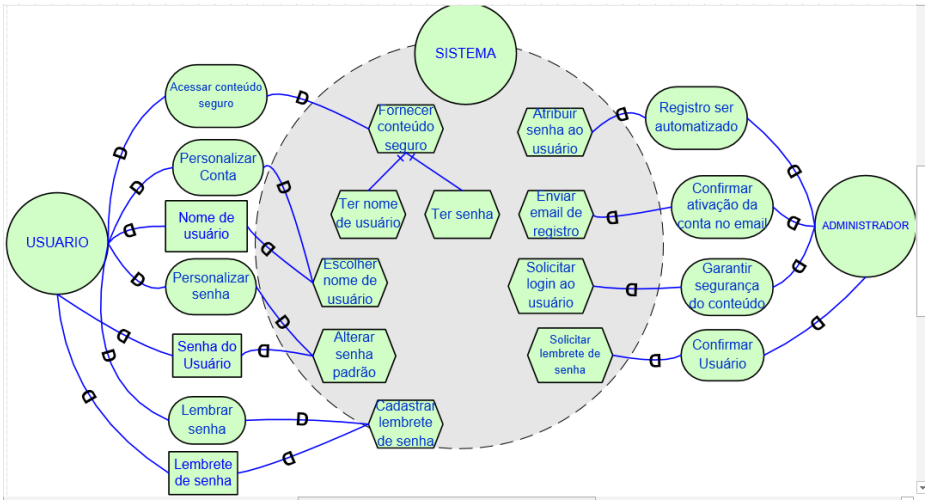


Fig. 1. Modelo SR do Sistema de *Login*

3 Contribuição Científica

Mesmo em um ambiente ágil é necessário desenvolver algum modelo antes da implementação para garantir uma compreensão compartilhada pela equipe, de modo que fique sincronizada com os objetivos do negócio, valor e contexto do projeto [6]. Modelos visuais auxiliam nesse entendimento e no entendimento de como os usuários precisarão usar o sistema. São eficazes para os *stakeholders* entenderem a solução proposta e também para mantê-los interessados e envolvidos e até mesmo para deixar claro o que a solução não vai entregar.

A contribuição deste trabalho é a elaboração de uma abordagem que utiliza modelos visuais propiciados pela técnica i* para amenizar algumas limitações das histórias de usuário. Além disso, essa abordagem poderá funcionar como uma forma de documentação visual dos requisitos do software, contribuindo assim para amenizar também o desafio da falta de documentação no ambiente de desenvolvimento ágil [12].

Outras contribuições podem ser citadas como a melhoria no entendimento do contexto do sistema a ser desenvolvido, o uso e o acesso mais fácil à informação das histórias de usuário a partir da visualização do modelo visual, melhoria do processo de tomada de decisão de acordo com a análise das histórias, uma vez que elas estão descritas em modelos i*.

Além disso, para aplicar a abordagem deste trabalho, está em desenvolvimento a ferramenta *StoryTeller*, cujo objetivo é fornecer um meio para armazenar e gerenciar as histórias de usuário dos projetos de software. A ferramenta está sendo desenvolvida como um projeto *open source* de aplicativo *web*. O *link* para o repositório no *Github* é https://github.com/bernardogfilho/storyteller e o *link* para onde a ferramenta se encontra hospedada no *Heroku* (http://storyteller-beta.herokuapp.com/).

Na *StoryTeller*, os usuários poderão se registrar e criar projetos vinculados à sua conta além de cadastrar as histórias de usuário dos projetos. Assim, será possível realizar operações básicas como editar ou excluir as histórias de um projeto. Será possível também pesquisar por histórias que possuam o mesmo ator ou a mesma meta. Além de fornecer uma base para organizar e gerenciar as histórias de usuário, a ferramenta será estendida para automatizar a abordagem deste trabalho, ou seja, mapear as histórias de usuário para os modelos i*, a fim de fornecer o modelo visual de maneira automatizada. Pelo fato de a ferramenta tratar as histórias de usuário de modo independente e flexível, as mesmas se tornam fáceis de manipular, sendo possível exportá-las de diversas maneiras e relações através de um arquivo .xml ou .json, por exemplo.

4 Conclusão

Uma abordagem baseada em modelos visuais pode fornecer ligações mais diretas e rastreáveis para o desenvolvimento do sistema, promovendo uma análise com maior impacto na concepção e implementação do software. Funciona também para facilitar a comunicação, compreensão, a detecção de problemas ou explorar cenários hipotéticos e potenciais soluções. Ao visualizar modelos, possíveis erros e/ou negligências são reconhecidos mais facilmente [11]. Tudo isso facilita o processo de análise e discussão a respeito do sistema a ser desenvolvido.

A partir do mapeamento das histórias de usuário para os modelos SD e SR da técnica i*, é possível organizar e representar todas as histórias em um modelo que fornece uma visualização geral das histórias e seus relacionamentos. Além disso, todas as histórias de um mesmo ator são apresentadas no mesmo modelo, permitindo encontrá-las com maior facilidade. Dessa maneira, é possível entender melhor o contexto do sistema, seus principais atores e suas metas.

A visualização através dos modelos torna mais fácil a identificação de dependências de cada ator com o Sistema bem como a identificação de tarefas do Sistema para atender a cada ator específico envolvido com o software.

Portanto, a abordagem deste trabalho possibilita melhor análise, comunicação, discussão e melhor entendimento do sistema a ser desenvolvido.

5 Trabalhos Futuros

Com o objetivo de dar continuidade à pesquisa desenvolvida por este trabalho algumas sugestões de trabalhos futuros podem ser citadas: (i) o desenvolvimento de diretrizes ou uma ferramenta para realizar a transformação das histórias de usuário para o formato utilizado neste trabalho, de modo a poder utilizar esta abordagem para todas as histórias de usuário; (ii) o desenvolvimento de diretrizes para realizar o mapeamento "de volta", dos modelos i* para as histórias de usuário; (iii) o tratamento da escalabilidade para o ator Sistema; (iv) a possibilidade de representar as tarefas nos demais atores do modelo; (v) a possibilidade de representar o relacionamento entre os demais atores no modelo.

Referencias

1. O'hEocha, C., Conboy, K.: The Role of the User Story Agile Practice in Innovation. In: P. Abrahamsson, N. Oza (Eds.), Lean Enterprise Software and Systems (pp. 20-30). Springer, New York (2010)
2. Cockburn, A.: Agile Software Development: The Cooperative Game. Pearson, Boston (2007)
3. Cohn, M.: Agile Estimating and Planning. Prentice Hall, Massachusetts (2006)
4. Sharp, H., Robinson, H., Segal, J., Furniss, D.: The Role of Story Cards and the Wall in XP Teams: a Distributed Cognition Perspective. Proceedings of Agile 2006, IEEE Computer Society Press, pp. 65-75. Washington (2006)
5. Sharp, H., Robinson, H., Petre, M.: The Role of Physical Artefacts in Agile Software Development: Two Complementary Perspectives. Interacting with Computers 21 pp. 108–116. New York (2009)
6. Beatty, J., Chen, A.: Visual Models for Software Requirements. Microsoft Press , Washington (2012)
7. Highsmith, J.,Cockburn, A.: Agile Software Development: The Business of Innovation. IEEE Computer, vol. 34, pp. 120-122 (2001)
8. Yu, E.: Modelling Strategic Relationships for Process Reengineering. PhD Thesis. University of Toronto, Department of Computer Science (1995)
9. i* Wiki Home, http://istar.rwth-aachen.de/tiki-index.php
10. IBM Data sets : Example of User Stories, http://www-958.ibm.com/software/analytics/manyeyes/datasets/example-of-user-stories/versions/1
11. Horkoff, J., Yu, E.: A Qualitative, Interactive Evaluation Procedure for Goal- and Agent-Oriented Models. In: CAiSE Forum. CEUR Workshop Proceedings (2009).
12. Jaqueira, A., Andreotti, E., Lucena, M., Aranha, E.: Desafios de Requisitos em Métodos Ágeis: Uma Revisão Sistemática. In: 3rd Brazilian Workshop on Agile Methods, São Paulo (2012)
13. Jaqueira, A.: Uso de Modelos i* para Enriquecer Requisitos em Métodos Ágeis. Dissertação de Mestrado. Departamento de Informática e Matemática Aplicada – UFRN. Natal (2013)

Utilizando sistemas de conhecimento para a identificação de metas flexíveis em uma linguagem de domínio

Antônio de Pádua A Oliveira[1], José Luis Braga[2], Cristiane Aparecida Lana[2], e Lucas Gonçalves Cunha[2]

[1]Departamento de Informática e Ciências da Computação, UERJ, Rio de Janeiro, RJ
padua@ime.uerj.br
[2]Departamento de Informática, UFV, PPGCC – Programa de Pós-Graduação em Ciência da Computação, Viçosa, MG
zeluis@dpi.ufv.br, {cristiane.lana, lucas.cunha}@ufv.br

Resumo: A utilização de sistemas de conhecimento em problemas que não são totalmente conhecidos ou bem definidos permite entender esses problemas de maneira exploratória, simulando situações e soluções utilizando técnicas de inteligência artificial. A abordagem inicial utilizada nos projetos de pesquisa do grupo tinha como foco apenas verificar a presença de atributos de transparência em modelos de requisitos de software modelados com iStar. Diante dos resultados percebidos em conjunto com a evolução dos objetivos de pesquisa, e ajustando o foco para identificar a presença de metas flexíveis (softgoals) de outras naturezas a partir da parametrização em bases de conhecimento de sistemas construídos com a ferramenta CLIPS, foi possível ampliar os objetivos dos projetos. Foi incluído o objetivo de encontrar candidatos a metas flexíveis partindo da identificação de sinônimos de termos indicadores das mesmas em textos da linguagem do domínio registrados no LAL – Léxico Ampliado da Linguagem. Este avanço foi possível com a utilização de sistemas com base de conhecimento desenvolvidos no ambiente CLIPS, definindo regras de produção adequadas a partir das características de metas flexíveis como percebidas pelo NFR Framework. Resultados iniciais utilizando a abordagem permitiram melhorar a identificação de requisitos de transparência pela inclusão do uso de sinônimos destes requisitos.

Keywords: Knowledge-based systems; iStar; softgoals; non-functional requirements

1 Introdução

Ainda não é possível, dado o estado atual de conhecimento em engenharia de requisitos, associar completa e precisamente com cada domínio de conhecimento ou descrição de problemas de cada domínio os requisitos não-funcionais (NFRs) que devem ser atendidos na implementação de um sistema de software. O assunto ainda está em exploração e desenvolvimento e técnicas de implementação que permitam simular e experimentar atributos, regras e exigências particulares devem ser usadas, permitindo aos pesquisadores focar em aspectos que interessam aos domínios em estudo, tirando

o foco de aspectos irrelevantes de implementação, uso de linguagem de programação e ambientes de desenvolvimento.

Algumas técnicas de inteligência artificial, em particular as que se referem a sistemas baseados em conhecimento, são úteis como ferramentas auxiliares em uma investigação inicial, permitindo flexibilidade e facilidade de experimentação além de fornecerem um pouco de "requirements expertise" ao engenheiro de requisitos com pouca prática ou experiência em uma área organizacional.

2 Objetivos da Pesquisa

O objetivo da nossa linha de pesquisa é explorar melhor o potencial de sistemas com base de conhecimento, utilizados anteriormente em um problema recorrente da fase de requisitos de sistemas de software. O objetivo principal pode ser resumido como: *"Suprir o engenheiro de requisitos com uma ferramenta auxiliar que possa acrescentar ao seu conhecimento o expertise de uma técnica para a elicitação de metas flexíveis"*.

Nossa pesquisa é motivada pela necessidade de lidar com metas devido aos problemas em sistemas (exemplo: serviço de ambulâncias da cidade de Londres [10]) decorrente da pouca importância dada à elicitação de NFRs desde as fases iniciais do desenvolvimento do software. Diversas publicações já relataram problemas de fracassos em sistemas de software associados por não ser dada a merecida importância aos NFRs pelos engenheiros de software.

Devido ao desafio do problema, premissas estão sendo assumidas em nossa linha de pesquisa, e estas podem ser descritas pelos seguintes pontos que tornam mais evidentes os benefícios da nossa abordagem:

- descrições de problemas carregam, por natureza, ambiguidades inerentes à linguagem utilizada no texto [1]. Alguns requisitos são mencionados explicitamente, mas talvez requisitos importantes estejam escondidos nas intenções implícitas de quem escreveu o texto, e devem ser extraídos utilizando técnicas apropriadas de elicitação;
- o uso de sinônimos dos termos que representam requisitos não-funcionais pode ser uma técnica que permita enxergar as intenções implícitas nos textos, facilitando a sua extração e anotação de requisitos candidatos a partir da análise dos mesmos feita por máquina ou por engenheiros de requisitos.

Sistemas baseados em conhecimento são ferramentas adequadas para ajudar a perceber a presença de requisitos funcionais ou não-funcionais em descrições de problemas ou modelos de requisitos. Neste projeto a seguinte estrutura é utilizada inicialmente:

- base de dados contendo termos principais indicativos de requisitos e seus sinônimos;
- representação de modelos de domínio usando o framework de descrições do Léxico Ampliado da Linguagem, já filtradas e em formato processável por máquina;
- base de conhecimento composta por regras de produção que serão usadas para análise de descrições do LAL, na busca por requisitos referenciados explicitamente ou extraídos pelo uso de sinônimos.

Para contextualizar a abordagem de nosso projeto de pesquisa colocamos a seguir um resumo das técnicas e dos conceitos que estamos utilizando para a focalização do nosso esforço de pesquisa.

2.1 Framework de Modelagem i*

O Framework de Modelagem i* (i-estrela) [2] modela contextos organizacionais com base nos relacionamentos de dependência entre os atores desse contexto e, principalmente, fazendo a representação, em diagramas i-estrela, das metas que os atores precisam que sejam atingidas. Para o i-estrela: *A goal is a condition or state of affairs in the world that an actor would like to achieve* [2]. Na representação do i-estrela, atores dependem uns dos outros para que metas sejam alcançadas, recursos sejam fornecidos, tarefas sejam realizadas e metas flexíveis (softgoals) sejam "razoavelmente satisfeitas". O i-estrela utiliza o mesmo conceito usado por Chung [3] de que uma meta flexível (softgoal) possui uma característica de incerteza para a satisfação ou não da meta flexível. Essa incerteza inerente ao problema é que o torna adequado ao tratamento com sistemas de conhecimento.

2.2 Os Conceitos de Ação Concreta e de Ação Flexível

Para o i-estrela, tarefas (tasks) são os principais meios ("means") para que metas ("ends") sejam atingidas. Por outro lado, pode-se identificar que existem dois tipos de tarefas ou ações, como a definida por Sá Carvalho [4] que escreveu que uma ação concreta é uma ação executada em um contexto organizacional que pode ser descrita por um verbo ativo concreto, bem definido (exemplos: comprar, vender, cobrar, multar, produzir, treinar, alocar, etc.). Para este autor, uma ação concreta não pode ser descrita por um verbo pouco preciso (como controlar, apurar, administrar, acompanhar, etc.). As tarefas ou ações, por exclusão a definição anterior, podem ter a qualificação de flexível [5], sendo que uma ação flexível tem o mesmo significado ou interpretação que o usado para metas flexíveis ("softgoals"). As ações flexíveis são ações pouco precisas, para as quais não se pode identificar um resultado concreto a priori e, além disso, a confirmação da execução da ação flexível pode depender de interpretação. Uma ação flexível (exemplos de "ações flexíveis": analisar, apurar, avaliar, conferir, controlar, gerenciar, verificar, validar, etc.) é diferente de uma ação concreta.

2.3 O Léxico Ampliado da Linguagem

O Léxico Ampliado da Linguagem (LAL ou Léxico) [6] se baseia na premissa de que em uma organização existe uma linguagem estendida da linguagem natural que é utilizada pelos atores da organização. Essa linguagem, com vocabulário peculiar (composto de símbolos), é denominada linguagem da aplicação [6]. O Léxico tem como justificativa uma razão simples: "entender a linguagem do domínio sem se preocupar com o entendimento do problema envolvido". O objetivo do LAL é capturar o vocabulário (termos e sentenças) intrínseco ao contexto organizacional.

A Figura 1 [5] mostra os componentes do LAL através de um diagrama de classes. No diagrama de classes o LAL é formado por símbolos e cada um deles é identificado por um nome. Um símbolo pode possuir mais de um nome (caso de sinônimos) e é

representado por duas descrições. A primeira, denominada noção, é o significado do símbolo, equivalente à descrição encontrada em dicionários. A segunda, chamada de impacto ou resposta comportamental, é a informação adicional sobre os efeitos provocados pelo símbolo. O diagrama mostra que ambos, noção e impacto, fazem menção a outros símbolos. Os símbolos no LAL são classificados em quatro categorias: objeto, sujeito, estado e verbo.

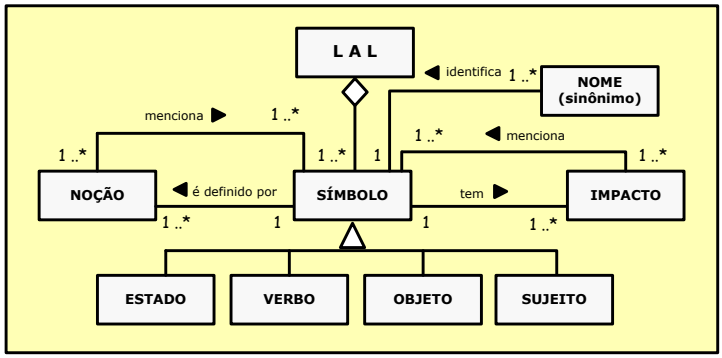


Figura 1 Os elementos do LAL através de um diagrama de classes.

O LAL segue dois princípios. O primeiro, "**o princípio da circularidade**", recomenda maximizar o uso dos outros símbolos do léxico quando se descreve a noção e o impacto de um novo símbolo. O segundo, "**o princípio do vocabulário mínimo**", recomenda minimizar o uso de símbolos externos ao vocabulário do contexto organizacional. Dicionários, de modo geral, só representam a denotação dos termos. A dificuldade do engenheiro de requisitos, decorrente da incerteza associada ao problema de elicitar as metas flexiveis devido à nebulosidade das ações flexíveis, pode ser contornada pela aplicação de algumas regras para a definição do LAL. Como por exemplo o tempo (infinitivo, presente, passado ou futuro) dos verbos (ações) descritos na definição dos impactos devem ser redigidos no infinitivo ou no presente do indicativo. Se na descrição do impacto, exemplos: (1) "VERIFICAR o cálculo dos impostos ..." e (2) "OCULTAR os valores transcritos ... " houver verbo que se refira a ação flexível, então podem ser sugeridos como candidatos a meta flexível os atributos *correto* para o exemplo (1) e o de *sigilo* para o (2).

3 Contribuições Científicas

O ambiente de desenvolvimento escolhido para aplicar as técnicas de inteligência artificial é o CLIPS - C Language Integrated Production System [7]. Os detalhes sobre o formato das bases de conhecimento e como se dá a interação entre ela e a base de regras está descrita nos artigos [8], para um problema mais restrito referente a requisitos não-funcionais. Os passos abaixo, originados de projetos anteriores, expressam em macro-nível a seqüência em que os resultados podem ser obtidos:

1. representar a descrição de domínios em formato adequado à análise utilizando sistemas baseados em conhecimento, utilizando sintaxe do CLIPS;

2. inspecionar essa base de dados utilizando a máquina de inferência do CLIPS e as regras de produção da base de conhecimento, que implementam as relações e regras de extração de requisitos utilizando os sinônimos;
3. para cada requisito identificado explicita ou implicitamente na descrição sendo analisada, anotar na base de fatos do problema a presença do requisito identificado e sua classificação em atributo funcional ou não-funcional;
4. onde o sistema não puder decidir utilizando os recursos disponíveis nas bases de conhecimento (fatos ou regras), acionar o analista de requisitos via diálogos simples, para melhorar a classificação do requisito identificado;
5. repetir os passos de 2 até 4 até terminar a análise.

Como resultado parcial, também já temos disponíveis tabelas de sinônimos, geradas para o caso especifico de atributos de transparência em projetos anteriores. Parte de uma dessas tabelas é mostrada a seguir, na Tabela 1.

Tabela 1. Exemplo de sinônimos dos atributos de Transparência, extraído da base Wordnet [9].

Atributos	Atributos	Sinônimo
Portável	Portable	portability
Disponível	Available	availability, usable, useable, uncommitted
Publicidade	Disclosure	revelation, revealing
Uniforme	Uniform	consistent, undifferentiated, unvarying
Simples	Simplicity	ease, easiness, simpleness
Intuitivo	Intuitive	intuitive , nonrational, visceral
Desempenho	Performance	execution, operation, functioning, carrying out, carrying into action

4 Conclusões

O objetivo da linha de pesquisa é facilitar e melhorar a extração de NFRs de descrições de domínios disponíveis no LAL – Léxico Ampliado da Linguagem, podendo ser futuramente expandido também para uso em requisitos funcionais. Resultados parciais mostram que as técnicas de inteligência artificial são perfeitamente aplicáveis ao problema, permitindo ao analista enxergar requisitos escondidos ou implícitos nas descrições. Porém sabemos que o sucesso do resultado depende de descrições bem formuladas, o que significa um trabalho competente de captura de símbolos em documentos e em entrevistas além de seguir as heurísticas, para que as descrições fiquem capazes de serem tratadas pela máquina de inferência.

Várias possibilidades de melhorias e avanços das técnicas são possíveis dentro da inteligência artificial. Por exemplo, o uso de principios e regras da lógica nebulosa pode melhorar a identificação de intenções implícitas nas descrições do LAL, atribuindo um grau de certeza ao grau de sinonimia envolvido com os termos. É possível, por exemplo, que estejamos assumindo que dois termos sejam sinônimos, em um contexto em que de fato eles não são sinônimos, o que levaria a conclusões erradas.

5 Trabalhos Futuros e em Andamento

Ainda não chegamos ao ponto de desenvolvimento dos projetos no qual seja possivel medir de alguma forma a efetividade da abordagem proposta, para podermos afirmar via números, o quanto a técnica melhorou a extração de requisitos. Entretando, pretendemos através da experimentação cientifica avaliar os resultados. Dois projetos estão sendo iniciados atualmente, dentro da linha de investigação descrita neste artigo: - um visa a aplicação de sinônimos para melhorar a elicitação de requisitos, fornecendo elementos mais precisos para preenchimento dos frameworks do LAL. Os resultados deste projeto permitirão gerar modelos MDE-Modelo de Dependências Estratégicas e MRE-Modelo de Razões Estratégicas do iStar mais precisos, com menos ambiguidades. As técnicas de inteligência artificial citadas neste artigo permitirão extrair intencionalidade a partir das descrições disponibilizadas no LAL; -o outro busca criar um gerador semi-automático de modelos MRE e MDE do iStar a partir de descrições disponíveis no LAL. Já existe um método para esta geração feita manualmente [5], ainda não existe uma ferramenta que apoie a transformação. A ferramenta automatizará parte do processo de elaboração visto que diversos problemas como ambiguidade e dependência de contexto tornam a completa automação um trabalho dificil e muito sujeito a erros.

Referências

1. Berry, D. M., Kamsties, E.: Ambiguity in requirements specification. Perspectives on Software Requirements, pp. 7-44. Springer US (2004).
2. Yu, E. Modelling Strategic Relationships for Process Reengineering. PhD Thesis – Dept. of Computer Science, University of Toronto, Toronto (1995).
3. Chung, L., Nixon, B., YU, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering – Kluwer Academic Publishers , Massachusetts, USA (2000).
4. Carvalho, L. C. S.: Análise de Sistemas: o outro lado da informática, LTC - Rio de Janeiro, (1988).
5. Oliveira, A. P. A.: Engenharia de Requisitos Intencional: Um Método de Elicitação, Modelagem e Análise de Requisitos. PhD Thesis - PUC-Rio, Rio de Janeiro (2008).
6. Leite, J. C. S. P., Franco, A. P. M.: A Client Strategy for Conceptual Model Acquisition. In: Proceedings of the International Symposium on Requirements Engineering, pp. 243-246. IEEE Computer Society Press, San Diego (1993).
7. Giarratano, J. C.: CLIPS User's Guide. Version 6.0. NASA Lyndon B. Johnson Space Center, Software Technology Branch, Houston, Texas, EUA (1993).
8. Baia, J. W., Braga, J. L.: Uso de sinônimos na identificação de atributos de transparência**.** In: 16th WER - Workshop em Engenharia de Requisitos (15th CibSE - Congresso Ibero-Americano em Engenharia de Software), pp.94-104, Montevideo (2013).
9. WORDNET: A lexical database for English, `http://wordnet.princeton.edu/wordnet/`.
10. Finkelstein, A. and Dowell J. "A Comedy of Errors: The London Ambulance Service Case Study“ Proceedings of the Eighth International Workshop on Software Specification and Design, IEEE Computer Society Press 1996, pp. 2-5.

Requirements and Architectures for Adaptive Systems

João Pimentel[1,2], Jaelson Castro[1], Emanuel Santos[1], Monique Soares[1], Jessyka Vilela[1], and Gabriela Guedes[1]

[1] Centro de Informática, Univ. Federal de Pernambuco (UFPE), Recife, Brazil
{jhcp,jbc,ebs,mcs4,jffv,ggs}@cin.ufpe.br
[2] Department of Information Eng. and Computer Science, University of Trento, Italy

Abstract. The growing interest in developing adaptive systems has led to numerous proposals for approaches aimed at supporting the development of such systems. Some approaches define adaptation mechanisms in terms of architectural designs, consisting of concepts such as components, connectors and states. Other approaches are requirements-based, thus concerned with goals, tasks, contexts and preferences as concepts in terms of which adaptation is defined. By considering only a partial view of software systems (either the problem space or the solution space), such proposals are limited in specifying the adaptive behavior of a software system. In this paper we present ongoing work towards deriving architectural models in order to support the design and runtime execution of software adaptation both at a requirements and architectural level.

Keywords: adaptive systems, architectural design, adaptation control mechanisms, requirements

1 Introduction

In [1] the authors, by conducting a comparative study, concluded that requirement and architecture based approaches for software adaptation share common elements, such as the use of feedback loops and of external control mechanisms. However, there are differences that reveal complementary advantages and disadvantages of the two approaches.

On the one hand the requirement-based approaches capture and model the objectives of the system, but they lack awareness about the capabilities and the limitations of the proposed solution. On the other hand, architectural models provide guidance for the deployment of the monitoring mechanisms and the effectors that apply the adaptation process on the target system. The objectives of the system, however, are not modeled making it difficult to handle changes at the requirements level.

A third dimension to this combination of models is related to the system behavior. It is often the case that a system fails because the execution plan of its tasks was not appropriate for the holding conditions. Therefore, we propose the derivation of statecharts from goal models, completing this variability puzzle that captures all the aspects of the software system.

2 Objectives of the research

2.1 Baseline

The baseline for this ongoing work is the *Zanshin* approach for the design and development of adaptive systems [2–4] which, in its turn, is based on Goal-Oriented Requirements Engineering (GORE) [5]. Focussed on the feedback loop for adaptation, *Zanshin* augments goal models with the requirements for the monitor and adapt phases of such loops.

To illustrate *Zanshin*, Fig. 1 shows a goal model specifying the requirements for an adaptive Meeting Scheduler system. Traditional $i^\star$ elements (goals, softgoals, tasks) appear alongside domain assumptions (rectangles) and quality constraints (round-cornered rectangles), which are necessary for our adaptation purposes. Also, lines connecting different elements represent refinement/operationalization relations, following obvious AND/OR Boolean semantics for goal satisfaction. Finally, small circles and diamonds are elements introduced by *Zanshin*, namely *Awareness Requirements* (*AwReqs*) and *Control Variables*.

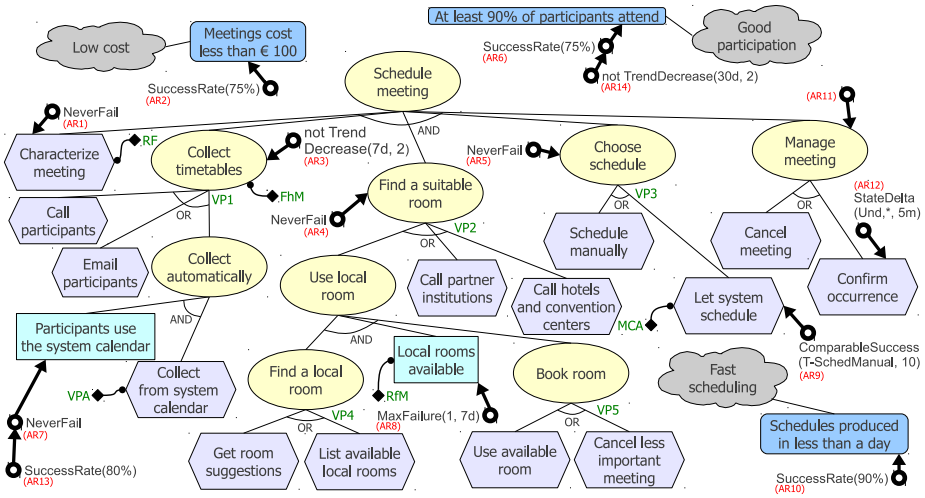


Fig. 1. Goal-based requirements specification for a Meeting Scheduler.

In the first step of the approach, *AwReqs* are elicited as requirements for the monitoring component of the feedback loop. *AwReqs* talk about the states assumed by other requirements—such as their success or failure—at runtime [2], representing, thus, situations in which the stakeholders would like the system to adapt. For example, *AR1* states that task *Characterize meeting* should never fail, whereas *AR2* indicates that the quality constraint that operationalizes softgoal *Low cost* should succeed 75% of the time.

The second step is called *System Identification* [3] and aims at identifying system parameters that can be changed at runtime and representing, in a qualitative way, how such changes can affect indicators of requirements convergence. We use *AwReqs* as indicators and consider two kinds of parameters: OR-refinements are called *Variation Points* (e.g., how to *Collect timetables* in *VP1*), whereas *Control Variables* abstract OR-refinements which are impossible or infeasible to be represented in the model (*FhM*, from how many people to collect timetables). The relation between parameters and indicators is represented by differential relations, e.g., $\Delta(AR3/FhM) < 0$, which reads "increasing (resp. decreasing) *FhM* contributes negatively (resp. positively) to *AwReq AR3*".

The third and last step concerns the requirements for the adaptation component of the feedback loop, represented by *Evolution Requirements* (*EvoReqs*) [4]. *EvoReqs* specify what the system should do to adapt in one of two ways: stakeholders can either give specific instructions or they can choose to ask the system to analyze the different parameters which have an impact on the failing indicator and change one (or more) of them accordingly. A framework[1], also called *Zanshin*, implements the generic features of a feedback loop in order to provide adaptation to base systems according to their requirements models. See, e.g., [6].

2.2 Objectives

Our research interest is to take further the baseline that we described above by combining requirement models with the software architecture. Towards this direction we propose a guiding methodology to derive architectural models from requirements. This effort would result in a more suitable specification for an adaptive software system, since the entire spectrum of its operational variability would be represented. Therefore, the system would have the maximum variety of alternatives when it has to deal with failures or with environmental changes. The last part of this work would be to advance the adaptation process implemented by *Zanshin* by making it quantitative, in order to acquire higher precision. Moreover, the framework will be extended to be able to deal with multiple failures, exploiting techniques inspired from Control Theory.

3 Scientific contributions

3.1 Architectural derivation

Architectural derivation is concerned with the generation of architectural models, which can include: components & connectors models for describing the system structure; statecharts for describing the system behavior; feature model for expressing the variability of the system; and so on. These different models are complementary, each one capturing a particular view of the system being designed. Thus, different approaches are required in order to derive these different models.

[1] See `https://github.com/sefms-disi-unitn/Zanshin/wiki`.

In previous work [7] [8], we proposed a set of methods to derive the aforementioned models from goal models. The key of that proposal was to derive the models in such a way as to preserve the variability expressed in the input model. However, when considering architectural derivation and its design decisions for the particular case of adaptive systems, there are three new concerns that arise:

a. *Additional variability* — there may be different alternatives to accomplish a given task. For instance, different algorithms can be used to schedule a meeting automaticallly, each with its different benefits and drawbacks. The alternatives identified during architectural derivation will expand the space of adaptation possibilities.
b. *Additional control elements* — besides referring to requirements concepts, *Zanshin* elements (such as *AwReqs* and *Control Variables*) may also refer to and influence architectural concerns. For instance, the time interval for a timed transitition could be defined as a *Control Variable*, rather than as a pre-defined, static interval.
c. *Additional features to support adaptation* — the support of self-adaptation may require the inclusion of new features in the system. This is the case, for instance, when the system requires some kind of instrumentation in order to monitor the satisfaction of *AwReqs*.

In [9] we handled the identification of additional features, considering the monitoring capabilities required to monitor the context. There, we were concerned with the derivation of components & connectors models. An approach for eliciting future requirements, which can be used to identify additional variability (both at requirements and architectural level) was presented in [10]. In [11] [12] we explore additional variability derived from different web services that are available.

Currently, we are working on including additional variability and additional control elements, while supporting the derivation of statecharts.

3.2 Derivation of statecharts

The process for deriving statecharts from goal models comprises 6 steps, depicted in Fig. 2. The first step, *Delegate tasks*, consists of assigning the tasks that will not be performed nor supported by the system proper – e.g., tasks that will be performed by an external actor (human or otherwise). Since these tasks are not carried out by the software system itself, they are not considered during the remainder of the process. In the next step, *Define basic flow*, the architect analyzes all refinements of the goal model and defines a flow expression for each. These flow expressions will be used in the next step, *Generate base statechart*, to create a skeleton of the statechart. Since these expressions are not as expressive as statecharts (although rich enough for defining the basic flow), and can be included as annotations in a goal model, they are a useful intermediate abstraction between goal models and statecharts.

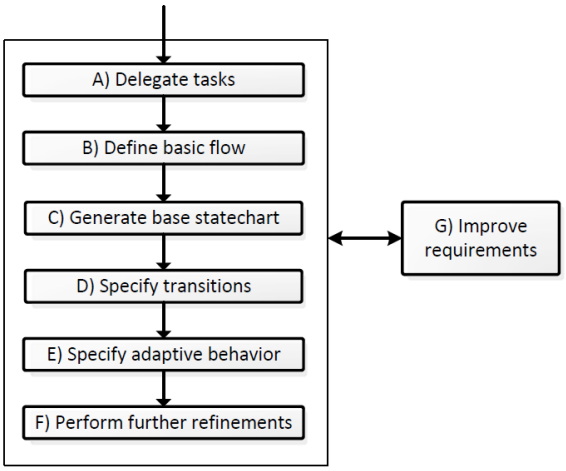


Fig. 2. Process for statechart derivation.

In the remaining steps the statechart will be refined, as follows. First, during *Specify transitions* the architect defines events and conditions of the derived transitions. Then, the statechart is enriched to describe the system's adaptivity behavior, including the interaction with an external component that provides adaptation-related functionality. This takes place during *Specify adaptive behavior*. As a last step, *Perform further refinements* allows the architect to expand the model in order to include technical details and other concerns that may not have been handled earlier, by exploiting the concept of sub-states.

4 Conclusions and Future Work

In this paper we present ongoing work towards improving the support for the development of adaptive software systems. The combination of requirements and architectural models will provide a richer space of adaptation specification. The proposed derivation methodologies will facilitate the creation of adaptive systems making use of the *Zanshin* framework.

A prototype tool for the derivation of statecharts, as presented in Section 3.1, is currently under development[2].

Acknowledgments. This work has been supported by the ERC advanced grant 267856 "Lucretius: Foundations for Software Evolution" and by Brazilian institutions CAPES and CNPq.

[2] Available at `https://github.com/jhcp/GoalArch`.

References

1. Angelopoulos, K., Souza, V.E.S., Pimentel, J.: Requirements and Architectural Approaches to Adaptive Software Systems: A Comparative Study. In: Proc. of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (to appear). (2013)
2. Souza, V.E.S., Lapouchnian, A., Robinson, W.N., Mylopoulos, J.: Awareness Requirements. In Lemos, R., Giese, H., Müller, H.A., Shaw, M., eds.: Software Engineering for Self-Adaptive Systems II. Volume 7475 of Lecture Notes in Computer Science. Springer (2013) 133–161
3. Souza, V.E.S., Lapouchnian, A., Mylopoulos, J.: System Identification for Adaptive Software Systems: A Requirements Engineering Perspective. In: Conceptual Modeling ER 2011. (2011) 346–361
4. Souza, V.E.S., Lapouchnian, A., Angelopoulos, K., Mylopoulos, J.: Requirements-driven software evolution. Computer Science - Research and Development (2012) 1–19
5. Mylopoulos, J., Chung, L., Yu, E.S.K.: From Object-Oriented to Goal-Oriented Requirements Analysis. Communications of the ACM **42**(1) (1999) 31–37
6. Tallabaci, G., Souza, V.E.S.: Engineering Adaptation with Zanshin: an Experience Report. In: Proc. of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (to appear). (2013)
7. Yu, Y., do Prado Leite, J.C.S., Lapouchnian, A., Mylopoulos, J.: Configuring features with stakeholder goals. In: Proceedings of the 2008 ACM symposium on Applied computing - SAC '08, ACM Press (2008) 645–649
8. Yu, Y., Lapouchnian, A., Liaskos, S., Mylopoulos, J., Leite, J.C.S.P.: From Goals to High-Variability Software Design. In: Foundations of Intelligent Systems. Volume 4994/2008. (2008) 1–16
9. Pimentel, J., Lucena, M., Castro, J., Silva, C., Santos, E., Alencar, F.: Deriving software architectural models from requirements models for adaptive systems: the STREAM-A approach. Requirements Engineering **17**(4) (June 2012) 259–281
10. Pimentel, J., Castro, J., Perrelli, H., Santos, E., Franch, X.: Towards anticipating requirements changes through studies of the future. In: 5th International Conference on Research Challenges in Information Science, IEEE (May 2011) 1–11
11. Pimentel, J., Castro, J., Santos, E., Finkelstein, A.: Towards Requirements and Architecture Co-evolution. In: Advanced Information Systems Engineering Workshops. (2012) 159–170
12. Franch, X., Grunbacher, P., Oriol, M., Burgstaller, B., Dhungana, D., Lopez, L., Marco, J., Pimentel, J.: Goal-Driven Adaptation of Service-Based Systems from Runtime Monitoring Data. In: 2011 IEEE 35th Annual Computer Software and Applications Conference Workshops, IEEE (July 2011) 458–463