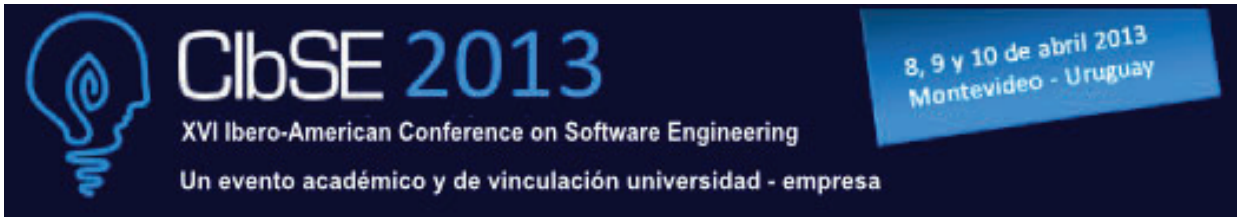


Memorias del XVI Workshop de Ingeniería en Requisitos WER 2013

Del 8 al 10 de Abril, 2013
Universidad ORT Uruguay – Campus Centro
Montevideo, Uruguay

Editores – Chairs de Programa: **Ph.D Nelly Condori Fernández**
Universidad Politécnica de Valencia - España

Ph. D Leandro Antonelli
Universidad Nacional de La Plata - Argentina

Organizadores: Universidad ORT Uruguay – Universidad de la República – Antel

Media partner: EL PAIS

Sponsors: GeneXus Consulting – Onetree – Microsoft – Gallito.com – TATA Consultancy Services – MecadoLibre – Globant – Tilsor – UNIT

Support: Centro Latinoamericano de Estudios en Informática – Agencia Nacional de Investigación e Innovación – IEEE Computer Society Capítulo Uruguay – Cámara Uruguaya de Tecnologías de la Información – Gega Multimedios

ISBN: 978-9974-8379-2-8

Message from the General Chair

I cordially welcome you to Montevideo, Uruguay, and to the 16th Iberoamerican Conference on Software Engineering (Cibse 2013).

From its inception Cibse is well known to provide a growing forum for those interested in exploring and discovering the advances and research in the field of software engineering in Ibero-america. Trying to follow this tradition, we believe this year conference program has much to offer to researchers, practitioners, students and educators.

As in previous editions, this conference will hold the three traditional tracks covering the areas of Software Engineering, Requirements Engineering, and Experimental Software Engineering; it will also host a Doctoral Symposium, three keynotes and four tutorials. For the first time this year's conference will host an Industry Forum that we hope it will encourage the exchange of experiences between scientists and professionals.

Cibse is a constantly growing research oriented conference. On this edition there were 36 accepted papers in the three main tracks and we are pleased to have authors representing several countries like Argentina, Brazil, Cuba, Italy, Spain, Peru, Venezuela and Uruguay.

We are fortunate to have three keynotes that span different areas of interest to the software engineering community. Bill Nichols´s keynote will take a deeper look into the stages of quality development using TSP. Carolyn Seaman will address on how the Technical Debt metaphor can be used by practitioners and empirical researchers as a new path to technology transfer. Eduardo Mangarelli and Fernando Machado will bring an industrial perspective on the challenges presented by the complexities that today's applications are expected to manage, and on how these challenges are disrupting the way we engineer software.

Tutorials have always been an efficient way to learn about diverse themes in Software Engineering. This year we will offer four tutorials covering leading edge topics and presented by the following renowned speakers: Renata Guizzardi, Luis Olsina, Andrea Delgado and Bill Nichols.

As many conferences, this Cibse was mostly organized by a group of dedicated people that generously offered their time to perform the countless tasks needed to make the conference a success. I would like to thank all the volunteers that have devoted their energy to the goal of making it an outstanding experience to you. These include, among others, the Steering Committee, Track Chairs, PC members, student volunteers and the university's staff. Thank you all for your dedication and support. I must specially recognize the work of our Organization Chair - Santiago Matalonga, our Academy/Industry Liaison – Ana Laura Trias, and to Liliana Pino our webmaster, we couldn't have done this without your leadership, patience and hard work!

I would also like to thank the invaluable support of the companies and organizations that, with their sponsorship, made it possible to offer full access to the conference to over fifty students. I believe this is a unique opportunity for our future Software Engineers and researchers.

Finally we hope that you will enjoy your stay at Montevideo - the Iberoamerican Capital of Culture - 2013. Welcome to Cibse 2013!

Gastón Mousqués / General Chair

Preface

Welcome to the sixteenth edition of the Workshop on Requirements Engineering series (WER'13) collocated at the Ibero-American Conference on Software Engineering (CibSE'13) and to be held in Montevideo, Uruguay.

WER 2013 continues with the tradition of previous editions, providing an active forum for researchers and practitioners. Although the workshop started as a meeting in order to consolidate the Ibero-american community of researchers on requirements engineering, over the last years, WER has also attracted attention of researchers from other parts of the world as well.

The international program committee, from 11 different countries, was composed of 34 relevant researchers coming both from academia and industry. We received 35 abstracts submissions, which were finally capitalized in 27 papers. After a rigorous reviewing process, carried out by at least three members of the program committee, and one additional discussion session, 9 papers were accepted as full papers (33%) and 3 as short papers. The 12 selected papers will be made available in the repository http://wer.inf.puc-rio.br/WERpapers.

In this edition the program also included a panel that aims to discuss issues in requirements engineering technology transfer.

We thank all members of the program committee and external reviewers for their dedication, effort and excellent work to improve papers quality. We also are greatly grateful to the local organization members for making possible to carry out this workshop collocated at CIBSE 2013. And we hope that you will enjoy and find useful what we have prepared for you.

Co-chairs

Leandro Antonelli (UNLP, Argentina)
Nelly Condori-Fernandez (UPVLC, Spain)

Program Committee

Alejandro Oliveros, Universidad Nacional de La Plata, Argentina
Alicia Martínez Rebollar, CENIDET, Mexico
Amador Durán Toro, Universidad de Sevilla, Spain
Anthony Finkelstein, University College London, United Kingdom
Antonio De Padua Albuquerque Oliveira, Universidade do Estado do Rio de Janeiro, Brazil
Carme Quer, Universitat Politécnica de Catalunya, Spain
Claudia P. Ayala, Technical University of Catalunya, Spain
Daniel Berry, University of Waterloo, Canada
Dolors Costal, Universitat Politecnica de Catalunya, Spain
Fernanda Alencar, Universidade Federal de Pernambuco, Brazil
Gemma Grau, Accenture, Spain
Gladys Kaplan, Universidad Nacional de La Matanza, Argentina
Graciela Hadad, Universidad Nacional de La Matanza, Argentina
Hugo Estrada, CENIDET, Mexico
Isabel Brito, Instituto Politécnico de Beja, Portugal
Jaelson Castro, UFPE - Universidade Federal de Pernambuco, Brazil
João Araújo, Universidade Nova de Lisboa, Portugal
Jose Luis de La Vara, Norway
Juan Pablo Carvallo, Universidad del Azuay, Ecuador
Juan Sanchez, Valencia University of Technology, Spain
Julio C. S. Do Prado Leite, Pontificia Universidade Católica do Rio de Janeiro, Brazil
Luiz E. Galvão Martins, Universidade Federal de São Paulo, Brazil
Luiz Marcio Cysneiros, York University, Canada
Lyrene Fernandes Da Silva, Universidade do Estado do Rio Grande do Norte, Brazil
Mabel Bertolami, Universidad Nacional de la Patagonia San Juan Bosco, Argentina
Maria Lencastre, Universidade de Pernambuco, Brazil
Marcela Ridao, INTIA UNICEN, Argentina
Maya Daneva, University of Twente, The Netherlands
Oscar Dieste, Universidad Politécnica de Madrid, Spain
Thomas Alspaugh, Georgetown University, USA
Vera Werneck, Universidade do Estado do Rio de Janeiro, Brazil

External reviewers

Sofia Azevedo,
Rodrigo Fonseca,
Alejandrina Aranda,
David Ameller.

Quality Properties Evaluation for Software Requirements Specifications: An Exploratory Analysis

Roxana Saavedra[1], Luciana Ballejos[2], Mariel Ale[3]

[1]CIDISI – UTN – Facultad Regional Santa Fe – rsaavedra@frsf.utn.edu.ar
[2]CIDISI- UTN - Facultad Regional Santa Fe – lballejos@santafe-conicet.gov.ar
[3]CIDISI- UTN - Facultad Regional Santa Fe – male@frsf.utn.edu.ar

Abstract. Most software problems arise from deficiencies in the manner in which software requirements are elicited and expressed. Ensuring that the Software Requirements Specification document (SRS) has the necessary quality is crucial to the success of any software development project, since its information is used across all project stages. However, assessing the quality of a SRS is not a simple process, mainly by the multitude of proposals, often contradictory, of the attributes to be evaluated and the methodologies used for that purpose. This work is intended to be a compendium of the most important tendencies and strategies in the field that serves as a starting point for developing comprehensive models and tools for quality attributes evaluation in a SRS.

Keywords. Software Requirements Specification, quality attributes evaluation, quality models

1 Introduction

The primary measure for an information system to be successful is the degree in which it meets the intended purpose. Requirements Engineering (RE) is a subtask of Software Engineering which deals with the discovering of that purpose by identifying stakeholders and their needs, and documenting them for their future analysis, communication, and subsequent implementation [1].

In RE processes there is a continual need for efficiently managing the great volume of information and knowledge generated and used during all activities which compose the software development process. Thus, diverse are the challenges that must be considered when managing requirements-related information in software development projects. In this sense, ambiguous requirements must be minimized since they produce waste of time and repeated work.

Related to this, there exist in the literature diverse proposals in order to give guidance in the assessment of different attributes or properties for requirements, which helps in controlling if their specification is made in a correct way. Some of them also propose quality models to be considered when evaluating a Software Requirements Specification (SRS), the main deliverable produced in RE, which is used throughout the project [2,3].

The main goal of this paper is to analyze the state-of-the art in this area, in order to give a more consistent support for software engineers when generating requirements

specifications. Moreover, the results might be considered for the future development of tools for supporting automate or semi-automate evaluation of SRS quality.

The paper is organized as follows: Section 2 describes the research methodology used for the study. Section 3 presents diverse proposals in SRS quality properties, analyzing their similarities and differences. Meanwhile, Section 4 discusses and analyzes diverse proposals or approaches for evaluating the attributes described in the previous section. Related to this, Section 5 describes influences between these properties. Finally, Section 6 is devoted to discuss future trends in this area and the conclusions of the paper.

2 Research Methodology

The purpose of this study is to understand the trend of quality-related attributes assessment for SRS by examining published articles and offering, at the same time, insights and future directions in this area.

To this end, the following electronic journal databases were searched to provide an exhaustive bibliographic revision of research papers in the area:

- EBSCO Discovery Service,
- EconPapers,
- Compendex. Engineering Village,
- Engineering Village 2: Referex Engineering,
- National Digital Library of Theses and Dissertations (NDLTD),
- NTRS: NASA Technical Reports,
- Scopus. Elsevier,
- Social Science Research Network (SSRN),
- ACM Portal,
- IEEE Library.

The search process was performed based on three descriptors: “Software Requirements Specification”, “Quality Models” and “Quality Attributes Evaluation”. The selection of the descriptors was performed according to their different levels of generality, which allows refining the search results. The search process started from more general (e.g. "quality models") to more specific (e.g. "Software Requirements Specification") phrases, using the facilities of “search within the results” offered by most search engines. Despite the above, and due to the large quantity of papers returned by the databases, the following exclusion criteria were used:

- Given the limited number of indexed publications addressing this particular subject, the consideration of publications only in English. This was done in order to ensure that these articles have been accessed and validated by the widest possible audience. For the same reasons, non-refereed conference papers, unpublished master and doctoral dissertations were excluded.
- Because the idea of evaluating quality attributes for the SRS is relatively new, the search was restricted from 1990 onwards.
- Only research papers referring specifically to SRS quality attributes were chosen.

3 Quality Attributes

In general, quality attributes for a SRS are part of a quality model and should be used to assess the quality of a requirements document, or individual requirements contained in a SRS [4]. Thus, each attribute might be related to the entire SRS or to each requirement defined in it. This is due to, since for many attributes to be attained by the SRS, they have previously to be achieved for each requirement defined in it. Related to this, many of the authors proposing the same property coincide in their considerations, some issues regarding particularities in the analysis deserve to be pointed out. In the next subsections diverse definitions and proposals for each property are presented.

3.1 Unambiguous

A SRS is *unambiguous* if every requirement stated therein has only one possible interpretation [2,5,6]. Some authors give similar definitions for unambiguous requirements [7,8,9,10,11]. Pohl [7] states that a requirement is unambiguous if all stakeholders with approximately the same knowledge about the system and its context interpret the requirement in the same way. It also requires that, at least, each characteristic of the final product be described using a single unique term [5], and those terms that could have multiple meanings in a particular context, should be included in a glossary where its meaning becomes more specific [5,8].

3.2 Complete

The IEEE 830:2009 Standard [5] establishes that a SRS is *complete* if it contains the following elements: 1) all significant requirements; 2) definition of the responses of the software to all realizable classes of input data in all realizable classes of situations; 3) complete labels and references to all figures, tables and diagrams in the SRS and definition of all terms and units of measure. Some authors also add the rule that all pages must be numbered and all referenced material must be present in SRS [2,12,13].

Furthermore, a SRS is *complete* if all relevant requirements are specified [2,7,8,10], and, therefore, irrelevant requirements should be absent [7,14]. Some authors [2,5], consider that a SRS is not complete if there are unfinished sections, i.e., there are marks "To Be Determined" (TBD). A requirement is *complete* if it is fully specified [7,6,8] and does not omit any piece of information that is relevant to some stakeholder [7].

3.3 Consistent

Some authors treat *internal* and *external consistency* as different quality properties [2,9], others seek only the *internal consistency* [5,7,14,15,16,17,18], and others consider the *internal* and *external consistency* as a same quality attribute [3,6,8,19].

3.3.1 Internally Consistent

A SRS is considered *internally consistent* if not subset of requirements outlined in it have conflicts [2,5,7,9,10,14]. Moreover, a requirement is *consistent* if the statements within it do not contradict each other [7].

3.3.2 Externally Consistent

A requirement in the SRS is considered *externally consistent* has no conflicts with any project documentation (i.e., system requirements specifications, statements of work, white papers, an early SRS version with which the new version must be compatible, and system requirements specifications of other systems to which this system must have interface). Thus, a SRS is considered externally consistent if the requirements described in it have no conflict with any project documentation [2,6,8,9,19].

IEEE 830:2009 [5] states that if a SRS disagrees with some superior-level document, such as a system requirements specification, then, this is not *correct* rather than *externally inconsistent* (see section 3.4).

3.4 Correct

A SRS is considered *correct* if every requirement is something required to build the system, i.e., each requirement contributes to the satisfaction of some need [2,5,6]. The SRS should be compared with any superior specification to ensure that it agrees. The customer or user can also determine whether the SRS correctly reflects the actual needs [5,8]. Instead, some authors define the quality attribute *validability/valid* when they establish that the client should be able to confirm that SRS requirements describe the system that meets their needs [10,14]. Pohl [7] states that a requirement is *correct* if the relevant stakeholders confirm that it is correct and require the system to perform the requirement completely. Thus, a requirement is *incorrect* if it unnecessary adds some functionality or quality property (gold platings).

3.5 Traceable

A SRS is considered *traceable* if each requirement has a clear source and is easily referenced in subsequent development phase or documentation [5,6]. IEEE 830:2009 [5] recommends two types of traceability: 1) *Backward traceability*, i.e., before the development stages. This is achieved when each requirement explicitly reference to its source in earlier documents. Some authors define this attribute as "*Traced*" [2,12,13,20]; 2) *Forward traceability*, i.e., all documents generated from the SRS. This is achieved when each requirement has a unique name or reference number. Some authors define it as "*Traceable*" [2,10,12,13,20]. A requirement is traceable if the origin and evolution as well as its impact and use in later stages of development is traceable [7,8].

Some authors treat separately *Traceable* and *Traced* quality properties [2,12,13, 20], others try these attributes together in Traceable [5,6,7,8], and others only consider Traceable attribute [10,14].

3.6 Verifiable (Testable)

A SRS is considered *verifiable* if every requirement stated therein can be verified [2,5,6]. A requirement is *verifiable* if there is a finite and cost-effective process with which a person or machine can check that the software product meets the requirement [2,3,5,6,7,8,14]. Pohl [7] states that if a requirement is underspecified, it is not objectively possible to decide if the requirement is realized as defined or not. It also states that, to facilitate verifiability, some acceptance criteria must be defined.

3.7 Modifiable

A SRS is considered *modifiable* if its structure and style are such that allow introducing easily, completely, and consistently any change, without affecting the structure and style [2,5,6,7,13]. To achieve modifiability a SRS must: 1) have a coherent and easy to use organization, a table of contents, an index, and explicit cross-references (see sections 3.11 and 3.14); 2) avoid redundancy (because problems can arise when a redundant requirement is altered in only one of the places where this occurs, resulting in a inconsistent SRS) (see section 3.12); 3) express each requirement separately (see section 3.13) [5,7,8].

3.8 Annotated by Relative Importance, Relative Stability or Version

A SRS is considered *ranked by importance* if each requirement in it has an identifier to indicate its importance [2,5,6,14]. One way to classify requirements is to distinguish classes of them as essential, conditional, and optional [2,5].

A SRS is considered *ranked by stability* if each requirement in it has an identifier to indicate the stability of that particular requirement [2,5,6,14]. Requirements stability can be expressed with the number of expected changes for any requirement [2,5].

A requirement is *rated* if its relevance has been determined and documented [7].

A SRS is considered *annotated by version* if a reader can easily determine which requirements will be satisfied in which program versions. One way of annotating requirements by version is to add a column in the SRS for each version of software to be produced and mark with an "X" next to each requirement in the respective columns [2].

Some authors distinguish the quality properties *Annotated by Relative Importance*, *Annotated by Relative Stability* and *Annotated by Version* [2,12,13]. Others consider *Annotated by Relative Importance* property as *Prioritized* [8], while others consider *Annotated by Relative Importance* and *Annotated by Relative Stability* as joint properties called *Ranked for Importance and/or Stability* or *Rated* [5,6,7,14].

3.9 Understandable

A SRS is considered *understandable* if its readers (customers, users, project managers, software developers, testers, and others) can easily comprehend the meaning of all requirements with a minimum of explanation [2,3,10].
According to Pohl [7], a requirement is *comprehensible* if its content is easy to understand. The comprehensibility of a requirement depends, among other things, on the selected document format and the stakeholders involved. Pohl [7] distinguishes between *Comprehensible* and *Readability*, while others define *Understandable* as part of *Readability* definition [2,3,10,20,21].

Furthermore, Pohl [7] defines that a SRS is *readable* if the reader can extract and easily understand its content. The readability and modifiability of a SRS is influenced by the structure and style of the document, so, in this sense, the SRS should have a coherent structure, each requirement should have a unique identification, redundancies should be avoided, and the requirements defined should be atomic. In order to reach these characteristics, diverse authors propose the use of IEEE Standard 830-2009 [5], which describes recommended practices for Software Requirements Specifications.

3.10 Concise

A SRS is considered *concise* if it is as short as possible without adversely affecting any other document quality. So, if there exist two SRS describing an identical system with identical quality measures, then the shorter the better [2,20].

3.11 Organized

A SRS is considered *organized* if its content is organized, that is, readers can easily locate information and logical relationships between adjacent sections are evident [2, 12,13,20]. According to Davis et al. [2], to achieve a useful organization: 1) a standard must be followed, and 2) one of the five organizational models must be used: group the functional requirements by user class, common stimulus, common response, feature or object. Some authors define the quality property *Organized* [2,12,13,20] and others only describe it as a required feature for achieving other quality attributes, such as *Modifiable* [5].

3.12 Not Redundant

A SRS is *redundant* if the same requirement is declared more than once [2,5,6,8]. Unlike other quality attributes, the redundancy is not necessarily bad. It is often used to increase readability of the SRS. However, it causes problems when a SRS is revised. If all occurrences of a redundant requirement are not modified, then the SRS becomes inconsistent [2,5,6,7,8]. If redundancy is necessary in the SRS, it should include explicit cross-references [5,7,8].

Some authors define the quality property *Not Redundant* [2,9]. Others treat this quality attribute as a guideline to be considered in requirements documentation [6], and others only describe it as a required feature for achieving other attribute quality as: *Modifiable* [5,7,8] and *Readability* (*Understandable*) [7].

3.13 Atomic

Each requirement in a SRS should be clearly determined and identified, without being mixed with other requirements [10]. A requirement is *atomic* if it describes a single and coherent event. A requirement is *not atomic* if it describes multiple isolated or just loosely coupled events which can be divided into several requirements [7].

Other authors treat this attribute similarly to the quality property *Not Redundant*, i.e., as a guideline to consider in requirements documentation [6] and as required feature to achieve the quality properties *Modifiable* [5,7,8] and *Readability* (*understandable*) [7].

3.14 Cross-Referenced

A SRS is considered *cross-referenced* if it cross-references are used to relate sections containing requirements to other sections containing: redundant requirements, more abstract or more detailed descriptions of the same requirements or requirements that depend on them or on which they depend [2].

3.15 Design Independent

A SRS is *design independent* if there are more than one system design and implementations that implement all requirements stated in it [2]. Génova et al. [10] call this quality property *Abstraction*, and state that requirements must tell what the system must do without telling how it must do it. Thus, a SRS should avoid excessive technical details about the implementation.

3.16 Electronically Stored

A SRS is considered *electronically stored* if the entire document has been produced with a word processor, was generated from a requirements database, or has been otherwise synthesized from some other form [2].

3.17 At Right Level of Abstraction/Detail

A SRS can provide different levels of detail [2,20]. A SRS being used as a contract between customer and developer should be relatively specific to ensure that the customer knows what is being acquired [2]. It is considered good practice to write the SRS requirements at a consistent level of detail [6].

3.18 Precise

The precision quality property is accomplished when all used terms in the SRS are concrete and well-defined [10]. Particularly, an SRS is considered *precise* if: 1) numerical quantities are used whenever possible, and 2) all numerical quantities have appropriate levels of precision [2].

3.19 Achievable

A SRS is *achievable* if there is at least one system that correctly implements all requirements stated in it [2]. Wiegers [8] calls this quality attribute *Feasible*, and defines it as the possibility to implement each requirement within the capabilities and limitations of the system and its operating environment.

3.20 Others Attributes

A SRS is considered *Prototypable* if there is a software tool capable of inputting it and providing a dynamic behavioral model that simulates the system behavior to build. This quality property is given only by Davis et al. [2].

A SRS is considered *Reusable* if it is possible to easily adopt or adapt their sentences, paragraphs and sections for use in a subsequent SRS [2]. This quality property is only described by Davis et al. [2].

A requirement is *Up to date* if it reflects the current status of the system and its context, such as the current stakeholders desire or current legal regulations [7]. It should be noted that this quality property was only proposed by Pohl [7].

4 Quality Properties Evaluation Analysis

There are many proposals for evaluating SRS quality attributes [2,3,6,10,11,12,13,14, 15,16,17,18,19,20,21]. The approaches found in the literature considering **vocabulary or language** include: *use of domain vocabulary,* related to the use of user vocabulary (glossary) in requirements descriptions [10,12,13]; *use of domain knowledge*, which considers interpretation and domain semantic knowledge (some authors use ontologies as knowledge resource) [2,15,16,17,18,19], and *natural language patterns detection* using keywords, key phrases and/or symbols as evidence of the occurrence for certain quality attributes [3,6,10,11,12,13,14,20,21].

Other approaches analyze **relations between requirements and artifacts** in SRS, in order to evaluate diverse attributes [10,15,16,17,18,19]. Among them, those proposing *overlapping between requirements* consider requirements referring to the same subject where contradictions between requirements, redundancy when there is a unnecessary repetition, and simple coupling when there is none of the above (and that implies some kind of dependency relationship) can be distinguished [10,15,16,17,18,19]. Also, other approach considers the evaluation of *requirements*

dependencies with other requirements or other artifacts of the development process [10]**.**

Diverse approaches are based in the analysis of **SRS structure** for evaluating some attributes [2,10,12,13,14,20,21]. Between them, *text structure in the SRS* considers requirements found in each hierarchical level of the SRS [10,14,20], while *specific characteristics of SRS document* includes presence of sections, table of contents and index, SRS size, etc. [2,12,13,14,20,21]. Meanwhile, *achievable features from SRS document* approach includes the evaluation of actual solution system designs, single system, etc. [2].

The analysis of the **requirements themselves** also is performed by diverse proposals [2,3,10,12,13,20]. In this sense, the *specific characteristics of a requirement* approach consider the use of explicit references, unique identifier, cross-references, versions and size for a requirement [2,3,10,12,13,20]. Moreover, the analysis of *deductible features from a requirement* includes the evaluation of cost and time required to verify a requirement [2].

Finally, as general approaches which consider the **jointly evaluation of groups of requirements**, approaches proposing *metrics that calculate the percentage of requirements that meet the analyzed attribute* [2], and *tools that use correct-by-construction paradigm* (where certain quality attributes are satisfied by the mere fact that tool be used to generate the SRS) are included in the analysis [12,13].

Thus, while Table 1 resumes which references have a concrete proposal on the evaluation or assessment techniques for each attribute, the approaches found together with the attributes evaluated by them are summarized afterwards.

The *use of domain vocabulary* is proposed for the evaluation of Unambiguous and Understandable [12,13], Atomic and Precise [10]. On the other hand, The *use of domain knowledge*, is included in assessment techniques proposed for Unambiguous [2,15,16], Complete [2,15,16,17,18,19], Internally Consistent [2], Correct [15,16,17,18], Understandable [2] and Atomic [18] quality attributes.

Moreover, the *use of natural language patterns* is proposed for evaluation techniques related to Unambiguous [11,13,14,20,21], Complete [3,12,13,20,21], Verifiable [3,21], Annotated by Relative Importance and Relative Stability [13], Understandable [3,10,14,20,21], Atomic [6,10], Design Independent and Precise [10] quality attributes.

Furthermore, within the *use of overlap between requirements* approach, proposals exist for the evaluation of Internally Consistent [15,16,17,18,19], Not Redundant [17], Unambiguous, Traceable, Understandable and Atomic [10] quality attributes.

Within the *use of requirements dependencies with other requirements or other artifacts of the development process* approach, evaluation techniques are proposed for quality attributes such as Traceable, Understandable and Atomic [10].

Also, the *use of text structure in the SRS* approach is considered when proposing evaluation techniques for quality attributes such as Understandable (considering the degree of nesting between requirements) [10], Organized (number of requirements at each hierarchical level) and Right level of Abstraction/Detail (considers the specification depth, i.e., the number of imperatives in each level of the SRS text structure) [14,20].

Table 1. References with Concrete Proposal on Attribute Evaluation Techniques

Reference / Property	[2]	[3]	[6]	[10]	[11]	[12]	[13]	[14]	[15]	[16]	[17]	[18]	[19]	[20]	[21]
Achievable	x														
Annotated by Relative Importance, Relative Stability or Version	x						x								
Atomic			x	x								x			
Complete	x	x				x	x		x	x	x	x	x	x	x
Concise	x							x						x	x
Correct	x			x					x	x	x	x			
Cross-Referenced	x														
Design Independent	x			x											
Electronically Stored	x						x								
Externally Consistent	x														
Internally Consistent	x	x							x	x	x	x	x		
Modifiable	x						x								
Not Redundant	x										x				
Organized	x					x	x	x						x	
Precise				x											
Prototypable	x														
Reusable	x														
Right Level of Abstraction/Detail	x							x						x	
Traceable	x			x		x	x							x	
Traced	x					x	x								
Unambiguous	x			x	x	x	x	x	x	x				x	x
Understandable	x	x		x		x	x	x						x	x
Verifiable (Testable)	x	x		x											x

The *use of specific characteristics of the SRS document* approach, is applied in assessment techniques related to quality attributes like Complete (considering that certain sections are present in the SRS) and Organized (considering whether the required sections are present in the required order and with the content required) [12,13], Concise (considering the size of the SRS) [2,14,20,21], Modifiable (considering the presence of table of contents and index, and the degree of cohesion and coupling of SRS sections), Electronically Stored (considers SRS volume that has been electronically stored) and Reusable (considers paragraphs in the SRS that exhibit reuse properties) [2].

Moreover, within the *achievable features from SRS document* approach, a set of evaluation techniques for diverse quality attributes is proposed. The attributes are: Design Independent (number of actual solution system designs that satisfy all requirements of SRS), Achievable (considering the existence of a single system), Prototypable (considers whether SRS can be partially written in a executable, interpretable or prototypable language) Reusable (considering whether the content of the SRS has been used in subsequent SRS) [2].

Also, the *use of specific characteristics of a requirement* approach is used for evaluating quality attributes like Internally Consistent (considering the use of explicit references in a requirement) [3], Traceable (considering using requirement unique identifier) [2,12,13,20], Traced (considering the use of cross-references) [2], Correct and Verifiable (considering the number of versions of a requirement) [10], Atomic (considering the size of a requirement) [10].

Moreover, in the *use of requirement deductible characteristics* approach, Davis et al. [2] propose assessment techniques for the quality attribute Verifiable (considering the cost and time required to verify the requirement).

In the *using metrics that calculate the percentage of requirements that meet the attribute in question* approach, Davis et al. [2] propose diverse metrics for evaluating quality attributes such as Externally Consistent, Correct, Annotated by Relative Importance, Relative Stability and Version, Not Redundant. The metrics proposed, in general, consider the ratio between the requirements satisfying some specific attribute, over the total number of requirements in the SRS. However, no concrete proposals are described by the authors, in order to give guidance over how considering or identifying the requirements which satisfy each attribute.

In the case of the *use of tools in the correct-by-construction paradigm* approach, evaluation techniques are proposed for Traced [12,13], Modifiable, Annotated by Version and Electronically Stored [13] quality attributes.

Finally, Davis et al. [2] indicate that Organized, Cross-Referenced, and Right Level of Abstraction/Detail attributes cannot be measured for different reasons. The authors affirm that "organization" is purely subjective and, thus, it cannot be measured. On the other hand, Cross-Referenced cannot be measured because there is no way to determine how many cross-references are appropriate in an SRS. In relation to Right Level of Abstraction/Detail, measuring the appropriateness of the SRS level of abstraction is highly scenario-dependent. Moreover, for Up to Date attribute was not found in the literature any evaluation proposal.

5 Influences between Quality Attributes

The possibility of creating a SRS of reasonable quality exists, but it has to be considered that most of the quality properties mentioned in this paper have positive or negative effect on other properties (see Figure 1). Because of this, it is necessary to determine which quality attributes are most important to the project, in order to achieve them. Below, the effects between attributes found in the literature analyzed are detailed:

- Generally, requirements considered unverifiable are ambiguous [5,8,10], incomplete or inconsistent [8,10], or infeasible requirements [8].
- The elimination of ambiguity in the SRS requires adding formality, which is not understood by people who are not computer experts, for example, users or customers [2,8,10].
- The less ambiguous is the SRS, the easier its modification [10].
- If a SRS is not complete in terms of objectives, rules, facts, and constraints of the problem domain, then it is considered incorrect for that domain [9,10].
- Exceeding the completeness of the SRS causes losing conciseness [2].

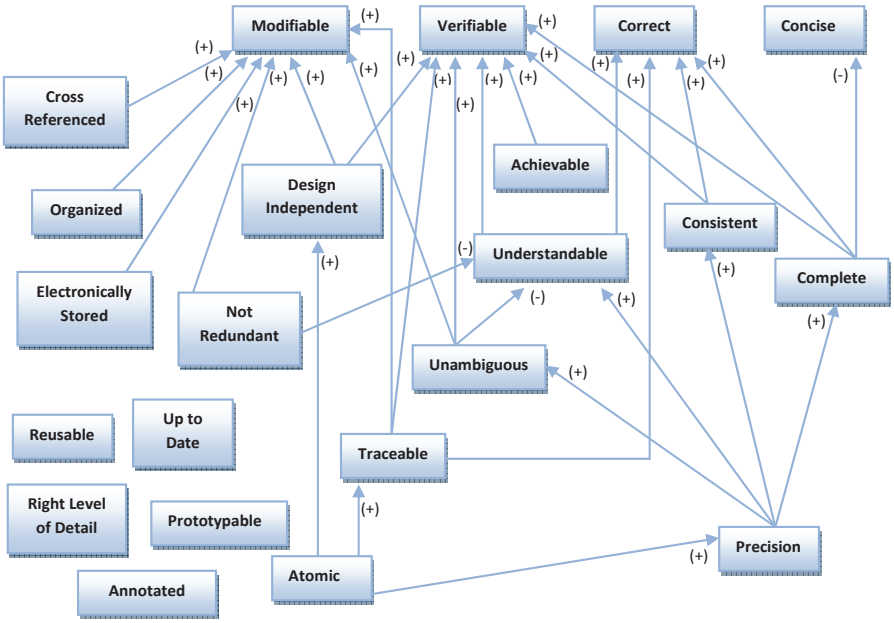


Fig. 1. Influence between quality properties.

- It is not always possible to significantly reduce the SRS size without negatively affecting other quality attributes [2].
- If a SRS is inconsistent, then the client cannot confirm that requirements effectively express the system that responds to their needs, i.e., its correctness [10].
- A SRS is more modifiable if it is traceable, organized, cross-referenced and stored electronically [2].
- Traceability facilitates correctness [5] and verifiability [10].
- Requirements not understood are not verifiable and its correctness may not be validated [10].
- The redundancy is often used to increase readability of the SRS. Furthermore, redundancy can negatively affect the modifiability, since not changing all occurrences of a redundant requirement generates an inconsistent SRS [2].
- A non-atomic requirement has the risk of excessive detail, and thus, may affect design independence [10].
- Atomicity influences requirements precision and traceability [10].
- Design independence can help verifiability. The presence of requirements technical details is more difficult to verify and modify [10].
- A more precise language helps to write more complete, consistent, understandable and unambiguous requirements [10].

Figure 1 resumes the influences among quality attributes previously described. In addition, it can be noted that no dependencies can be found for some attributes, due to

it is possible to reach them without affecting other attributes. They are: Reusable, Up to Date, Annotated, Prototypable and At Right Level of Detail.

6 Conclusions

In this work, an exploratory analysis of various proposals for quality attributes evaluation of a SRS was performed. On the one hand, there exist different quality models for SRS. In the ones analyzed, some characteristics are worth to note. Firstly, some of the attributes are used by most models, for example, unambiguous and complete. Secondly, there are attributes which are considered separately by some authors, while others use them together, aggregated in a single attribute, e.g., Traced and Traceable. Additionally, other attributes, although being the same, are differently referenced by some authors: for example, Independent Design is also called Abstraction. Finally, some attributes are only used by the author who defines it, for example, Up to Date. Furthermore, it is clear that some attributes are applicable to the entire SRS, while others apply only to individual requirements.

On the other hand, there are many proposals describing approaches for SRS quality attributes evaluation. Some of them are only conceptual and, therefore, difficult to automate, since they require a high degree of human reviewers intervention. Other proposals describe tools implementations in different technologies that pose heuristics or metrics for just some quality attributes. Usually, this assessment only can be achieved with the technology used and, in many cases, does not cover all possible cases or have certain shortcomings that require optimization.

To conclude, is important to note that the definition of heuristics, metrics and/or objectively measurable indicators is needed in order to cover all possible cases for requirements and SRS quality attributes evaluation in a software project. It is also required the construction of a supporting tool for implementing these defined heuristics, metrics and indicators, so an automatically assessing can be performed for obtaining quality SRS. This constitutes the main goal for future research in this area.

7 References

1. Nuseibeh, B.; Easterbrook, S.: Requirements engineering: a roadmap. In: Proc. Conference on The Future of Software Engineering, pp. 35-46. (2000).
2. Davis, A.; Overmyer, S.; Jordan, K.; Caruso, J.; Dandashi, F.; Dinh, A.; Kincaid, G.; Ledeboer, G.; Reynolds, P.; Sitaram, P.; Ta, A.; Theofanos, M.: Identifying and measuring quality in a software requirements specification. In: Proc. 1st International Software Metrics Symposium, pp. 141-152. (1993).
3. Fabbrini, F.; Fusani, M.; Gnesi, S.; Lami, G.: An Automatic Quality Evaluation for Natural Language Requirements. In: Proc. 7th International Workshop on Requirements Engineering: Foundation for Software Quality, Interlaken, Switzerland. (2001).
4. Gnesi, S.: Analysis of Software Requirements. IEI-CNR Pisa http://www.iei.pi.cnr.it/ERI/iei/qmslideseri.ppt (2000).
5. IEEE Recommended Practice for Software Requirements Specifications. IEEE Standard 830-1998 (R2009), Institute of Electrical and Electronics Engineers. (2009).

6. Swathi, G.; Jagan, A.; Prasad, Ch: Writing Software Requirements Specification Quality Requirements: An Approach to Manage Requirements Volatility. Int. J. Comp. Tech. Appl., **2**(3), 631-638. (2011).
7. Pohl, K.: Requirements Engineering: Fundamentals, Principles, and Techniques. Springer-Verlag Berlin Heidelberg. (2010).
8. Wiegers, K.: Software Requirements, Second Edition. Microsoft Press. (2003).
9. Loucopoulos, P.; Karakostas, V.: System Requirements Engineering. McGraw-Hill, Inc. New York, NY, USA. (1995).
10. Génova, G.; Fuentes, J.M.; Llorens, J.; Hurtado, O.; Moreno, V.: A Framework to Measure and Improve the Quality of Textual Requirements. Requirements Engineering, **18**(1), pp 25-41 (2013).
11. Tjong, S.F.: Avoiding Ambiguity in Requirements Specifications. PhD Thesis. University of Nottingham Malaysia Campus, Faculty of Engineering & Computer Science, Malaysia. (2008).
12. Durán, A.; Bernárdez, B.; Ruiz, A.; Toro, M.: An XML–based Approach for the Automatic Verification of Software Requirements Specifications. In: Proc. 4th Workshop on Requirements Engineering, pp. 181-194. (2001).
13. Durán, A.; Ruiz-Cortés, A.; Corchuelo, R.; Toro, M.: Supporting requirements verification using XSLT. In: Proc. IEEE Joint International Conference on Requirements Engineering, pp. 165–172. (2002).
14. Wilson, W.M.: Writing Effective Requirements Specifications. Software Technology Conference Proceedings. (1997).
15. Kaiya, H.; Saeki, M.: Ontology Based Requirements Analysis: Lightweight Semantic Processing Approach. In: Proc. 5th International Conference on Quality Software, pp. 223-230. (2005).
16. Kaiya, H.; Saeki, M.: Using Domain Ontology as Domain Knowledge for Requirements Elicitation. In: Proc. 14th IEEE International Requirements Engineering Conference, pp. 189-198. (2006).
17. Dzung, D.; Ohnishi, A.: Ontology-based Reasoning in Requirements Elicitation. 7th IEEE Int. Conf. on Software Engineering and Formal Methods, pp. 263-272. (2009).
18. Hu, H.; Zhang, L.: Ye, C.: Semantic-based Requirements Analysis and Verification. In: International Conference on Electronics and Information Engineering, pp. 241-246. (2010).
19. Verma, K.; Kass, A.: Requirements Analysis Tool: A Tool for Automatically Analyzing Software Requirements Documents. In: Proc. 7th International Conference on The Semantic Web, pp. 751–763. (2008).
20. Ali, M.J.: Metrics for Requirements Engineering. Master's Thesis. Umea Univ. (2006).
21. Rosenberg, L.; Hammer, T.; Huffman, L.: Requirements, testing, and metrics. In: 16th Pacific Northwest Software Quality Conference, Utah, USA. (1998).

Analyzing the Use of an Enterprise Model as a Stakeholder Requirements Model: An Experiment

Fábio Levy Siqueira[1, 2] 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. There are several possible representations for stakeholder requirements, such as goal models, scenarios, and natural language. This paper analyzes the use of an enterprise model as a stakeholder requirements model. We conducted a quasi-experiment with 29 graduate students who received either a textual problem statement or an enterprise model, representing the stakeholder requirements, and refined it into use cases, representing the software requirements. The subjects were evaluated considering the use case quality and the time spent, for two different scopes. The results indicate that the mean quality of the use cases created using an enterprise model was equal to or greater than the mean quality of the use cases created by the other groups. In addition, the mean time spent to create a use case using an enterprise model was equal to or less than the mean time needed for the other groups.

Keywords: stakeholder requirements, enterprise model, use case, experiment.

1 Introduction

ISO 29418 [6] differentiate three levels of requirements: stakeholder requirements, system requirements, and software requirements. Stakeholder intentions (needs, goals, and objectives) are refined into stakeholder requirements that describe "the intended interaction the system will have with its operational environment" [6, p.19]. These requirements are transformed by a requirements engineer into system requirements, based on the organization, the environment, and other constraints (this process is called refinement by Jackson and Zave [13]). System requirements represent a technical specification for the system. Therefore, they must be measurable, but they must avoid describing implementation details. Some system requirements may be allocated into a software element of the system, resulting in software requirements.

There are several possible representations for stakeholder requirements. For instance, stakeholder requirements can be represented using goal models, scenarios, or natural language. Another possible representation of stakeholder requirements is an enterprise model, describing the environment as the stakeholders would like it to be [11]. This possibility seems to be even more appropriate given the content of the stakeholder requirements specification (StRS) defined in ISO 29148 [6]. Even though

an enterprise model may not contain all the StRS information, it could be used as a partial description of the stakeholder requirements.

In order to analyze the use of an enterprise model as a stakeholder requirements model, this paper presents a quasi-experiment – i.e., an experiment in which the participants are not randomly assigned to the treatments. It considers software-intensive systems, in which software requirements are almost all the system requirements. The experiment investigates the impacts of using this model during stakeholder requirements refinement, analyzing the quality of the resultant software requirements and the time spent to create it. Among the possible representations of software requirements we selected use cases due to its popularity. Therefore, the experiment evaluates use case quality and the time spent to generate it in order to analyze the adequacy of using an enterprise model during requirements refinement.

The paper is structured as follows: Section 2 presents the related work. Section 3 presents the experiment planning. Next, in Section 4, it is presented the experiment results and analysis. Finally, in Section 5 we discuss the results and present our conclusions, including proposals for future works.

2 Related Work

Few approaches propose using an enterprise model as a stakeholder requirements model, for instance the approaches proposed by De La Vara, Sanchéz and Pastor [4], and by Molina et al. [7]. These approaches propose methods to manually refine enterprise models into use cases, considered as a software requirements model. However, they do not analyze experimentally their method.

There are some experiments that evaluate the use case quality, in general to investigate the use of guidelines in use case authoring [1, 2, 3, 9]. Consequently, these studies analyze the approach used to generate the use cases – and not the model used as input, as described here. A similar experiment that evaluates the input model is presented by Yue, Briand, and Labiche [12]. It evaluates the quality of the analysis model created from use cases in order to analyze an approach for use case authoring. However, it analyses a different task in the software development process.

3 Experiment Planning

The experiment's goal is to analyze if the use of an enterprise model as a stakeholder requirements model affects the quality of the resultant use case model, used as software requirements, and the time spent to create it. Based on this goal, we formulated the following hypotheses:

H1. The quality of a use case obtained by refining an enterprise model is equal to or greater than the quality of the use case obtained considering a textual stakeholder requirements model.

H2. The time spent to create a use case model by refining an enterprise model is equal to or less than the time spent considering a textual stakeholder requirements model.

Therefore, if both hypotheses are true, then it can be concluded that there are no negative impacts in using an enterprise model as a stakeholder requirements model.

3.1 Context

The experiment was conducted with graduate students of the Software Technology course at the Continued Education Program in Engineering (PECE) of the Polytechnic School of the University of São Paulo (USP). The subjects were 29 students at the last period of the course. They had already concluded a Requirements Engineering discipline, which covered use case modeling. The students also had an overview of business process modeling during another discipline. The experiment was conducted during a discipline, but the results were not used to grade the subjects.

3.2 Experiment Design and Operation

Table 1 presents the experiment design considering the stakeholder requirements model used and the project scope, which were chosen as factors. The factor "stakeholder requirements model" was divided into three groups: *Textual* (using a textual problem statement); *Enterprise Model* (using an enterprise model created by another subject); and *Both* (creating first an enterprise model and then the use cases, thus using both stakeholder models). The other factor, the project scope ("Project"), considered two alternatives: a *Bank Teller* (*ATM*) and a *Video Rental Store* (*Video Store*). When possible, the values and the overall experiment were based on similar experiments [1, 2, 3, 9, 12], thus allowing future comparisons.

Table 1. The experiment design and the number of subjects for each treatment.

		Stakeholder Requirements Model		
		Textual	Enterprise Model	Both
Project	ATM	5	5	4
	Video Store	5	5	4

First, all subjects watched a 10 minutes presentation about business process modeling. Then, they answered an initial questionnaire covering the independent variables regarding the knowledge and the experience of the subjects.

The subjects were divided into 2 groups. The Group 1 corresponds to the subjects for *Textual* and *Enterprise Model*, and the Group 2 corresponds to subjects who created the enterprise model. As there was no time available to evaluate the subjects' responses to the initial questionnaire, they were assigned to these groups considering their seating arrangement in the classroom. The subjects for the Group 1 received a problem statement presenting textually the stakeholder requirements. They were requested to create a use case in one hour. The subjects for the Group 2 received the same problem statement as the other group and a description of the environment as-is (which is domain knowledge). They were requested to create the enterprise model as-is and to-be in one hour.

In a different day, the subjects from the Group 1 received an enterprise model created by another subject and considering a different project scope from the previous activity. They were requested to create a use case in one hour. Part of Group 2 received the enterprise model he/she created, and the original problem statement. They were requested to create a use case in one hour.

The models (use case and enterprise) were created manually by the subjects. The enterprise model was created considering a simplified version of the metamodel described in [11], while the use cases were described using a template based on the meta-model proposed in [10]. As some subjects needed to use the enterprise model created by another subject, the experimenter used a tool [11] to represent all the enterprise models created, in order to avoid influences or misunderstandings.

Based on the experimental hypotheses, the results of the experiment are evaluated by two dependent variables: the quality of the use case created, and the time spent to create the use case. To evaluate the use case quality, this work uses the seven quality factors to help use case communication, proposed by Phalp, Vincent, and Cox [8]. In order to use a more objective evaluation, we defined criteria for each quality factor and sub-factor. Each sub-factor is graded from 0 to 5, and the maximum grade is 65, as there are 13 sub-factors.

4 Experiment Results and Analysis

On average, the subjects needed 46 minutes to create the use cases in the *ATM* scope, obtaining a score of 60.47. On the *Video Store* scope, the average time spent was 42 minutes and the average score 61.49. Analyzing the data, it was found two outliers, which were discarded from the analysis.

Each hypothesis will be analyzed separately for each scope and control group (*Textual* and *Both*). As a result, eight experimental hypotheses were formulated. They will be tested using a Wilcoxon test [5], with a 0.05 significance level (α).

Table 2 presents the summary of the Wilcoxon test results for the four experimental hypotheses regarding the use case quality (H1). All the alternative experimental hypotheses compare if the mean quality of the group *Enterprise Model* (μ_{QE}) is less than the quality of the groups *Textual* or *Both* (μ_{QT} and μ_{QB}, respectively). Considering the format of the Wilcoxon test presented in [5], the null hypothesis of HE1 and HE2 can be rejected if the rank sum is less than or equal to the rejection value. On the other hand, the null hypothesis of HE3 and HE4 can be rejected if the rank sum is greater than or equal to the rejection value.

Table 2. Results of Wilcoxon test for hypothesis H1.

Experimental Hypothesis	Project Scope	Null Hypothesis	Alternative Hypothesis	Rank Sum	Rejection Value	Reject Null hypothesis
HE1	ATM	$\mu_{QE} = \mu_{QT}$	$\mu_{QE} < \mu_{QT}$	27	13	No
HE2		$\mu_{QE} = \mu_{QB}$	$\mu_{QE} < \mu_{QB}$	17	12	No
HE3	Video Store	$\mu_{QT} = \mu_{QE}$	$\mu_{QT} > \mu_{QE}$	21	27	No
HE4		$\mu_{QB} = \mu_{QE}$	$\mu_{QB} > \mu_{QE}$	22	27	No

It is not possible to reject the null hypothesis for all experimental hypotheses. Therefore, the hypothesis H1 cannot be rejected. In other words, it indicates that the use of an enterprise model as a stakeholder requirements model did not resulted to lower quality use cases.

Table 3 presents the summary of the Wilcoxon test results for the four experimental hypotheses regarding the time spent. The alternative hypotheses consider that it is faster (demands less time) to create use cases for *Textual* and *Both*. For the group *Both*, only the time spent creating the use case was considered. The null hypothesis of HE5 and HE6 can be rejected if the rank sum is greater than or equal to the rejection value. On the other hand, the null hypothesis of HE7 and HE8 can be rejected if the rank sum is less than or equal to the rejection value.

Table 3. Results of Wilcoxon test for hypothesis H2.

Experimental Hypothesis	Project Scope	Null Hypothesis	Alternative Hypothesis	Rank Sum	Rejection Value	Reject Null hypothesis
HE5	ATM	$\mu_{TE} = \mu_{TT}$	$\mu_{TE} > \mu_{TT}$	10	27	No
HE6		$\mu_{TE} = \mu_{TB}$	$\mu_{TE} > \mu_{TB}$	11	24	No
HE7	Video Store	$\mu_{TT} = \mu_{TE}$	$\mu_{TT} < \mu_{TE}$	19	13	No
HE8		$\mu_{TB} = \mu_{TE}$	$\mu_{TB} < \mu_{TE}$	24	13	No

None of the four null hypotheses could be rejected. It indicates that the use of an enterprise model as a stakeholder requirements model did not demanded more time to create the use cases. That is, hypothesis H2 cannot be rejected.

5 Discussion and Conclusion

This study presented an experiment to analyze the use of an enterprise model as a stakeholder requirements model. The analysis considered the quality and the time spent to refine an enterprise model into software requirements, which were represented with use cases. The results obtained by subjects who created the use cases based only on an enterprise model (group *Enterprise Model*) were compared with the results obtained by two groups: subjects who received a textual problem statement (group *Textual*), and subjects who received both a textual problem statement and an enterprise model they created earlier (group *Both*). Additionally, experiment had two scopes, *ATM* and *Video Store*.

The results indicate no negative effect in the use case quality when using an enterprise model. For both scopes the mean quality of the use cases was equal to or greater than the mean quality of the use cases created by the other groups. The results also indicate no negative effect in the time spent when using an enterprise model as a stakeholder requirements model. For both scopes the mean time spent to create a use case using an enterprise model was equal to or less than the mean time needed considering a textual problem statement. Nevertheless, there are threats to the validity of this experiment. The most important threats are the small statistical power and some limitations of the experiment setting (time constraint, the scopes, the use of students, and the generation of only one use case). Another important threat is that the

subjects for the treatment *Textual* did not receive information about the environment as-is, as it was not considered useful for their activity.

Other analysis must be done before recommending the use of an enterprise model as a stakeholder requirements model in real projects. Other experiments must analyze if these results are true when generating more than one use case (a complete use case model) and real scopes. In addition, a case study or an experiment analyzing an industrial setting must be executed. Anyhow, the results obtained in this experiment motivate further analysis.

Acknowledgements

This work was partly supported by Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), grant no. 2007/58489-4.

References

1. Achour, C.B., Rolland, C., Maiden, N.A.M., and Souveyet, C.: Guiding use case authoring: results of an empirical study. In: IEEE International Symposium on Requirements Engineering, pp.36-43 (1999)
2. Anda, B., Sjøberg, D., and Jørgensen, M.: Quality and Understandability of Use Case Models. In: ECOOP, LNCS 2072, pp.402-428, (2001)
3. Cox, K. and Phalp, K.: Replicating the CREWS Use Case Authoring Guidelines Experiment. Empirical Software Engineering, v.5, n.3, pp.245-267 (2000)
4. De La Vara, J.L., Sanchéz, J, and Pastor, O.: Business Process Modelling and Purpose Analysis for Requirements Analysis of Information Systems. In: CAiSE, LNCS 5074, pp. 213-227 (2008)
5. Devore, J.L.: Probability and Statistics for Engineering and the Sciences. 6th ed., Duxbury Press (2003)
6. ISO: Systems and software engineering - Life cycle processes - Requirements engineering, ISO/IEC/IEEE 29148 (2011)
7. Molina, J.G., Ortín, M.J., Moros, B., Nicolás, J., and Toval, A.: Towards Use Case and Conceptual Models through Business Modeling. In: ER, LNCS 1920, pp.281-294 (2000)
8. Phalp, K.T., Vincent, J., and Cox, K.: Assessing the quality of use case descriptions. Software Quality Journal, v.15, n.1, pp.69-97 (2007)
9. Phalp, K.T., Vincent, J., and Cox, K.: Improving the Quality of Use Case Descriptions: Empirical Assessment of Writing Guidelines. Software Quality Journal, v.15, n.4, pp.383-399, (2007)
10. Siqueira, F.L. and Muniz Silva, P.S.: An Essential Textual Use Case Meta-model Based on an Analysis of Existing Proposals. In: WER, pp. 419-430 (2011)
11. Siqueira, F.L. and Muniz Silva, P.S.: Using an Enterprise Model as a Requirements Model in Process Automation Systems: A Proposal. In: CONTECSI, pp.3064-3088 (2011)
12. Yue, T., Briand, L.C., and Labiche, Y.: A Use Case Modeling Approach to Facilitate the Transition towards Analysis Models: Concepts and Empirical Evaluation. In: MODELS, LNCS 5795, pp.484-498 (2009)
13. Zave, P. and Jackson, M.: Four Dark Corners of Requirements Engineering. ACM Transactions on Software Engineering and Methodology, v.6, n.1, pp.1-30 (1997)

Introducing Use Cases in a small Organization: An Experience and Lessons Learned[1]

Dolors Costal, Xavier Franch, Nuria Rodríguez, Luis Delgado, Carme Jiménez

Universitat Politècnica de Catalunya (UPC), Barcelona (Spain)
{dolors|franch}@essi.upc.edu
{nuria.rodriguez-camara|luis.delgado|carme.jimenez}@upc.edu

Abstract. In this paper we report the adoption of use cases by a small organization in a university setting. Use cases were first introduced in the middle of a huge project and adopted thereafter for later projects. The paper mostly focuses in the first experience, whose most interesting characteristics were the large size of the resulting specification, the fact that it took place once the project had started (for documentation purposes instead of driving the development) and the limitation that resources allocated were not as much as required. We present the lessons learned from this experience.

1 Introduction

Use cases [1, 2] are widely accepted artefacts for describing the functional behaviour of a system. As a particular model of the specification, use cases are supposed to be defined in advance to the system itself (or more accurately, from an iterative point of view, a functionality should not be implemented before the use case that describes it).

We report an experience of adoption of use cases in a small organization. Use cases were introduced in the context of a particular project where the assumptions above failed. The undertaken project was the specification, for documentation purposes, of a large system, and it was developed in an organization that never before had used scenarios or use cases. The specification started more than one year after a first prototype of the system was deployed. Due to scheduling and budget constraints, resources were not allocated as required. We show which concrete problems appeared, and how they were tackled. We finally give some lessons learned.

The project ran at the PRISMA organization, a UPC unit focused on delivering software for the university. PRISMA, with 23 employees, is a typical example of small organization. Its first mission was to produce the PRISMA information system for providing an integrated management of academic data. A quality team was constituted (the authors of this paper). It was composed of two groups, one coming from the PRISMA organization itself, and the other by some researchers coming from the Software Engineering for Information Systems research group (GESSI) at the UPC. The collaboration embraced several activities, and building some UML artefacts [3] was one of them.

[1] This work has been partly supported by the *Ministerio de Ciencia y Tecnologia* and FEDER under project TIN2010-19130-C02-01.

2 Departing Situation and Decisions Made

Developing a UML documentation was not scheduled from the beginning of the quality assessment activity. The project manager took the decision of using UML for four purposes: (1) to validate systematically the suitability of processes and data, (2) to create a high-level documentation for future maintenance of the system, (3) to provide a view understandable for other prospective users, and (4) to obtain a "UML-label", considered a strategic goal by PRISMA managers for disseminating and trying to sell the product to other universities.

The use case specification was a milestone of this UML documentation. Due to the size of the project, it was foreseen to be a challenging activity. The use case specification started once a first prototype of the system was already running in some pilot schools. At that time, the PRISMA software engineering group was developing new functionalities and supporting the pilot schools. Due to the small size of the organization, the use case developer role was played by software developers. They had less time than required to write the use cases and they did not perceive use cases specification as important as implementation.

Due to these conditions, we (i.e., the quality team) took the following decisions:

- We prioritized the use cases because the manpower allocated to this task was not enough to develop all of them in detail. We considered as prioritisation units the modules that were already defined as PRISMA main functional areas.
- We wrote a very prescriptive use case template. We analysed some classical, well-established proposals (remarkably [4, 5, 6]), and developed a style sheet.
- We defined a methodology to develop the use cases, with special emphasis in stepwise refinement and linkage with the requirements management part.
- We designed tool support for the methodology. This tool support put together the previously selected Visure Requirements tool, which was used to write the use case diagrams, with Lotus Notes for writing the individual use cases.
- We designed a procedure for requirements change management. The procedure covered both the organizational side (e.g., who is in charge of doing what) and the technical side (e.g., how to use the tools to ensure consistency).

In this paper, we will focus on the first two issues.

3 Use Case Prioritization

The first decision we took was to divide the use case specification activity into two phases. The first phase, within project development yet, consisted in documenting some designated modules, whilst the second one aimed at finishing the specification during the first months of the project maintenance phase.

During the first phase, the prioritization criteria we used were: significance of the addressed functional areas and, within them, those functionalities that were more stable. Both "significance" and "stability" are fuzzy concepts, therefore we undertook a joint analysis with the development manager. Four modules were designated as the most significant because of their highly strategic nature and their large size: *Students and Files*, *Registration*, *Evaluations* and *Economic Management*. Between them, the

first two were pointed out as the most stable. We were in charge of developing the specification for them, interacting with the developers as questions arose. This kind of smooth interaction turned out to be very important for overcoming the barrier among developers and the quality team (see Section 5).

Once the running system became more stable and developers more available, the second phase started. Then, the rest of functional areas (i.e. *University Programs Management*, *Undergraduate Final Projects*, *Teaching Management* and *Information Management*) were documented in parallel by the developers under our direction.

4 Use Case Template

We proposed a template to establish the items that have to be included in a use case description. Together with the template we had the purpose of providing a clear definition of its key concepts, such as primary actor, stakeholder, etc., to facilitate its use by developers and create a common understanding in the organization. Altogether was encoded in a conceptual model with a glossary, which in fact can be considered a kind of ontology for our use cases.

In Fig. 1 we present a simple use case example from the PRISMA project. This use case has several extensions though only one is included below. The template is mostly based on Cockburn's fully dressed use case format [4] with the exception of the last two items: use case data definitions and auxiliary definitions.

Use case data definitions simplify the description and maintenance of use case scenarios where some long list of related data is referred to, by allowing the definition of composite data. In the PRISMA project, the data was synthesised basically from the data base definition and the user interface, which at some times exhibit some discrepancies; building the use cases was a good opportunity for detecting and solving them. Enclosing data definition in a separated data definition section facilitates the quick reference to this set of elements and increases understandability and modifiability of the use case while reducing its size. In addition, it facilitates maintenance because, if there is any change in personal data elements, a single definition has to be modified. In a similar way, auxiliary definitions simplify the description of use case steps where a complex calculation is performed or a complex term is used. For instance, in the first category falls the computation of the final mark of a student, a weighted sum taking into account marks, credits, type of course, etc. These two components are not part of the use case itself, they are separated and therefore may be referred from several use cases. References to them are underlined.

Use case understandability and independence from GUI were considered of special relevance for the purposes of the project. We considered many recommendations for use case development, most of them extracted from [4], to fulfil these goals. We found especially useful those related with lexical and syntactic conventions.

Use Case Name: Modify the data of a student.
Primary Actor: School.
Scope: Academic Record Management System.
Stakeholders and interests:
School: Wants to have available up-to-date data from its students.
Student: Wants the School have their data up-to-date.
Wants no one else gain access to their data.
Precondition: None.
Minimal guarantees:
When a student data is updated, his/her student identifier is not updated in such a way that another student with the same student identifier exists in the system
Success guarantees: Student's data has been updated.
Trigger: The School has a change request form filled by a Student.
Main success scenario:
1. The School indicates to the System the search identifier of the student whose data has to be updated.
2. The System shows to the School the Student personal data and Student access data.
3. The School indicates to the System the updates of these data of the Student.
4. The System registers the updated data of the student.

Extension: Wrong student identifier update.
Condition: The School has indicated an update of the Student identifier and there already exists another Student with the indicated new value.
Step: 4
Scenario:
1. The System communicates to the School that the Student identifier cannot be updated because there already exists another Student with the indicated new Student identifier.
2. Go to step 3 from main success scenario.

Use Case Data:
Student personal data: DNI, entry date, surname, name, birth date, country of birth, nationality, family address, address during semester, username, password, email.
Access data: access year, access semester, title, access date, access type, access mark.

Auxiliary definitions:
Search identifier of the student: DNI, passport or name and surname.
Student identifier: DNI or passport

Fig. 1. Example of use case

5 Lessons Learned

We summarise in this section some lessons learned that may be considered the scientific outcome of our experience and that we summarize in this section. We have focused on lessons about: 1) criteria that affect the general appearance of use cases; 2) project management with use cases; 3) technical aspects of use case definition.

Lessons on use case design criteria:

Lesson 1. Do not get rid completely of the existing implementation.

- It is necessary to take into account the general structure of the solution (not the user interface).

- Define packages in the use case model which correspond to main system areas. This makes easier the involvement of the development people.
- Use navigational maps to identify includes and alternative courses.
- Use forms to check whether all relevant data is included in the specification.

Lesson 2. Give preference to robustness over accuracy for dealing with small variations of business processes.

- Useful in cases such as ours, in which different units of an organization have essentially the same business processes but differ slightly in their implementation.
- By using abstract actors, the number of use cases dramatically decreases, because it is not necessary to differentiate the several variants.
- Future changes in these details may not imply changes in the use case template.

Lessons on project management:

Lesson 3. Prioritise use cases if resources are insufficient.

- When it is obvious that resources are not enough to develop all use cases, which is often the case for small organizations, it is preferable to perform an explicit task of prioritization than to leave this undefined.
- Define criteria for the prioritization of use cases. It is important to provide a rationale for prioritization decisions. In our experience, strategic relevance of use cases, stability and size were used (see Section 3). Criteria should be as objective and measurable as possible.
- Group use cases in prioritization units and apply criteria.

Lesson 4. Do not start to write the use case specification until a template has been defined. Otherwise, chances are that some work has to be redone.

- The large size requires a uniform style to write, learn, validate, compare, etc., in an easy way.
- Skills are acquired more quickly and productivity increases.
- Facilitates the writing of a checklist that makes possible the verification of the use case properties.

The template shall define both the parts of the use case and syntactic conventions.

Lesson 5. Introduce use cases smoothly in the organization. In a small organization, introducing a new activity or artefact is especially tough.

- Try to involve developers by encouragement instead of obligation. Try to find quickly some advantage of using use cases applied to a developer problem.
- Provide templates, methods, procedures, etc., in a very plain way with well-defined steps and parts.
- Do not force the developers to learn new tools further than necessary.

Lesson 6. Monitor continuously use case models to reduce risks. Among the most characteristic issues to monitor, we have:

- The need of abstracting from the user interface details, for avoiding commitment to a specific style of data communication.
- The need of distinguishing the scope, because system requirements do not focus on actor-actor interactions, which become of interest when writing use cases.

- Writing use cases in parallel with system implementation is hard. Functionalities evolve, some parts (e.g., data forms) are not updated timely, etc. Therefore validation is continuously needed.

Lessons on use cases technical aspects: We emphasize which are the most critical aspects (from those described in Section 4) that contribute to use case understandability and to save use case development resources.

Lesson 7. Introduce mechanisms to define the structure of system data and a glossary for domain-specific terms.

- Makes the specification self-contained by encoding knowledge about the domain.
- Ensures coherence among different use cases.
- Makes use cases text (and thus their development time) shorter.
- Supports validation with existent GUI forms.
- Supports maintenance with respect to changes in data or domain definition.

Lesson 8. Do not abuse of use case relationships.

- With large specifications, too many relationships make the structure of the specification difficult to understand.
- Workflows referred to change management get complicated: a change on a requirement makes it difficult to detect all the use cases that are affected.
- They raise some minor technical points: linkage among the included and the one that includes, incorrect use of preconditions, etc.

6 Conclusions

We have presented some relevant issues of the process followed for introducing use cases in a small organization. The experience can be considered successful. A very complete use case model was built for the first project fulfilling the goals enumerated in Section 2. The developers inside the PRISMA organization were a bit sceptical with respect to use case modelling. However, we overcame the difficulties found during this long process and at the end they found use cases useful. The organization recognized the convenience of use case models and they are using them in the ongoing projects.

References

1. I. Jacobson, M. Christerson, P. Johnson, G. Övergaard. *Object-Oriented Software Engineering, a Use-Case Driven Approach*. Addison Wesley, 1992.
2. I. Jacobson. *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison Wesley, 2004.
3. G. Booch, J. Rumbaugh, I. Jacobson. *The Unified Modeling Language User Guide* (2nd edition). Addison Wesley, 2004.
4. A. Cockburn, *Writing Effective Use Cases*. Addison Wesley, 2002.
5. S. Adolph, P. Bramble. *Patterns for Effective Use Cases*. Addison Wesley, 2004.
6. I. Alexander, N. Maiden (eds.). *Scenarios, Stories, Use Cases: Through the Systems Development Life-Cycle*. John Wiley and Sons Ltd, 2004.

GT4CCI: An Approach Based on Grounded Theory for Crosscutting Concerns Identification in Requirements Documents

Larissa de Alencar Sobral, Lyrene Fernandes da Silva

Departamento de Informática e Matemática Aplicada (DIMAp)
Universidade Federal do Rio Grande do Norte
lariasobral@gmail.com, lyrene@dimap.ufrn.br

Abstract. When crosscutting concerns identification is performed on the activities involved in requirements engineering there are many gains in terms of quality, cost and efficiency throughout the lifecycle of software development. However, despite these gains, this identification faces several difficulties such as the lack of systematization and tools that support it and the difficult to justify why some concerns are identified as crosscutting or not, since this identification is often made without any methodology that systematizes and bases it. In this context, this paper proposes and evaluates an approach based on Grounded Theory, called GT4CCI, for systematizing the process of identifying crosscutting concerns in requirements document. Through the use of GT4CCI it is possible to better modularize the requirements document, make it more consistent, detect possible failures and improve traceability among requirements, adding significant gains in terms of quality and reliability to crosscutting concerns identification and to requirements engineering.

Keywords. Grounded Theory, Crosscutting Concerns Identification, Software Modularity, GT4CCI, Crosscutting Concerns Approach.

1 Introduction

According to [1], identifying and documenting crosscutting concerns in the beginning of the software lifecycle, in phases involved in requirements engineering, is essential. This action provides significant improvement in requirements traceability, facilitates the evaluation of the impact of changes in the system, facilitates requirements evolution and improves the modularization of the system, among other advantages. According to [2], despite all these favorable points, the crosscutting concerns identification in the initial stages of the software development process has been neglected in most software projects. This neglecting is mainly caused by the absence of habit of applying this kind of identification and by the lack of methodologies that basis this identification.

Considering the difficulties in identifying crosscutting concerns during the requirements engineering and, more importantly, the need to justify why some concerns

are considered crosscutting, we present a new approach, called GT4CCI, which organizes and supports this process, making it more grounded. GT4CCI is based on the analysis process proposed by Grounded Theory [9], a renowned and well known methodology originate from the social sciences that enable qualitative analysis of data by codifying these data. The use of Grounded Theory (GT) adds the following significant gains in the process of crosscutting concerns identification:

- GT is based on contextual analysis of data, making analysis more complete and consistent, since it takes into account the context in which the concern is embedded;
- GT bases its results on the data existing on the document analyzed, consequently the results obtained through its use can be easily traced in this document;
- GT does not limit its use to documents that are previously structured in accordance with established standards. This means that GT allows the analysis of any requirements document already developed;
- GT does not restrict the data to be analyzed. This means that analysis is not limited to one type of concern, for example non-functional requirements or use cases, ignoring the other concerns present in the document. The use of GT allows the analysis of any relevant item, or part of the requirements document;
- There are tools that support to the process proposed by GT and that automate some of its activities. In this work, we have used Atlas.ti Tool [10].

GT4CCI approach uses the requirements document as input for the analysis and identification of crosscutting concerns. Thus, the main goal of GT4CCI is to systematize and improve the process of identifying these concerns. By the use of this approach is believed that there are gains in highlighting the intrinsic complexity of some concerns, which are scattered and tangled on requirements document, and warning the analysts that it is necessary to analyze these concerns more carefully.

Some approaches, such as Theme/Doc [3], DISCERN [2], Early-AIM [7] and CCCINPL [8], have been developed for the purpose of systematizing the identification of crosscutting concerns on the more initial steps of the software development process. However, these approaches have some limitations, best exposed in section 5 of this work. Consequently, GT4CCI is an alternative to these approaches, since the gain brought by the use of Grounded Theory decreases some of these limitations.

In order to present the GT4CCI approach, this paper is organized as follows. Section 2 presents a Toy Example, which serves as a basis example for demonstrating the use of GT4CCI. Section 3 presents the GT4CCI approach, detailing each of its steps and presenting the results generated from the use of the approach in the Toy Example. Section 4 presents an experimental study applied to evaluate the approach GT4CCI. Section 5 presents and compares some related work and GT4CCI. Finally, Section 6 contains final remarks and future work.

2 A Toy Example: Crisis Management Systems

In order to facilitate understanding and to demonstrate the process defined by the GT4CCI approach, the requirements document of Crisis Management Systems [11] is used as a Toy Example in the section below.

Crisis Management Systems (CMS) is a requirements document defined as a standard case study used by researchers on aspect- oriented modeling. This document was firstly defined and used in the Transactions on Aspect-Oriented Software Development VII (TAOSD), published in 2010 [11]. CMS describes the requirements and defines a system that helps identify, assess and deal with crisis situations, allowing communication between all parties involved in managing the crisis. This is done through the allocation and management of resources and also through access to information concerning the crisis, made by allowed users.

This requirements document is composed of eight sections. The first section presents an overview of the system. The second section describes all requirements for this system. Section 3 presents the feature model. In Section 4 are presented all use cases involved in the system. The following sections of this document present the Domain Model, the Activity Diagrams and the Informal Physical Architecture Description. Finally, in the last section, the Selected Design Models are presented.

The data considered most important in the document CMS were analyzed and coded by the GT4CCI approach in order to make the identification of crosscutting concerns. Thus, special attention was given to the sections relating to the description of use cases and detailed description of system requirements, since they are more geared to the objective of this work, containing the information about the requirements of this system. A part of the results from the application of GT4CCI in the Toy Example are presented and discussed in the following sections.

3 GT4CCI Approach

GT4CCI – acronym to Grounded Theory for Crosscutting Concerns Identification - is an approach based on the process of collection, analysis and data coding proposed by Grounded Theory (GT). GT is a methodology, arising from the social sciences, which is based on qualitative analysis of data and data coding in order to determine the relationships among them. Following and adapting the process proposed by this methodology, it is possible to extract information that facilitates the establishment of relations between relevant points of a requirements document. As a result, it is possible to identify which of these points are scattered and tangled, determining whether they may be said crosscutting concerns.

The process proposed by GT4CCI consists of five steps: Open Coding, Axial Coding, Selective Coding, Graph Analysis 4CCI and Results Table Creation. Figure 1 shows the flowchart of the process proposed by GT4CCI. It is important to note that the first three steps are original of the GT, while the last two steps are defined by the approach GT4CCI. These two steps are used to accommodate the identification and

documentation of crosscutting concerns. Each of these steps are detailed and illustrated in the following subsections.

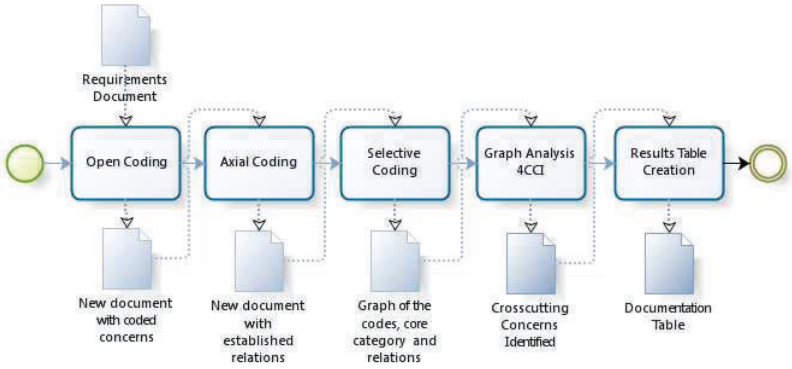


Fig. 1. Flowchart of the GT4CCI Process

3.1 Open Coding

GT4CCI approach is initiated by applying the Open Coding, that has the requirements document as input. In this step, all the relevant data in this document are analyzed, compared and coded. Consequently, all the requirements and other relevant information outlined in this document are analyzed and codes are created for each of these. These codes are created in order to identify and register the data considered relevant in the document analyzed and they are to be treated in the next step.

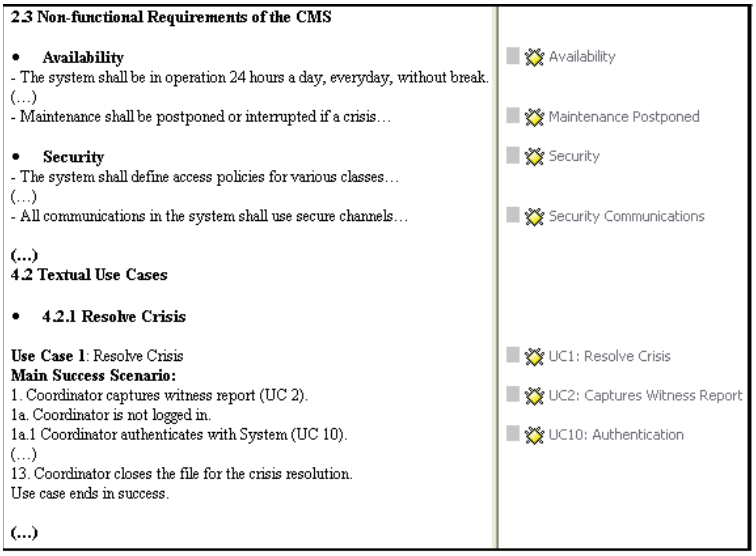


Fig. 2. Open Coding in CMS

In our Toy Example, codes were created to identify each concern specified by CMS document. Figure 2 presents the results generated by open coding in only a small part of this document. This part is related to two non-functional requirements (Availability and Security) and a one textual use case (Resolve Crisis). In the right side of Figure 2 are the codes created for each concern identified in this section. It is possible to see in this figure, for example, the code 'Availability', created for the concern Availability and the code 'UC1: Resolve Crisis', generated for the Use Case 1. Is worth mentioning that all concerns identified in the CMS document were also coded, although not shown in Figure 2.

3.2 Axial Coding

After establishing codes during the Open Coding, presented in the previous subsection, the coded requirements document is submitted to the Axial Coding. In this step are establishes the relations between the codes previously created.

These relations are established through connectors. Each connector identifies the type of relationship between two codes. In GT4CCI two special connectors are used: the connector 'is part of', that indicates that a code is tangled within another and the connector 'is in', that indicates that a code is scattered. The establishment of these relations in the document, however, is done by the user with the support of Atlas.ti tool, once it provides a Codes Manager, which supports the establishment of relationships between these codes. For instance, considering the CMS, the relation between the codes 'UC1: Resolve Crisis' and 'Security' is established. In accordance to information extracted from the requirements document, 'UC1: Resolve Crisis' is related to 'Security' through connector "is part of" which means that UC1 is tangled within non-functional requirement Security.

Is worth noting that, in addition to the relationship between these two codes, relations with other concerns of CMS were also established. These relationships are explained by the graph, generated for each of these codes, in Selective Coding.

3.3 Selective Coding

After Axial Coding, the Selective Coding is applied to the requirements document, which were defined the relationships among codes. At this stage of the process (established by Grounded Theory), the entire coding process by which the document in question has already been submitted is refined. This refinement consists in analyzing the whole document and codes defined and, thereafter, set the core category. The core category is the most relevant code of analysis, from which a graph is generated, showing all relations between this and other codes, established in the preceding steps.

It is also important to note that GT4CCI treats each code individually, in order to facilitate the understanding and visualization of relationships established with them. This means that each code is subjected to Selective Coding in an individual way, so as to be regarded as core category of analysis. Consequently, a graph presenting the results of coding process is generated for each category. Figure 3 illustrate the graphs generated for two categories that we defined for CMS, 'Security' and 'UC1:Resolve

Crisis', respectively. These graphs were automatically generated for Atlas.ti tool, and explained in next section.

The graphs generated in this step are analyzed in detail in the next step of the GT4CCI: Graph Analysis 4CCI.

3.4 Graph Analysis 4CCI

In this step, entirely conceived by GT4CCI approach, the graphs generated to the core categories are carefully analyzed in order to identify crosscutting concerns. This analysis is based on the relationships between the codes presented in graph in order to identify and determine whether the core category may or may not be said a crosscutting concern. For the correct identification, GT4CCI sets some basic guidelines:

- The identification of crosscutting concerns is made by verifying scattering and tangling of the codes defined for a requirements document;
- A concern is considered scattered when its specification is necessarily scattered between many others concerns (whether requirements, use cases, functionalities, etc.) of the same document. This scattering is represented by at least two relations 'is in' between the core category and other codes;
- A concern is considered tangled when its specifications is interleaved with the specifications of others concerns (whether requirements, use cases, functionalities, etc.) in some parts of the same document. This tangling is represented by at least two relations 'is part of" between the core category and other codes;
- Thus, GT4CCI approach considers crosscutting concern, that concern is the origin point of at least two relations 'is in' and the target of at least two relations 'is part of'.

It is essential to highlight that the data analyzed are not restricted, for example, to the description of the requirements and use cases, as in many other approaches. In the case of GT4CCI approach, any data considered relevant within the document can and should be analyzed. The more points in the document are analyzed, the higher the quality and reliability of the conclusions reached at the end of the analysis process.

The left top of Figure 3 presents the graph generated from the step of Selective Coding to the code 'Security'. From the analysis of this graph, it is possible to see that the core category 'Security' is scattered as it is associated with many other concerns of this system through 'is in' relations. Thus, this category can be said tangled, since it is target of many 'is part of' relations. This way, according to the concepts adopted by this approach and considering only a part of CMS requirements document, the non-functional requirement Security is a crosscutting concern, since it can be considered both tangled and scattered.

Continuing with the CMS Toy Example, the right top of Figure 3 shows the graph generated for 'UC1: Resolve Crisis'. In this case, can be seen that 'UC1: Resolve Crisis' is considered tangled, since it is associated with many others concerns of this system through connector 'is part of'. This core category is also considered to be scattered, since it is the origin of many 'is in' relations. Thus, it is possible to affirm that

the Use Case 1, present in the CMS, is a crosscutting concern since it has both the scattering and tangling characteristics.

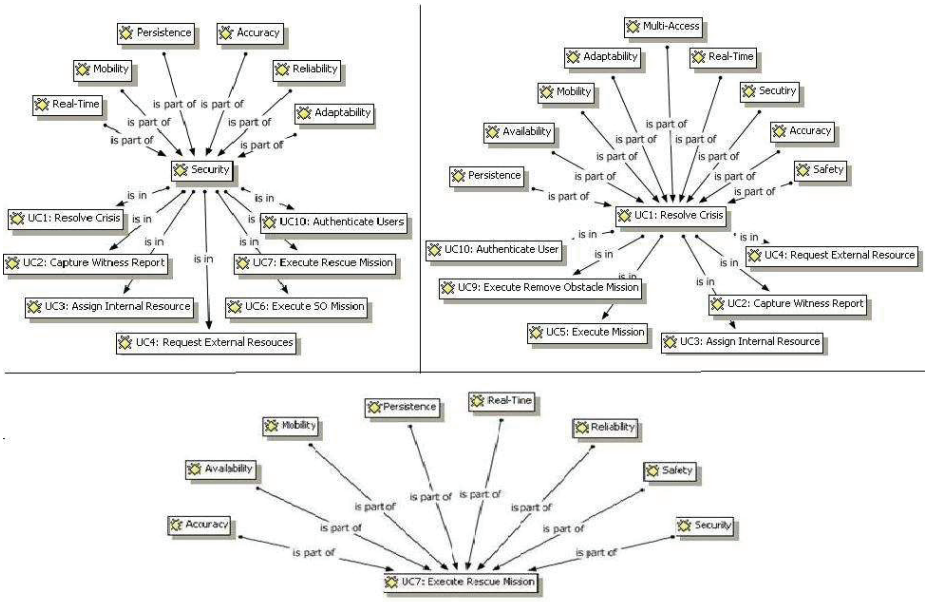


Fig. 3. Graphs of core categories ‘Security’, ‘UC1: Resolve Crisis’ and ‘UC7: Execute Rescue Mission’

Finally, the bottom of Figure 3 shows the graph generated for 'UC7: Execute Rescue Mission’, from the CMS documents. From analysis of this graph, it is possible to see that the core category 'UC7: Execute Rescue Mission' is tangled as it is target of many 'is part of' relations. However, this category cannot be said scattered, since it is not associated with any other concern of this system by any ‘is in’ relation.

This way, according to the concepts adopted by this paper, it is possible to affirm that ‘Use Case 7: Execute Rescue Mission’ is not considered a crosscutting concern, since it cannot be considered scattered.

3.5 Results Table Creation

After identifying, in the previous steps, the final step proposed by GT4CCI is the Creation of a Results Table. The purpose of this Results Table is to document, objectively, all data resulting from application of the approach in a requirements document, enabling a simpler and agile query of these data when necessary in requirements validation and verification or in subsequent stages of software development.

This table contains four columns: 'Concern', Scattered', 'Tangled’ and 'Crosscutting Concern'. Table 1 illustrates a small example of the results table created to CMS. The column 'Concern' contains the core category analyzed, to which the other fields will be directly related. The column ‘Scattered' lists in which other concerns of the system

that core category is scattered. The field 'Tangled' lists all the other concerns that the core category is tangled within. Finally, the Column 'Crosscutting Concerns' indicates, in an objective way, if the core category in question may or may not be considered crosscutting concern. It is worth noting that the names of the columns, consisting of very simple terms, are done so in order to facilitate the identification and understanding during any future queries made to this documentation.

Table 1. Part of the Results Table of Toy Example

Concern	Scattered	Tangled	Crosscutting Concern
Security	UC1: Resolve Crisis UC2: Capture Witness Report UC3: Assign Internal Resource UC4: Request External Resource UC 6: Execute SuperObserver Mission UC7: Execute Rescue Mission UC10: Authenticate Users	Adaptability Accuracy Mobility Persistence Real-Time Reliability	**Yes**
UC1: Resolve Crisis	UC2: Capture Witness Report UC3: Assign Internal Resource UC4: Request External Resource UC5: Execute Mission UC 9: Execute Remove UC10: Authenticate Users	Accuracy Adaptability Availability Mobility Multi-Access Persistence Real-Time Safety Security	**Yes**
UC7: Execute Rescue Mission	-	Accuracy Availability- Mobility Persistence Reliability Safety Security	**No**

Table 1 shows only a part of the Results Table generated for CMS requirements document, illustrating part of the final results, since the CMS document is very complete and extensive. In this table, it is possible to see three concerns. One of these concerns is a non-functional requirement and two functional requirements. The first, Security, is considered scattered and tangled, and for this reason it is considered a crosscutting concern. The second and third concerns are use cases, representing functional requirements, called Resolve Crisis and Execute Rescue Mission, respectively. UC1: Resolve Crisis is tangled within and scattered among the system, and for this reason this concern is considered a crosscutting concern, while UC7: Execute Rescue Mission is considered tangled, but not scattered, and therefore is not considered a crosscutting concern. Once again it is emphasized that this is just an excerpt of Re-

sults Table generated for the CMS. With the use of the entire table is possible to make a quick identification of all the concerns of this system, as well as analyzing in which part of the requirements document these can be said scattered and tangled, and especially check whether a concern can or cannot be considered crosscutting.

During the application of the GT4CCI approach in the requirements document CMS, some points might be highlighted. 21 concerns were analyzed in the CMS document, thus defined as core categories of analysis. 11 of these concerns are related to non-functional requirements, while 10 are related to use cases of the system. During the analysis of the CMS document were defined more than 150 codes and over 200 relationships were established.

With this, it was possible to analyze in detail each concern identified in CMS. By applying this approach in the document CMS was possible to identify the scattering and tangling of these concerns in the system and determine whether it may or may not be considered crosscutting concerns. With these results, it is possible to understand which concerns of this system need to be better modularized in order to add quality to it. Furthermore, by the results obtained through the use of GT4CCI is possible to call attention to possible errors in the development of the requirements document analyzed, such as poor specification of a requirement, for example.

Thus, beyond simply identifying if a concern can or cannot be said or crosscutting, GT4CCI approach also aims to better view the requirements document, highlighting potential problems and issues that should be better understood and analyzed.

4 Experimental Study

In order to evaluate GT4CCI, the approach proposed by this paper, an experimental study was designed and implemented. The main objective of this study is to evaluate the correctness of the results presented from the use of the approach GT4CCI. Details of the context and execution of this study and the analysis of their results are presented below.

4.1 Context and Implementation of the Experimental Study

Nine postgraduate students in Systems and Computing at the Federal University of Rio Grande do Norte voluntarily participated in this experimental study. These participants were divided into two groups in order to evaluate and use the approach in two distinct scenarios, since each of the groups performed this experimental study using different requirement documents. Group I performed this experimental study using the Methodology Explorer [15] system requirements document as a basis for the analysis, while Group II used the Meeting Scheduler [16] system requirements document.

The sectioning of participants in two groups was made according to their previously declared level of knowledge and experience in Requirements Engineering and in Identification of Crosscutting Concerns, before the execution of the experimental study, by each one of the participants. The distribution of these participants into the

groups was made attempting to establish equivalent groups in terms of the previously declared knowledge held by them.

Before the experimental study's implementation, a training with all the participants was conducted, aiming at exposing some of the basic knowledge needed to this study, such as the definition of crosscutting concerns, the explanation of the process proposed by Grounded Theory methodology and the exhibition of examples, both theoretical and practicals, of the identification of these in a requirements document. This training strived to reduce the difference in the level of knowledge presented by the participants. Also in this training, the GT4CCI approach and the Atlas.ti were introduced. Each step of this approach was minutely explained and demonstrated, through a simple example, using the Atlas.ti tool. During and immediately after this training, the participants had the opportunity to clarify their doubts regarding the use of both the approach and the tool.

In order to perform this experimental study, a sequence of activities was performed by the participants in a full and sequential fashion, thus ensuring more safety to the resulting data. Firstly, all participants were required to access the Atlas.ti tool and the requirements document to be analyzed. Then each one of the participants individually made a preliminary reading of the requirements document. Shortly thereafter, the participants, together, applied the Open Coding phase in the requirements document they were analyzing. The application of this group phase is made with the objective of assuring that, in future phases, the participants will work with the same concerns defined in this first stage. The individual application of the Open Coding could lead to distinct definition and identification of interests by each participant, which would produce a less coherent result and a less accurate analysis. Finally, the concerns identified in the previous phase were equally divided among the participants and each one of them applied, individually, the next phases defined by the GT4CCI approach, aiming to ascertain whether each of the identified interest could be defined as crosscutting.

In the end of the experimental study implementation, all the resulting data of the GT4CCI approach application were collected to be analyzed. The analysis of these data will be presented in the following section.

4.2 Analysis of Results

Two metrics, recall and precision, were utilized in order to evaluate the correctness of the obtained results through the use of GT4CCI approach. Both of these metrics are hereby utilized for displaying efficiency in relation to the correctness of data, as well as for being extensively employed in other experimental works of similar nature.

According to [17], Recall can be defined as a metric utilized to evaluate the plenitude of the obtained data. Applying this metric into the context of the study, we can state that Recall evaluates the rates of plenitude presented by the identification of concerns done through the use of GT4CCI approach. In other words, the amount of actual concerns existent within a certain requirements document that this approach was able to identify. The metric of Recall is calculated through the following formula:

$$Recall = \frac{Elements\ Correctly\ Identified}{Correct\ Elements}$$

Still in accordance with [17], Precision can be defined as a metric utilized to evaluate precision, or fidelity of obtained data. Therefore, applying this metric into the context of this study, we can state that the Precision metric evaluates the rates of fidelity presented by the identification of concerns made through the use of GT4CCI approach, that is, how many concerns identified by the approach corresponds exactly those concerns actually in the document. The Precision Metric is calculated through the following formula:

$$Precision = \frac{Elements\ Correctly\ Identified}{Identified\ Elements}$$

It is worth noting that the definition of the correct elements used as the basis of this experimental study were defined by the author of this work in conjunction with an expert in crosscutting concerns identification. The analysis of the collected data through the application of the experimental study enabled us to grasp some relevant information. First, it is possible to observe that Group I identified 18 concerns in the requirements document Methodology Explorer. 13 of these were functional concerns, while the other 5 were non-functional concerns. The identification of functional concerns through the GT4CCI approach in the Methodology Explorer requirements document, utilized by Group I, resulted in a rate of Precision and Recall of 100%. This demonstrates that all the existing functional concerns within this document were correctly identified by GT4CCI approach, without neglecting any one of them. The same occurred with the identification of both non-functional concerns and non-functional crosscutting concerns. On the other hand, it is possible to find a Recall rate of 100% amongst the identification of functional crosscutting concerns, although the encountered Precision rate is 75%. This also occurred with the case of identification of crosscutting concerns (both functional and non-functional). This identification obtained a Recall rate of 100% as well. Albeit the Precision rate encountered was 89%.

Employing the same analysis regarding the application of GT4CCI approach in the Meeting Scheduler requirements document, utilized as an analytical artifact by Group II, it is possible to highlight certain points. Group II identified 21 concerns in the requirements document Meeting Scheduler. 13 of these were functional concerns, while the other 8 were non-functional concerns. One may observe, then, that the identification of functional concerns, the identification of non-functional concerns and the identification of non-functional crosscutting concerns present a Precision rate of 100% along with a Recall rate of 87,5%. Thus implying that all of the existing concerns in these types of documents were correctly identified through the use GT4CCI approach. Having said that, one may conclude through the Recall rate of 87,5%, that the GT4CCI approach neglected a few interests of such types existent in the requirements

document. Furthermore, the case of identification of functional crosscutting concerns made by Group II in this experimental study, permits one to observe that the Precision and Recall rates were 78%. In the case of crosscutting concerns (both functional and non-functional) we encountered a Precision rate of 87,5% and a Recall rate of 82%.

In face of these results, it is possible to conclude that the GT4CCI approach provides results with good correctness, considering that in both analyzed scenarios the rates of Recall and Precision never appeared inferior to 75% in any one of the identification of interests. These numbers reflect every type studied, having presented in several situations Recall and Precision rates of 100%, demonstrating identifications without error or negligence. Further information on this experimental study can be seen in [18].

5 RELATED WORKS

With the perception of the benefits that come from identification and documentation of crosscutting concerns in the earliest phases of software lifecycle, some approaches that systematize this identification were developed. Some of these are applied to a specific type of requirements model, such as i* [13], AOV-graph [12], BPMN [14] and Use Cases [4, 5, 6], while other analyze textual requirements documents. Among the principal of these approaches, four are highlighted by this paper: Theme/Doc [3], DISCERN [2], Early-AIM [7] and CCCINPL [8].

Each of these approaches has characteristics that are individual to them. These characteristics may sometimes represent limitations or gains of an approach in relation to another. Table 2 is a comparative table that allows easy viewing the main differences and similarities presented by each of these approaches and GT4CCI.

Table 2. Approaches Comparison Table

Approach	Identification Technique	Artifacts Analyzed	Requirements Analyzed	Automation and Tools
GT4CCI	Contextual Analysis	Any Req. Document	Analysis of All Relevant Data	Semi-automatic. Atlas.ti Tool
Theme/Doc	Contextual Analysis	Structured Req. Document	Any Textual Req.	Semi-automatic. Theme/Doc Tool
DISCERN	Contextual Analysis	Any Req. Document	Non-functional Textual Req.	Without automation. Does not cite any tool.
Early-AIM	Natural Language Processing	Any Req. Document	Any Textual Req.	Semi-automatic. EA-MINER Tool
CCCINPL	Natural Language Processing and Contextual Analysis	Any Req. Document	Any Textual Req.	Automatic. C3I Tool

In Table 2, four comparison criteria were established. These criteria are: technique used to identify crosscutting concerns, artifacts analyzed to proceed the identification, kind of requirements analyzed by the approach and the level of automation of the approach and what tools it uses to support its process. From this table, we can observe some interesting points. The first of these points, referring to the identification technique, shows that most approaches perform the contextual analysis of the requirements document. However, Early-AIM approach does not make this kind of analysis, disregarding the context in which the concern analyzed is embedded.

Another point worth mentioning is the artifacts analyzed. Theme/Doc approach, for example, only makes the analysis of requirements documents previously structured by a predefined pattern by this approach. This way, makes its use impractical in documents developed without this pre-defined structure.

The requirements analyzed by the approach are also a relevant point of this comparison. It can be noticed that one of the approaches analyze just one kind of requirement, the non-functional requirements, limiting the analysis. Moreover, a fact that deserves mention is the possibility of analysis of any relevant data, provided by GT4CCI. This means that this approach not only analyzes textual requirements, but also enables the analysis of the use-case diagram, for example.

Finally, another point worth mentioning is the level of automation of these approaches. DISCERN approach have no automation and do not have tool that supports the any activity of its process. This makes the application of the approach generally slower and less reliable. In contrast there is also a fully automatic approach, the CCCINPL approach. The total automation provides certainly gains in celerity. However, the analyst is unable to follow the process of identifying crosscutting concerns and is unable to see possible flaws in the requirements document and in the analysis, since the approach returns only the final result of analysis.

6 CONCLUSIONS

This paper briefly introduced GT4CCI approach. This approach aims to organize and assist the identification of crosscutting concerns in the initial stages of the software development process using the requirements document as artifact for analysis.

Using the approach GT4CCI is possible to identify the crosscutting concerns of a system through a qualitative analysis of data, based on the context in which the concerns are embedded into the requirements document. Thus, it is possible to identify crosscutting concerns in any requirements document by analyzing all data deemed relevant in this document.

Moreover, the results obtained by applying GT4CCI are strongly based on data present in the document analyzed, and thus can be traced and more easily justified. Therefore, through the benefits arising from the use of GT4CCI approach, it is expected that it provides a better view of the requirements document, highlighting potential problems and issues that should be better understood and analyzed. Moreover, from the results found from the execution of the experimental study reported in this work, we can conclude that GT4CCI approach presents a good correctness of the results, once showed high values of recall and precision metrics in both scenarios

evaluated. This ensures the quality and reliability found from the use of GT4CCI approach in identifying crosscutting concerns in requirements documents.

As future work, we suggest performing experimental studies to measure GT4CCI performance compared to others existing approaches. From the results of these experiments, it is expected to make possible improvements in this approach and expand the use of GT4CCI for other kinds of artifacts developed during the software lifecycle. With this, it will be possible to track and analyze whether these crosscutting concerns are propagated in several activities of software development or to analyze the nature of crosscutting concerns in each of these activities.

References

1. Ali, B.S., Kasirun, Z.M. A Review on Approaches for Identifying Cross-cutting Concerns. In Advanced Computer Theory and Engineering, ICACTE '08. 2008.

2. Rosenhainer, L. The DISCERN Method: Dealing Separately with Crosscutting Concerns. In Proceedings of OOPSLA Early Aspects 2005. San Diego, USA, 2005.

3. Baniassad, E., Clarke, S. Finding Aspects in Requirements with Theme/Doc. In Proceedings of Early Aspects 2004 (AOSD 2004). Lancaster, United Kingdom, 2004.

4. Araújo, J., Moreira, A., Brito, I., Rashid, A. Aspect-Oriented Requirements with UML. In Workshop on Aspect-Oriented Modelling with UML (UML 2002). Dresden, Germany, 2002.

5. Jacobson, I., Ng, P. Aspect-Oriented Software Development with Use Cases. Addison-Wesley. 2005.

6. Brito, I. Aspects Oriented Requirements Engineering. In 7th International Conference on Unified Modelling Language (UML 2004). Lisbon, Portugal, 2004.

7. Sampaio, A., Rashid, A., Rayson, P. Early-AIM: An Approach for Identifying Aspects in Requirements. Poster Paper. In Requirements Engineering Conference. Paris, France., 2005.

8. Ali, B. S., Kasirum, Z. M. Crosscutting concern identification at requirement level. In Malaysian Journal of Computer Science, vol. 21(2), pp.78-87, 2008.

9. Glaser, B.; Strauss, A. The discovery of Grounded Theory: Strategies for Qualitative Research. New York: Aldine Transaction, 1967.

10. Atlas.ti. Version 6.2. Available: <http://www.atlasti.com>. Accessed in: 28 oct. 2011.

11. Kienzle, J., Guelfi, N., Mustafiz, S. Crisis Management Systems: A case study for aspect-oriented modeling - Technical Report. McGill Univ. 2009.

12. Yu, Y., d. P. Leite, J.C.S., Mylopoulos, J. From goals to aspects: Discovering aspects from requirements goal models. In *Proceedings of the 12th IEEE International Requirements Engineering Conference*. 2004.

13. Alencar, F., Castro, J., Moreira, A., Araujo, J., Silva, C., Ramos, R., Mylopoulos, J. Intgration of Aspects with i* Models. In *Agent-Oriented Information Systems IV*. 2008.

14. Cappelli, C., Leite, J.C.S.P., Batista, T., Silva, L. An aspect oriented approach to business process modeling. In *Proceedings of the 15th workshop on Early aspects, EA '09*. ACM. New York, NY, USA, 2009.

15. Silva Júnior, C.R., Perrelli, H., Mesquita, S. Documento de Requisitos do Sistema Methodology Explorer. Available: http://www.cin.ufpe.br/~mexplorer, 2001.

16. van Lamsweerde, K., Darimont, R., Massonet, P. The Meeting Scheduler System – Preliminary Definition. Internal Report. Université Catholique de Louvain, 1993.

17. Anacleto, A.C.S. Aplicação de Técnicas de Data Mining em Extracção de Elementos de Documentos Comerciais. Tese de Mestrado. Universidade do Porto. Porto, Portugal. 2009.

18. Sobral, L.A. GT4CCI: Uma abordagem para identificação de Interesses Transversais em Documentos de Requisitos. Dissertação de Mestrado. UFRN. Natal, Brasil. 2013.

A Semi-Automatic Strategy to Identify Crosscutting Concerns in PL-AOVgraph Requirement Models

Maíra Medeiros[1], Lyrene Silva[1], Ana Luisa Medeiros[1]

[1] DIMAp – Departamento de Informática e Matemática Aplicada
UFRN – Universidade Federal do Rio Grande do Norte
Campus Universitário Lagoa Nova, Natal, RN, Brasil.

{mairafbmedeiros, analuisafdm}@gmail.com,
lyrene@dimap.ufrn.br

Abstract. The Requirements Engineering area faces problems because the requirements are often ambiguous, incomplete or confusing. These points are commonly obscured by the natural language, which abstracts the complexity of interactions among requirements. However, these interactions need to be analyzed because they influence on how the software development life-cycle activities can be modularized from a well formulated and concise requirements description. In this context, PL-AOVgraph is an aspect-oriented requirement modeling language, which offers support to represent relationships among concerns and provides separation of crosscutting concerns. However, in order to identify these crosscutting concerns in PL-AOVgraph, there are only some heuristics, which help analysts to perform this activity manually. Therefore, this paper proposes a semi-automatic strategy to identify crosscutting concerns in PL-AOVgraph models. This strategy is based on analysis of an adjacency matrix composed of the relationships among requirements. In order to evaluate this strategy, a case study is applied.

Keywords: Crosscutting concerns, crosscutting concern identification, PL-AOVgraph, ReqSys-MDD tool.

1 Introduction

The requirements engineering activity is responsible for discovering, documenting and maintaining software requirements [17]. Some of the artifacts produced during these activities are written in natural language. Hence, the requirements engineering faces problems due to the fact that the requirements are often ambiguous, incomplete and confusing. These problems happen because there are many ways to describe the same requirement using natural language, and also, it abstracts the complexity of interactions among requirements.

One way to deal with these problems is by using the separation of concerns. It decomposes software systems into smaller modular units, each one is related to one concern. In this context, the Aspect-Oriented Software Development (AOSD) [7] is particularly interesting because the modularization is achieved by encapsulation of

crosscutting concerns through specific abstractions of language used. Crosscutting concerns are parts of a system which are strongly related, scattered or tangled, influencing or restringing each other, making the system complex and difficult to analyze.

The aspect-oriented requirements engineering area proposes methods and techniques to identify, to separate and to compose crosscutting concerns focusing on requirement artifacts.

In the context of the aspect-oriented requirements engineering, there are some approaches of identification, modeling and analysis of crosscutting concerns. Among these, PL-AOVgraph (Product Line – Aspect-Oriented Vgraph) is a goal-oriented requirement modeling language which supports the description, composition and visualization of requirements. Regarding PL-AOVgraph models, the crosscutting concern identification is performed in an ad hoc way, based on some general heuristics, such as quantity of input or output relationships (fan-in or fan-out), or the possibility of reusing a particular concern [16], leaving to the analyst the subjective decision to model some requirements as crosscutting concern or not. Additionally, even deciding to model a particular concern as crosscutting, the analyst may do it incompletely, not modularizing such concern in the most appropriate way.

Therefore, the objective of this paper is to present a semi-automatic strategy to identify crosscutting concerns in PL-AOVgraph models and to report a case study which evaluates if this strategy is efficient. This strategy was inserted in ReqSys-MDD tool, which is an Eclipse plug-in to validate requirement specifications (in PL-AOVgraph) and execute automatic bi-direction transformations between PL-AOVgraph models and Features Models [14][15]. The motivation for attaching our strategy to ReqSys-MDD is the reuse of some ReqSys-MDD functionalities, such as the validation of PL-AOVgraph documents and the parsers Xtext and Acceleo, which transform text to model and model to text, respectively. Furthermore, our aim is to integrate into ReqSys-MDD tool all functionalities about PL-AOVgraph.

This paper is organized as follows. Section 2 presents PL-AOVgraph and its features; Section 3 defines our semi-automatic strategy to identify crosscutting concerns and to write the crosscutting relationships. Section 4 specifies how this strategy was inserted in PL-AOVgraph tool. Section 5 describes the results achieved using our approach with Crisis Management System (CMS) case study [9]. Section 6 summarizes some related works. Section 7 presents our final remarks and future works.

2 PL-AOVgraph

PL-AOVgraph [14] is an extension to the AOV-Graph goals model [16] with support to the variability. In other words, PL-AOVgraph is a requirement modeling language which inherits all the AOV-Graph features. These models consist of oriented graphs composed by the following element types: (i) task – functional requirement; (ii) softgoal – non-functional requirement; and, (iii) goal – organizational goal. The relationships among the PL-AOVgraph elements may be one of three types: contribution; correlation; and, crosscutting [12]. Although these models are graphics, we have worked in their textual representation, which is more easily manipulated.

Correlation relationships indicate the influence, whether positive or negative, from a goal to a softgoal (one-to-one relation). This influence is labeled as follows: Make, Break, Help, Hurt or Unknown.

Contribution relationships are represented by the hierarchical relationships (one-to-one) between child and parent elements, respectively, source and target of the relationship. These contributions may be labeled as one of the following types: And, Or, Xor, or Inc-or.

In opposition to contributions and correlations, which are one-to-one relationships, crosscutting relationships represent many-to-one relations. These relationships can modularize many interactions in one relationship, decreasing the quantity of contributions. The amount of contributions must be specified and also it must be determined which concerns are crosscutting each other [14]. The description of the crosscutting relationship is based on elements of Aspect-Oriented Software Development (AOSD) proposed by AspectJ language, so, it is composed of: (i) Source – which is the origin of the relationship, i.e., what concern influences other requirements; (ii) Pointcut – which is the target set of the relationship, i.e., the requirements which are affected by the source; (iii) Advice – which specifies what requirements (child of source) are scattered or tangled on the pointcuts; (iv) Intertype Declaration – which defines new instances (if it is element type) or types of elements (if it is attribute type) to the model.

Additionally, PL-AOVgraph allows the insertion of new properties to a model by using "property" elements. There are six pre-defined properties to support the variability and the transformation between feature models and PL-AOVgraph [14]. However, these properties are not used in this work.

We use the Crisis Management System (CMS) [9] like a demonstrative example in next sections. This system was defined with purpose to create a common case study for aspect-oriented modeling community and it was presented like special issue in the Transactions on Aspect-Oriented Software Development journal. The CMS domain helps in the identification, evaluation to handle a crisis situation, allowing the communication among all parties involved.

Figures 1 and 2 illustrate a small part of the CMS PL-AOVgraph model. Figure 1 illustrates its graphical representation and Figure 2 is its textual representation. In this example, the task "Manage [Communication]" (line 5) contributes to two tasks (lines 5 and 10) and four softgoals (lines 15, 20, 25 and 30). These contributions are showed by the "and" pointers between the nodes shown in Figure 1, and included in the labels in parenthesis in Figure 2.

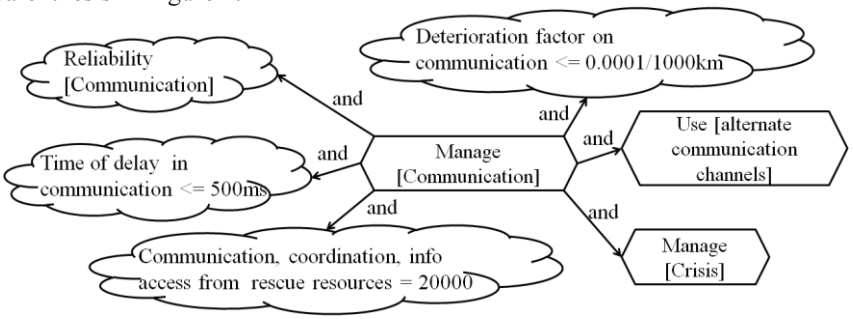


Fig. 1. Contribution relationships in PL-AOVgraph graphic mode.

```
01 aspect_oriented_model {
02   goal_model "Crisis Management" {
03     goal "Crisis resolved" (and) {
04       task "Manage [Crisis]" (and) {...
05         task "Manage [Communication]" (and) {}
06       }
07     }
08     softgoal "Security" (and) {...
09       task "Use [alternate communication channels]" (and) {
10         task_ref "Manage [Communication]" (and) {}
11       }
12     }
13     softgoal "Reliability" (and) {...
14       softgoal "Reliability [Communication]" (and) {
15         task_ref "Manage [Communication]" (and) {}
16       }
17     }
18     softgoal "Multi-Access" (and) {...
19       softgoal "Communication, coordination, info access from rescue resources = 20000" (and) {
20         task_ref "Manage [Communication]" (and) {}
21       }
22     }
23     softgoal "Real-time" (and) {...
24       softgoal "Time of delay in communication <= 500ms" (and) {
25         task_ref "Manage [Communication]" (and) {}
26       }
27     }
28     softgoal "Accuracy" (and) {...
29       softgoal "Deterioration factor on communication <= 0.0001/1000km" (and) {
30         task_ref "Manage [Communication]" (and) {}
31       }
32     }
33   }
34 }
```

Fig. 2. Contribution relationships in PL-AOVgraph textual mode.

As requirement models are usually extensive, insofar as the model is evolving and growing it becomes difficult to maintain its readability as well as its understanding. Therefore, the crosscutting relationships are a strategy to separate concerns, reduce the number of contribution relationships and tangled and scattered elements. Figure 3 shows a crosscutting relationship example, which replaces those relationships represented by task references (task_ref) showed in Figure 2(lines 10, 15, 20, 25 and 30).

Figure 3, line 2, represents the source of this crosscutting relationship, it is the goal "Crisis resolved", because it is the parent of the elements which are scattered or tangled. The pointcut block (lines 03 to 08) defines the elements which are affected by the advice: the two tasks, "Use [alternate communication channels]" and "Manage [Crisis]"; and four softgoals "Reliability [Communication]", "Communication, coordination, info Access from rescue resources = 20000", "Time of delay in communication <= 500ms" and "Deterioration factor on communication <= 0.0001/1000km". The advice block (lines 09 a 11) defines the elements which are scattered or tangled with pointcut elements, referenced by "PC1", in this case, the task "Manage [Communication]" (line 10).

```
01 crosscutting {
02   source: goal_ref "Crisis resolved" {
03   pointcut PC1: include "Use [alternate communication channels]"
04                 and include "Manage [Crisis]"
05                 and include "Reliability [Communication]"
06                 and include "Communication, coordination, info access from rescue resources = 20000"
07                 and include "Time of delay in communication <= 500ms"
08                 and include "Deterioration factor on communication <= 0.0001/1000km"
09   advice (around): PC1 {
10     task_ref "Manage [Communication]" (and) {}
11   }
12 }
```

Fig. 3. Crosscutting relationship in PL-AOVgraph textual model.

3 Crosscutting Concerns Identification in PL-AOVgraph

The identification strategy presented in this paper is based on the fan-out analysis of relationships among requirements. For this analysis, it is used an adjacency matrix to identify and to account these output relationships. In this context, the identification process is comprised of three major steps, as illustrated in Figure 4. A PL-AOVgraph requirements model is the input of this process. From it, a matrix with the relationships among requirements is created. After that, crosscutting concerns may be identified by accounting how many contribution relationships are sourced at each requirement. Hence, crosscutting concerns may be specified, considering the data of the matrix and the data of the input model. Finally, the PL-AOVgraph model is updated by substituting contributions by crosscutting relationships. This process is detailed and exemplified below.

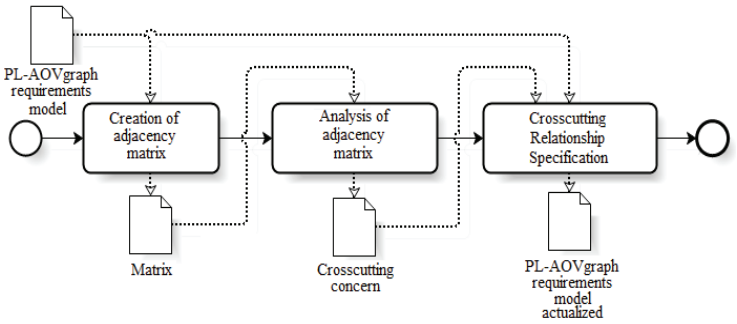


Fig. 4. Process to identify and specify crosscutting relationships.

The first stage is the creation of an adjacency matrix, considering that it shows the relationships among requirements. This activity takes as input a PL-AOVgraph requirement model from that it generates an adjacency matrix. This matrix is built from contribution relationships among the PL-AOVgraph elements. As soon as the PL-AOVgraph model is read by the tool, the matrix is fulfilled in a way that the horizontal lecture indicates the relationships which the element, refered by the matrix lines, originates.

In order to illustrate this activity, Figure 5 shows a small part of the requirement specification in PL-AOVgraph of CMS. In this Figure, there are two major concerns, represented by the goal "Crisis resolved" (line 03) and by the softgoal "Security" (line 24), as well as contribution relationships between them and other (sub) tasks (lines 04 to 22 and lines 25 to 31), "Authenticate [User]" is a task that contributes, besides to the softgoal "Security" (line 24), to the tasks "Manage [External Resource]" and "Manage [Internal Resource]" and it is represented by task_refs elements (lines 09 and 19).

Based on this PL-AOVgraph model, the adjacency matrix is fulfilled, see Table 1, where its horizontal lecture indicates the direction of the contribution relationships, for instance, "Select [Employee]", "Receive confirmation of acceptance mission" and "Inform [Mission info]" (lines 13 to 15 in Figure 5) contribute to "Assign [Internal Resource]" (line 12 in Figure 5).

```
01 aspect_oriented_model {
02   goal_model "Crisis Management" {
03     goal "Crisis resolved" (and) {
04       task "Manage [Crisis]" (and) {
05         …
06         task "Manage [Resource] " (and) {
07           task "Manage [External Resource] " (and) {
08             task "Request mission to ERP" (and) {}
09             task_ref "Authenticate [User]" (and) {}
10           }
11           task "Manage [Internal Resource] " (and) {
12             task "Assign [Internal Resource]" (and) {
13               task "Select [Employee]" (and) {}
14               task "Receive confirmation of acceptance mission" (and) {}
15               task "Inform [Mission info]" (and) {
16                 task "Request the employee to login" (and) {}
17               }
18               task "Update availability of employee" (and) {}
19               task_ref "Authenticate [User]" (and) {}
20             }
21           }
22       }
23     }
24     softgoal "Security" (and) {
25       task "Authenticate [User]" (and) {}
26         task "Request [Login info]" (and) {}
27         task "Validate [Login info]" (and) {}
28         task "Block [Access] after 3 consecutive failed attempts" (and) {}
29       }
30       task "Use [alternate communication channels]" (and) {…}
31     }
32   }
33 }
```

Task references

Fig. 5. Example of PL-AOVgraph textual model

Once the relationships between the requirements are identified and added to the matrix, the second step of the process takes place, which consists on analyzing this matrix to identify crosscutting concerns.

This analysis consists on evaluating the scattered and tangled phenomenon. This phenomenon is identified when a requirement affects (or is affected by) several other requirements. Therefore, how much more a requirement contributes to other, more scattered and tangled it is. Hence, the crosscutting concern identification can be

achieved by counting these relationships. For example, using our adjacency matrix (Table 1), we can quickly visualize that "Authenticate [User]" contributes to three other requirements while the others only contributes to one (highlighted in Table 1). It is important to mention that some cells, in which the element did not generate output relationships, were removed to improve the visualization.

It is important to remark that we do not determine a minimum value to the amount of contributions that a requirement must have to be considered a crosscutting concern. We prefer to make this value configurable, once this quantity depends on the context and it varies from one system to another, so, the requirements engineer must set it. Because of this, our strategy is not completely automatic. For example, considering the example of Table 1, if the requirements engineer set this value with three (it can be any number greater than two – the minimum defined by literature), then it can be inferred that the task "Authenticate [User]" is a crosscutting concern, whereas if the requirements engineer set this value with four (or more) then it would not be considered a crosscutting concern in this model, because there is not any element which generates this quantity of relationships.

Table 1. Example of adjacency matrix

⟶	Crise resolved	Manage [Crisis]	Manage [Resource]	Manage [External Resource]	Request mission to ERS	Manage [Internal Resource]	Assign [Internal Resource]	Inform [Mission info]	Security	Authenticate [User]
Manage [Crisis]	x									
Manage [Resource]		x								
Manage [External Resource]			x							
Request mission to ERS				x						
Manage [Internal Resource]			x							
Assign [Internal Resource]					x					
Select [Employee]							x			
Receive confirmation of acceptance mission							x			
Inform [Mission info]							x			
Request the employee to login								x		
Update availability of employee					x					
Authenticate [User]				x		x			x	
Request [Login info]										x
Validate [Login info]										x
Block [Access] after 3 consecutives failed attempts										x
Use [alternate communication channels										x

Once the crosscutting concerns are identified, it is necessary to represent them in crosscutting relationships, it is the third and last step of the process. Therefore, to specify crosscutting relationships, it is indispensable to define the elements that form this kind of relationship: source, pointcut, advice and/or intertype declaration.

It is important to emphasize that if there are crosscutting relationships defined in the input specification, then they need to be updated with other pointcuts and advices.

Furthermore, there must never be relationships with the same source otherwise, there will not modularization.

The source is the crosscutting relationship origin that is represented by the parent requirement of the crosscutting concern. The Table 1 illustrates that the task "Authenticate [User]" was identified as crosscutting concern by analysis of matrix. Therefore, to define the source of the relationship, it is necessary to analyze the specification (Figure 5) to identify its parent requirement. Analyzing this specification, we can notice that the softgoal "Security" is the parent requirement, consequently, it is the source of the crosscutting relationship (line 25).

Pointcuts indicate the requirements which are affected by crosscutting concern, then in our example, the tasks "Manage [External Resource]" and "Manage [Internal Resource]" compose the elements of the pointcut. As explained in Section 2, advice and intertype declaration indicate requirements, which affect other requirements. However, the pieces of advice are stated in the model while intertype declarations are not stated in the model. So, this strategy cannot identity intertype declarations. Therefore, in our example, the advice is composed of the task "Authenticate [User]".

It is worth highlighting that since each crosscutting relationship accommodates many contributions, then these contributions are replaced by crosscutting relationships. This reduction and modularization of the relationships aids traceability and consistency management, because each part involves only one feature, thus it makes it easier to locate changes and to deal with one important issue at a time.

Finally, concluding these stages, a new requirement specification is created which removes some contributions and adds crosscutting relationships. The Figure 6 shows the crosscutting relationship (lines 01 a 08) created to represent that "Authenticate [User]" is a crosscutting concern and it crosses to "Manage [External Resource]" and "Manage [Internal Resource]".

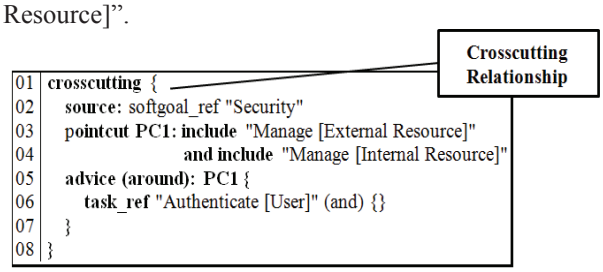


Fig. 6. Example of Crosscutting Relationship

4 Extending ReqSys-MDD Tool to Identify Crosscutting Concerns

ReqSys-MDD [15] implements a bi-directional mapping between PL-AOVgraph and Features Model, using MDD approach. This tool was coded at Eclipse environment, which offers the Plug-in Developer Environment (PDE) and the Eclipse modeling framework (EMF), both needed for the elaboration of metamodels, and the ATL (ATLAS Transformation Language) Development Tool, needed for implementation of mapping rules.

Furthermore, Xtext and Acceleo plug-ins are utilized and integrated to the EMF in order to transform text into model and model into text, respectively. In ReqSys-MDD, Xtext is responsible for transforming a PL-AOVgraph textual specification (or a Feature Model described in XML), in a XMI model, which is the input to ATL transformation rules. The Acceleo, on the other hand, is responsible for doing the inverse process, that is to transform a XMI model, produced by ATL transformation, in a PL-AOVgraph textual specification (or in a Feature Model) described in XML.

ReqSys-MDD also helps to edit PL-AOVgraph specifications, offering keyword coloring and lexical and syntactic analysis, certifying that input models are conformed to their metamodel.

In this context, the method proposed in this paper was implemented as an additional module to the ReqSys-MDD plug-in. This module was coded using the Java programming language, this choice occurred because ATL did not present support to strategy implementation proposed in this paper.

Figure 7 shows the crosscutting concern identification flow in an additional module of ReqSys-MDD tool: (i) the input PL-AOVgraph textual specification is analyzed by the Xtext module and thus transformed in a XMI model; (ii) from this XMI model, Java objects are created and through analysis of adjacency matrix the crosscutting relationships are identified and written, generating a new XMI model; (iii) this XMI is transformed, through Acceleo, in a PL-AOVgraph textual specification.

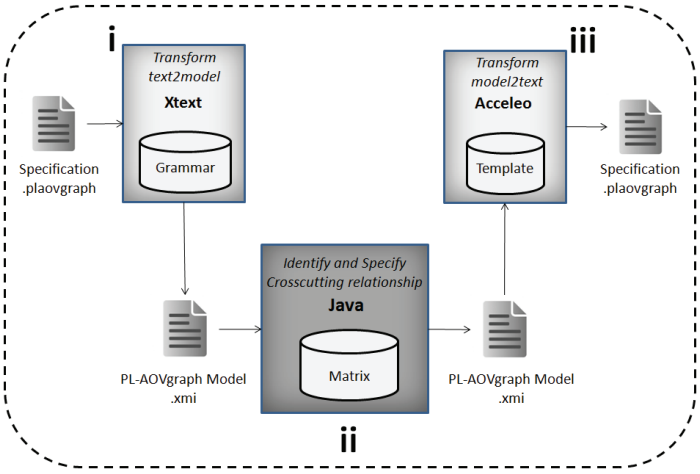


Fig. 7. The crosscutting concern identification flow in ReqSys-MDD.

5 Case Study

We have used in this case study the same system used like a demonstrative example in this paper – the Crisis Management System (CMS). This system was selected because it has many of the PL-AOVgraph elements which are fundamental for an accurate analysis.

As mentioned in this paper, the CMS system aims to help in identifying, assessing, and handling a crisis situation by orchestrating the communication between all parties

involved, by allocating and managing resources, and by providing access to relevant crisis-related information for authorized users [9].

The case study reported here aimed to compare the crosscutting concerns obtained from the manual technique and the results from the strategy described in this paper. It was consisted of 3 stages:

1. **Manual crosscutting concerns identification and specification** – this stage is responsible to model a PL-AOVgraph model before and after the composition process, based on [9]. It is important to remind that composition process generates a specification whose crosscutting relationships are disunited in contribution relationships. So, firstly, it is modeled a specification using heuristics to identify crosscutting relationships (model 1), defined in [16]. After that, a version without crosscutting relationships of this model was generated (model 2) (by the composition process explained in [16]) in order to be used as input to the ReqSys-MDD. It is important to remark that these models were not created by the authors of this strategy. However these models were created by a person that was expert in PL-AOVgraph because of this we assumed that these models are correct;
2. **Crosscutting concerns identification using ReqSys-MDD** – the second model cited on stage 1 was used as input in ReqSys-MDD in order to identify crosscutting concerns. It was set that a requirement is scattered or tangled if it affects 3 or more elements. ReqSys-MDD generate another model with crosscutting relationships (model 3); and
3. **Results comparison** – the output generated in the stage 2 (model 3) was analyzed and compared with the first model created in the stage 1. Therefore, we compare the results presented by the manually case study with the result of applying the semi-automatic strategy proposed in this paper.

Table 2. Statistics about CMS Case Study

	Manual case study		Semi-automatic case study
Elements	Quantity of elements - Model 1, stage 1	Quantity of elements - Model 2, stage 1	Quantity of elements – Model 3, stage 2
Goals	6	6	6
Softgoals	27	27	27
Tasks	79	79	79
Contribution relationships	108	156	106
Correlation relationships	20	20	20
Crosscutting relationships	6	0	4
Crosscutting concerns	17	0	12

Table 2, summarizes the results achieved; columns 2 and 3 present the quantity of elements comprised in models created in stage 1, while the last column present the quantity of elements created in the stage 2. The CMS PL-AOVgraph model is comprised of 27 softgoals, 6 goals, 79 tasks and 20 correlations. These elements are the same in all models. On the other hand, the amount of contributions and crosscutting relationships are different in each model: in model 1, there are 108 contributions and 6 crosscutting relationships; in model 2, there are 156 contributions and zero crosscut-

ting relationships; and in model 3, there are 106 contributions and 4 crosscutting relationships.

The elements showed like crosscutting concern in Table 2 are the advice elements, i. e., the elements which are repeated along the model.

Comparing the results presented by manual case study and semi-automatic case study (see Table 2), the main difference is the quantity of crosscutting relationships. In the semi-automatic case study were identified 4 crosscutting relationships, while in the manual case study were showed 6 crosscutting relationships; and the quantity of crosscutting concerns, in the semi-automatic case study were identified 12 crosscutting concerns, while in manual case study were identified 17 crosscutting concerns.

Analyzing these relationships, it was noticed that: 5 of the 6 crosscutting relationships manually identified were covered by semi-automatic case study. It could be observed due the level of hierarchy used in both case studies. Hereinafter, it will be explained clearer.

Thus, 2 of them were written in the same manner. However, in one of them, the manual case study presented one additional element of advice than the semi-automatic case study. Figure 8 shows the crosscutting relationship identified by ReqSys-MDD that was equivalent in both case studies, i.e., the source, pointcut and advice elements were equals in both case studies.

In this perspective, 3 of these 5 elements were not written in the same manner, because the semi-automatic strategy defines the source of this relationship with the element which is at the top of the hierarchy and, the manual case study, defines the second level of hierarchy, but they are equivalents. Additionally, these 3 relationships represented 2 relationships in semi-automatic case study, and also the semi-automatic case study identified one more advice element than the manual case study. This element was not identified by manual case study because it was not used only the quantity of relationships, but also the possibility of reuse.

```
01  crosscutting {
02    source: softgoal_ref "Mobility"
03    pointcut PC1: include "Monitor [weather]"
04                  and include "Monitor [terrain conditions]"
05                  and include "Monitor [criminal activity]"
06                  and include "Determinate [safe operating distances and perimeter]"
07    pointcut PC2: include "Monitor [weather]"
08                  and include "Monitor [terrain conditions]"
09                  and include "Monitor [criminal activity]"
10                  and include "Determinate [safe operating distances and perimeter]"
11    advice (around): PC1 {
12      task_ref "Access [maps, terrain and weather conditions and routes]" (and) {}
13    }
14    advice (around): PC2 {
15      task_ref "Provide [location sensitive info]" (and) {}
16    }
17  }
```

Fig. 8. Example of crosscutting relationship identified

Finally, 1 of the 6 crosscutting relationships manually identified was not covered by semi-automatic case study. The reason for this is that the amount of output relationship was not equal to or greater than 3. This relationship grouped 4 elements in advice, i.e., 4 crosscutting concerns.

Therefore, in general, we consider which the crosscutting concerns identified by ReqSys-MDD are correct, but the insights of the requirement engineer can identify other elements, which the semi-automatic strategy could not do.

Thus, this strategy facilitated the crosscutting concerns identification since it identified almost all crosscutting relationships. In this perspective, semi-automatic strategy proposed to help efficiently in the crosscutting concerns identification. More details for this case study can be found in [10].

6 Related Work

In order to respond to the necessity of identifying crosscutting concerns early in the software development lifecycle, some methods were created to systematize and make the execution of this activity easier. Therefore, we made a literature review, which aimed to seek methods and techniques for identifying crosscutting concerns in requirements documents to guide the development of the approach presented in this paper. Then, we could describe two major groups of identification approaches: (i) those which process textual requirements documents; and (ii) those which deal with a specific type of model, for instance, Use Cases and I* models.

Among the methods which process textual documents, we can highlight CCCINPL [1], Theme/Doc [6], DISCERN [11] and Early-AIM [13]. The first difference among them is the identification technique: Theme/Doc performs lexical analysis and the other three perform semantic analysis. Although these three approaches perform the same identification technique, each one is performed in a different way, such as using natural language processing or aspect mining. Other difference between them is the requirement type that is identified as crosscutting concern: Theme/Doc and CCCINPL identify crosscutting concerns in functional or non-functional requirements, but DISCERN and Early-AIM only identify in non-functional requirements.

Among the method which process specific models, we can highlight the approaches which extend UML [4][5] and the approaches which use I*[2][3]. They perform the crosscutting concern identification by different techniques: those which extend UML use semantic analysis and those which work with I* use rules (considering the intrinsic elements of those languages). Furthermore, other difference between them is the requirement type that they identify as crosscutting concern. Only those which work with I* identify crosscutting concerns in both requirement types, the other identify only in non-functional requirements.

This strategy proposed by this paper is in this second group, because it process PL-AOVgraph model and like those process I* models, this strategy identifies crosscutting concern in both requirements types – functional or non-functional requirements. Furthermore, this strategy uses the fan-out analysis as identification technique.

Above all, some of these works, or parts of them, were used as a base to define the strategy proposed here, among them CCCINPL, Theme/Doc and those which works with I*. The first, by the use of a relationship matrix to identify the verbs that influence each requirement; and the others by the heuristics used to identify crosscutting concerns.

Therefore, the contribution of the strategy and tool presented in this paper is to make semi-automatic some heuristics to identify and define crosscutting concerns in PL-AOVgraph models. This application makes a good use of its natural characteristics, such as, the modeling of both functional and non-functional requirements, the ability to make explicit the relationships between the requirements, and then, help to analyze them.

Table 3 presents a summary of the related works, showing the identification technique used by the approach and in what type of requirement it identifies crosscutting concern.

Table 3. Summary of related works

Approaches that process textual requirements documents		
Approach	**Identification technique**	**Requirement types**
Theme/Doc	Lexical analysis	Functional Requirements Non-Functional Requirements
DISCERN	Semantic analysis	Non-Functional Requirements
Early-AIM	Semantic analysis	Non-Functional Requirements
CCCINPL	Semantic analysis	Functional Requirements Non-Functional Requirements
Approaches that process specific models		
Approach	**Identification technique**	**Requirement types**
Crosscutting concern identification with UML	Semantic analysis	Non-Functional Requirements
Identifying crosscutting concern with I*	Rules	Functional Requirements Non-Functional Requirements

7 Final Remarks

This paper presents, briefly, a semi-automatic strategy for crosscutting concerns identification in PL-AOVgraph models. This strategy aims to help the requirement engineer to identify crosscutting concerns and to write the crosscutting relationships properly early in the software development process. This strategy is supported by a tool, named ReqSys-MDD. This support makes the PL-AOVgraph models better modularized and then, it makes them more easily analyzed and mapped to other stages of software development.

The Crisis Management System (CMS) [9] was used as a case study in this paper in order to evaluate the heuristics defined to identify crosscutting concerns and their implementation. This strategy facilitated the crosscutting concerns identification since it identified almost all crosscutting relationships. This result can be due the crosscutting concern identification in manual way is not only by output relationships but also by requirements engineer's insights.

As future works, it is suggested to carry out other studies case to evaluate the efficiency of the tool and, consequently, of the method proposed here, in different contexts. Furthermore, we intend to perform controlled experiments to compare the results obtained by this method and the results obtained by other crosscutting concerns identification approaches. And also, we aim to make our strategy more generic, allowing its use with others goal-based languages.

Acknowledgements. This research was performed with support from The Brazilian National Council for Scientific and Technological Development – CNPq.

References

1. Ali, B.S., Kasirun, Z. M. Crosscutting concern identification at requirement level. In: Malaysian Journal of Computer Science, vol. 21(2), pp.78-87 (2008).
2. Alencar, Fernanda; et al. "Identifying Candidate Aspects with I-star Approach". In: International Conference on Aspect-Oriented Software Development. (2006).
3. Alencar, Fernanda; et al. "Using Aspects to Simplify i* Models". In: International Conference on Requirements Engineering. (2006).
4. Araújo, J.; Moreira, A. An Aspectual Use Case Driven Approach. In: VIII Jornadas de Ingeniería de Software y Bases de Datos (JISBD 2003), Alicante, Spain. (2003).
5. Araújo, J.; et al. Aspect-Oriented Requirements with UML. In: Workshop on Aspect-Oriented Modelling with UML (UML 2002). Dresden, Germany. (2002).
6. Baniassad, E., Clarke, S. Finding Aspects in Requirements with Theme/Doc. In: Proceedings of Early Aspects 2004 (AOSD 2004). Lancaster, United Kingdom, (2004).
7. Filman, Robert E., et al. Aspect-Oriented Software Development. Boston: Pearson Addison-Wesley (2005).
8. Kiczales, G.; et al. Aspect-Oriented Programming. In: ECOOP 1997 - Proceedings of the 11th European Conference on Object-Oriented Programming, p.p. 220–242. Finland (1997).
9. Kienzle, Jörg; Guelfi, Nicolas; Mustafiz, Sadaf. Crisis Management Systems: A Case Study for Aspect-Oriented Modeling. Technical Report. McGill University, Montreal, Canada (2009).
10. Medeiros, Maíra de Faria Barros. Identificando Interesses Transversais em Modelos de Requisitos PL-AOVgraph. Master Dissertation – Federal University of Rio Grande do Norte, Natal, RN (2013).
11. Rosenhainer, L. The DISCERN Method: Dealing Separately with Crosscutting Concerns. In Proceedings of OOPSLA Early Aspects 2005. San Diego, USA (2005).
12. Sampaio, Américo; et al. EA-Miner: a tool for automating aspect-oriented requirements identification. In: International Conference on Automated Software Engineering (2005).
13. Sampaio, Américo; Rashid, Awais; Rayson, Paul. Early-AIM: An Approach for Identifying Aspects in Requirements. In: International Conference on Requirements Engineering (2005).
14. Santos, Lidiane Oliveira dos. PL-AOVgraph: uma extensão de AOV-Graph para linha de produto de software. 84 f. Monograph in Computer Science – State of University of Rio Grande do Norte, Natal (2010).
15. Santos, Lidiane Oliveira dos. ReqSys-MDD: Uma ferramenta para mapeamento entre modelos de features e requisitos em linhas de produto de software. 113 f. Master Dissertation – Federal University of Rio Grande do Norte, Natal, RN (2012).
16. Silva, Lyrene Fernandes da. Uma estratégia orientada a aspectos para modelagem de requisitos. 222 f. PhD Thesis – Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro (2006).
17. Sommerville, Ian. Engenharia de Software. 8. ed. São Paulo: Pearson Addison-Wesley, 552 p. (2007).

Uma Abordagem para Engenharia de Requisitos no Domínio de Software Embarcado

Milena R. S. Marques, Eliane Siegert, Lisane de Brisolara

Ciência da Computação, Grupo de Arquiteturas e Circuitos Integrados,
Centro de Desenvolvimento Tecnológico - CDTec,
Universidade Federal de Pelotas, Brasil
{mrsmarques, esiegert, lisane}@inf.ufpel.edu.br

Resumo. Este artigo apresenta uma abordagem para engenharia de requisitos orientada a modelos para o domínio de software embarcado. Para suportar a modelagem e a gerência dos requisitos são utilizadas as linguagens de modelagem UML, SysML e o perfil MARTE, todos padrões da OMG. Desta forma, a abordagem permite a completa modelagem de requisitos funcionais e não funcionais frequentes no domínio de sistemas embarcados, além da gerência desses requisitos desde a especificação até a validação. Um estudo de caso é usado para demonstrar a abordagem proposta.

Palavras Chave: UML, SysML, MARTE, Engenharia de Requisitos, Sistemas Embarcados, rastreabilidade.

1 Introdução

Sistemas embarcados executam funções dedicadas e estão inseridos em um sistema maior [1]. Usualmente, estes sistemas são constituídos de hardware e de software e incluem vários requisitos não funcionais (RNF) relacionados a desempenho, consumo energético e tamanho de memória. Estes requisitos afetam decisões de projeto e tornam-se restrições rígidas aos engenheiros de software embarcado. Estas restrições rígidas, aliadas a alta complexidade e prazos de entrega curtos, dificultam o projeto e requerem processos de desenvolvimento especializados.

Para lidar com projetos complexos, linguagens de alto nível de abstração têm sido utilizadas e integradas em metodologias orientadas a modelos (MDE, do inglês, *Model-driven Engineering*). Dentre as linguagens, destaca-se a UML, a qual é padrão para modelagem de software [2]. No entanto, a UML não suporta a modelagem de aspectos específicos do domínio de sistemas embarcados, como por exemplo, requisitos relacionados a consumo de energia, desempenho e temporização, os quais precisam ser considerados desde a modelagem do sistema. O perfil MARTE [3] estende a UML resolvendo esta limitação. No entanto, estes RNFs são aspectos transversais que influenciam vários artefatos do sistema [4], o que indica a necessidade de um procedimento que permita além da modelagem, a completa gerência destes requisitos.

A engenharia de requisitos define um caminho desde o levantamento dos requisitos até a verificação/validação do software, incluindo a gestão dos requisitos durante todas as fases do projeto [5]. A gestão ajuda a equipe a identificar, controlar e acompanhar as necessidades e suas mudanças a qualquer momento do projeto. Para gerir os requisitos, é importante dar suporte a rastreabilidade dos mesmos. Para isso, é preciso estabelecer relacionamentos entre os requisitos e artefatos do projeto como casos de uso, componentes e casos de teste [6]. A linguagem SysML [7], oferece recursos para a modelagem de requisitos e relacionamentos entre eles e por esta razão vem sendo também aplicada no domínio de sistemas embarcados [6].

O objetivo deste trabalho é apresentar a MDEReq, uma abordagem para engenharia de requisitos orientada a modelos focada no domínio de software embarcado, que suporta a gerência de requisitos. Nesta abordagem, a rastreabilidade de requisitos é suportada através de notações SysML. Mas, além desta linguagem, a abordagem também faz uso de modelos UML/MARTE. Estes modelos são integrados permitindo a completa modelagem do sistema usando uma única ferramenta de modelagem e facilitando a rastreabilidade dos requisitos em todas as fases do projeto.

Este artigo está organizado da seguinte forma. A Seção 2 discute trabalhos relacionados. A abordagem proposta é apresentada na Seção 3 e o estudo de caso na Seção 4. Por fim, as conclusões e trabalhos futuros são discutidos na Seção 5.

2 Trabalhos Relacionados

A modelagem de requisitos de sistemas embarcados visando a rastreabilidade foi estudada por outros autores, dentre eles Albinet [8] e Dubois [6]. Ambos propõem uma metodologia baseada em modelos e rastreabilidade com foco em aplicações automotivas [8]. A diferença entre eles está na rastreabilidade dos requisitos. Em [8] por não ser um modelo integrado a rastreabilidade é feita somente no mesmo nível de abstração. Já em [6], um meta-modelo é proposto o qual inclui SysML e permite a rastreabilidade de requisitos em todos os níveis de abstração do projeto.

O diferencial da MDEReq em relação aos trabalhos citados é que a abordagem proposta é baseada em UML e extensões padronizados pela OMG, sem redefinir estas linguagens e, portanto, é suportada por qualquer ferramenta de modelagem disponível no mercado. Além disso, nossa abordagem integra o processo de engenharia de requisitos no desenvolvimento de software embarcado, o que favorece a gestão dos requisitos. Além disso, a MDEReq pode ser utilizada em qualquer aplicação do domínio de sistemas embarcados, pois não é focada em nenhum subdomínio específico.

3 Abordagem MDEReq

A MDEReq (do inglês, *Model Driven Engineering for Requirement Management*), é uma abordagem orientada a modelos para engenharia de requisitos de software embarcado. Esta abordagem visa facilitar a especificação de softwares embarcados através da construção de modelos abstratos baseados em linguagens padronizadas, que permitam a modelagem e gestão de requisitos funcionais e não funcionais (temporais,

de confiabilidade, consumo, etc). Desta forma, a abordagem contribui para que projetistas de sistemas embarcados tenham um controle mais efetivo das mudanças de requisitos e de seus impactos nos demais artefatos do projeto em todas as etapas do processo. A abordagem proposta baseia-se nos princípios da MDE [9], onde modelos guiam todas as decisões de projeto e são detalhados e refinados a cada etapa, até que a implementação seja alcançada.

A MDEReq é baseada no modelo de processo de software tradicional, onde a primeira atividade é a definição de requisitos. Esta é parte de um procedimento conhecido como Engenharia de Requisitos, o qual define atividades que envolvem concepção, levantamento, elaboração, negociação, especificação, validação e gestão de requisitos [5]. Com base neste procedimento, foi definido o workflow da MDEReq, ilustrado na Figura 1, o qual é composto de quatro atividades: o levantamento, a análise e especificação, a validação e a gestão dos requisitos. Segundo Pressmam [5], estas são as principais atividades da engenharia de requisitos.

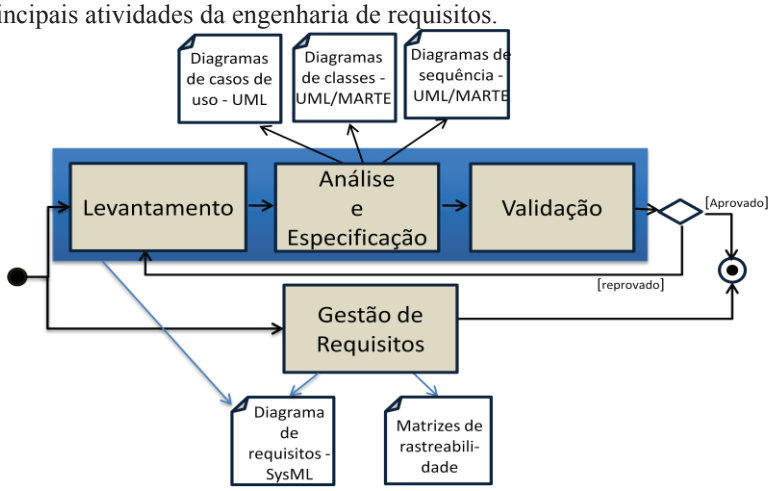


Fig. 1. Workflow da abordagem MDEReq

A atividade de levantamento de requisitos identifica e classifica os requisitos do sistema. Nesta atividade da MDEReq, o primeiro artefato gerado é uma lista de requisitos, onde os requisitos devem ser classificados em funcionais e não funcionais e podem ser priorizados. Esta lista serve como base para a criação do diagrama de requisitos, que indica os relacionamentos entre os requisitos. Este artefato evolui nas próximas atividades do workflow a medida em que os modelos são refinados.

Na atividade de análise e especificação da MDEReq, as informações obtidas no levantamento são refinadas [5] através da construção de diagramas UML de casos de uso, de classes e de sequência. Estes diagramas são adotados para a modelagem de diferentes visões dos requisitos do sistema. Para modelar aspectos específicos do domínio de sistemas embarcados (ex: *deadlines* e período de tarefas ou outros RNFs ou ainda a interação com componentes físicos do sistema), estes diagramas são decorados com estereótipos do MARTE. Após, o diagrama de requisitos é incrementado de forma a incluir relacionamentos entre requisitos e elementos dos diagramas UML,

tais como casos de uso ou diagramas de sequência. Da mesma forma, durante a atividade de validação, casos de teste são definidos e relacionados aos requisitos no diagrama de requisitos.

O workflow define também uma atividade de gestão de requisitos que ocorre em paralelo as demais. A gestão na MDEReq baseia-se na geração de matrizes de rastreabilidade. Tais matrizes são geradas a partir do diagrama de requisitos da SysML. Este diagrama permite definir diferentes tipos de relacionamentos entre requisitos e entre requisitos e outros artefatos (ou seja, elementos do modelo), além de mostrar a hierarquia dos requisitos. Na MDEReq, são utilizados os relacionamentos *derive*, *composite*, *refine*, *verify* e *satisfy* para dar suporte as matrizes de rastreabilidade em diferentes níveis de abstração ou em diferentes fases do projeto.

Nas etapas iniciais do projeto, quando os requisitos estão sendo levantados, é gerada a matriz de rastreabilidade de requisito para requisito usando relacionamentos do tipo *derive* e *composite*. Esta matriz tem a finalidade de apoiar a gestão de mudanças nos requisitos, quando um requisito é derivado a partir de um requisito base ou quando um requisito é decomposto em outros. A matriz de rastreabilidade de requisitos para os artefatos de projeto é gerada após a atividade de análise e especificação através dos relacionamentos de *refine* e *satisfy*, indicando que um elemento do modelo refina ou satisfaz um requisito. Neste caso, os elementos do modelo são diagramas de casos de uso e de sequência. Desta forma, quando o requisito é alterado, é possível identificar de forma gráfica quais elementos do modelo devem ser alterados. Por fim, a matriz de rastreabilidade de requisitos para casos de testes é gerada através do relacionamento de *verify*. Esta matriz se destina a equipe de testes e indica quais testes são usados para testar a satisfação de um requisito e também pode indicar quais testes devem ser redefinidos em função de uma mudança em um requisito.

As matrizes propostas na MDEReq permitem que os envolvidos no projeto tenham várias visões dos requisitos do sistema em vários níveis de abstrações, facilitando a gerência e o planejamento das mudanças ao longo do projeto, além de permitir o controle mais rígido sobre requisitos funcionais e não funcionais do sistema.

4 Estudo de caso

Seguindo a MDEReq, a modelagem do software de controle de freio ABS foi realizada para este estudo de caso, iniciando com a listagem de requisitos e diagrama de requisitos inicial e seguindo com o detalhamento destes requisitos através de diagramas UML/MARTE. O diagrama de requisitos é incrementado a cada atividade, incluindo novos elementos e relacionando-os aos requisitos do sistema, sejam eles funcionais ou não funcionais. Por limitação de espaço, não são apresentados todos os diagramas usados neste estudo de caso. Como muitos trabalhos já demonstram a modelagem UML/MARTE de sistemas embarcados, este estudo foca na gestão de requisitos. A Figura 2 ilustra o diagrama de requisitos completo (após as evoluções), o qual serve como base para a geração das matrizes de rastreabilidade propostas pela MDEReq. A Figura 3 ilustra a matriz requisito x requisito gerada a partir deste diagrama. Por limitação de espaço, não são apresentadas as outras duas matrizes. A matriz ilus-

trada indica, por exemplo, que uma mudança no requisito Monitorar Velocidade exige uma verificação nos requisitos Ler velocidade das rodas e Temporização do ABS.

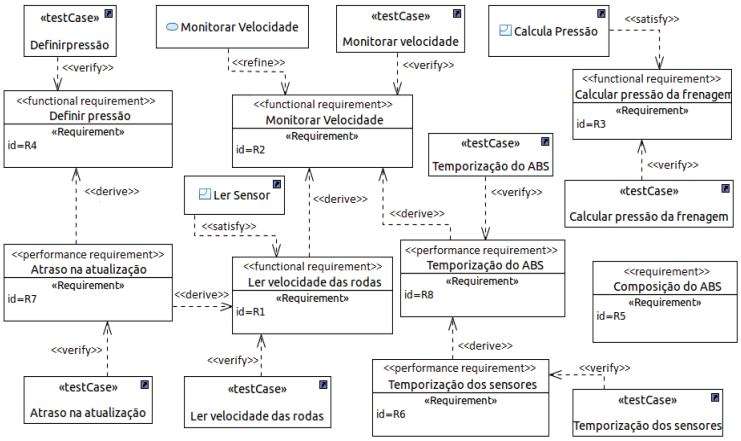


Fig. 2. Diagrama de Requisitos do sistema de freio

	Ler Velocidade das rodas	Monitorar Velocidade	Calcular pressão	Diminuir pressão	Composição do ABS	Temporização do Sensor	Atraso na atualização	Temporização do ABS
Ler Velocidade das rodas							✓	
Monitorar velocidade	✓							✓
Calcular pressão								
Diminuir pressão							✓	
Composição do ABS								
Temporização do Sensor								
Atraso na atualização								
Temporização do ABS						✓		

Fig. 3. Matriz de Rastreabilidade de requisito para requisito.

5 Conclusões

Este artigo apresentou uma abordagem de engenharia de requisitos orientada a modelos para o domínio de software embarcado, chamada MDEReq. Esta abordagem baseia-se em modelos UML, SysML e MARTE para suportar a completa especificação

de requisitos, bem como a gestão de requisitos para o domínio de sistemas embarcados. Esta integração das notações é importante neste domínio, sobretudo pelo grande número de RNFs que devem ser especificados e geridos. A gestão proposta é baseada em matrizes de rastreabilidade geradas a partir do diagrama de requisitos. Para demonstrar a abordagem, um estudo de caso foi usado, o qual envolve um sistema embarcado crítico com vários RNF a serem modelados e gerenciados. Atualmente, as matrizes de rastreabilidade são geradas manualmente, no entanto, a implementação de um suporte automatizado está prevista em nossos trabalhos futuros.

Referências

1. Marwedell, P.: Embedded Systems Design 1st edn. Springer (2006)
2. OMG: UML. In: Object Managment Group. (Accessed 2012) Available at: http://www.omg.org/spec/index.htm
3. OMG: MARTE. In: Object Management Group. (Accessed 2012) Available at: http://www.omg.org/spec/index.htm
4. Werhmeister, M.: An aspect-oriented model-driven engineering approach for distributed embedded real-time systems. In UFRGS, ed. : Tese (Doutorado) – Universidade Federal do Rio Grande do Sul, Porto Alegre, p.206 (2009)
5. Pressman, R.: Engenharia de Software: Uma abordagem profissional 7th edn. AMGH, Porto Alegre (2011)
6. Dubois, H., Peraldi-Frati, M.-A., Lakhal, F.: A model for requirements traceability in an heterogeneous model-based design process: Application to automotive embedded systems. In : Engeneering of Complex Computer Systems, Oxford, pp.233-242 (2010)
7. OMG: SysML. In: Objet Management Group. (Accessed 2012)Available at: http://www.omg.org/spec/index.htm
8. Albinet, A., Begoc, S., Boulanger, J.-L., Casse, O., Dal, I., Dubois, H., Lakhal, F., Louar, D., Peraldi-Frati, M.-A., Sorel, Y., Van, Q.-D.: The MeMVaTEx methodology: from requirements to models in automotive application design. In : 5th European Congress ERTS Embedded Real Time Software, Toulouse, pp.19-21 (2010)
9. Selic, B.: UML 2: A model-driven development tool. IBM Systems Journal 45(3), 607-620 (2006)

Una estrategia de integración de Modelos de Objetivos con Análisis Comunicacional

María Carmen Leonardi[1], Roxana Giandini [2]

(1) INTIA; Universidad Nacional del Centro de la Provincia de Buenos Aires
cleonard@exa.unicen.edu.ar
(2)LIFIA, Facultad de Informática, Universidad Nacional de La Plata
giandini@lifia.info.unlp.edu.ar

Abstract. El Análisis Comunicacional (CA del inglés Comunicational Analysis) es un método basado en la comunicación para el modelado de proceso de negocio y requisitos que propone el análisis de los sistemas de información desde una perspectiva comunicacional. Este método se basa en analizar los flujos de información. En un contexto de desarrollo To-Be, donde no se cuenta con documentación previa para analizar, este tipo de perspectiva puede ser complementada con un modelo motivacional, comenzado el proceso de análisis desde una perspectiva de objetivos de la organización, hasta llegar a modelar lo que se quiere hacer concretamente. Este trabajo propone una estrategia de integración entre CA y un modelo de objetivos. Esta integración permitirá captar la intencionalidad y objetivos de la organización en sus diferentes niveles de abstracción, dándole racionalidad al sistema y alineando los procesos con los objetivos.

Keywords: Análisis Comunicacional, Modelo de Objetivos, Procesos de Negocio.

1 Introducción

En la Ingeniería de Software, el proceso de desarrollo de software no siempre cuenta con un contexto definido o sistema previo (automatizado o no) donde llevarse a cabo. Aún en la actualidad existen situaciones de exploraciones de negocios innovadores o nuevas oportunidades de negocio en base a adelantos tecnológicos [12]. Este es un contexto de desarrollo To-Be, es decir, se pretende construir el sistema de información que dará soporte al negocio. En este tipo de contextos, comenzar el análisis desde una perspectiva de objetivos permite modelar la visión de lo que se quiere para luego ir construyendo los procesos que permitirán llevar a cabo esa visión. Tener presente y modelar la visión es fundamental para los subsecuentes procedimientos. Distintas visiones y objetivos generales producirán diferentes procesos operacionales [15]. Los modelos de objetivos son ampliamente utilizados en Ingeniería de Requisitos. La razón de su uso se debe al alto grado de abstracción que ofrecen, su mayor estabilidad comparada con los requisitos definidos para implementarlos, la posibilidad que brindan para explorar alternativas de solución, la verificación de la completitud de los requisitos y la trazabilidad desde el contexto organizacional hasta los requisitos [25].

El Análisis Comunicacional (CA) es un método de Ingeniería de Requisitos para el desarrollo de Sistemas de Información (SI) [6]. Se basa en una perspectiva comunicacional, poniendo énfasis en las interacciones comunicacionales que ocurren entre el SI y su entorno. CA está integrado a la metodología de desarrollo de software OO-method [23], de esta manera se cuenta con una metodología de desarrollo alineada a Model Driven Architecture (MDA), [19] que parte de los requisitos desde una perspectiva comunicacional hasta la obtención del código. En un contexto de desarrollo To_Be (from Scratch), comenzar el análisis desde una perspectiva comunicacional puede ser no tan efectiva, ya que ésta se concentra en las interacciones ente el actor y el sistema, útil particularmente cuando se quiere hacer el análisis de procesos existentes y sus posibles flujos de comunicación [15]. Por esta razón, en un contexto de desarrollo To-Be una perspectiva comunicacional (y en particular CA) puede ser complementada con una perspectiva motivacional, como propone el modelo Business Motivation Model [20] comenzado el análisis desde una perspectiva de la visión y objetivos de la organización, "lo que quiere ser" (el fin) que se irá descomponiendo hasta "lo que decida hacer"(los medios para llegar la fin). Esta integración entre objetivos y procesos permitirá captar la intencionalidad y objetivos de la organización en sus diferentes niveles de abstracción, dándole racionalidad al sistema y alineando los procesos con los objetivos. Este trabajo está organizado de la siguiente manera: en la Sección 2 se describe brevemente la estrategia de Análisis Comunicacional. En la sección 3 se presenta nuestra propuesta de integración de CA con un modelo de objetivos y se ejemplifica a partir de un caso de estudio. En la Sección 4 se analiza nuestra propuesta respecto a otras existentes. Finalmente, en la Sección 5 presentamos conclusiones y futuros trabajos.

2 El método CA

CA es un método basado en la comunicación para el modelado de procesos de negocio y requisitos que propone el análisis de los sistemas de información desde una perspectiva comunicacional [8]. Este método está soportado dentro de un Framework de desarrollo MDA. Este Framework puede verse como un proceso de ingeniería *forward* que permite la generación de código fuente desde los modelos de requisitos. CA está organizado en 5 niveles de refinamiento: el nivel N1: Sistemas/Subsitemas) describe la estructura y estrategia organizacional. En el nivel N2: Procesos se identifican las interacciones comunicacionales entre el ambiente y el sistema de información y se detectan los procesos. En el N3: Interacciones comunicativas, se detallan los procesos y las comunicaciones del nivel anterior. El nivel 4: Ambiente, se enfoca en el diseño de las interfases y el almacenamiento de los datos. Finalmente el N5: Operativo, se realiza la implementación.

CA propone el modelado de los Diagrama de Eventos Comunicacionales (CED de su nombre en ingles), técnica de modelado de procesos de negocio que adopta una perspectiva comunicacional. Un CED está formado por eventos comunicacionales. Un evento describe acciones relacionadas al manejo de la información que se llevan a cabo de manera completa e ininterrumpidamente al recibir un estímulo externo. Los eventos son identificados con un criterio de unidad de eventos comunicacionales [9]

el cual guía la modularidad de los procesos de negocio. Este criterio consiste en tres conceptos que se tienen que cumplir para detectar un evento: unidad de desencadenamiento o *trigger* (implica que el evento ocurre como respuesta a una interacción externa y por consiguiente algún actor la desencadenó); unidad de comunicación (que implica que cada evento debe producir información relevante y nueva) y la unidad de reacción (que implica que el evento es una composición de acciones sincrónicas). Asociados a los eventos están los actores: el actor primario que desencadena el evento y provee la información; el actor interfase que actúa físicamente con el sistema de información y el actor receptor, que es informado de la realización del evento. Las comunicaciones en los eventos son: comunicación de entrada que es una interacción comunicativa que proviene del actor primario y alimenta al SI con nueva y significativa información y comunicación de salida es una interacción de comunicación que consulta el SI y brindando la información al actor receptor. Los eventos son organizados en el diagrama de acuerdo a su precedencia. CA completa el modelado de los CED con plantillas para describir los eventos y las estructuras de mensajes para especificar la información. Los modelos producidos por CA constituyen la entrada para la estrategia de transformación que obtiene modelos OOMethod [23]. Esta Estrategia, implementada en Integranova, permite posteriormente compilar los modelos OOMethod para obtener código fuente. De esta manera, CA cubre los aspectos de modelo de requisotas y procesos de negocio dentro de Framework de desarrollo MDD. En [6,7, 8, 9, 23], se detalla esta propuesta.

3 Una propuesta de integración de objetivos con CA

En esta sección se presenta la integración de un modelo de objetivos con CA. La estrategia comienza con los objetivos generales de la organización que se van refinando hasta llegar al nivel de tareas, en el que se pueden definir los procesos comunicacionales modelados con CED. De esta manera quedan alineados los objetivos estratégicos con la identificación de las tareas y las interacciones comunicacionales que se llevarán a cabo para cumplirlos. El modelo de objetivos complementa el Nivel 1 de CA, en donde se realiza una descripción estratégica del sistema organizacional y su entorno.

Los pasos propuestos son los siguientes:

1- Definición del Modelo de Objetivos y tareas
2- Definición de los Actores
3- Modelado de Tareas
4- Definición de procesos

Los pasos 1 y 2 no son necesariamente secuenciales, pueden ser realizados en paralelo o en el orden inverso. A continuación se desarrolla y se ejemplifica cada paso.

3.1 Definición del Modelo de Objetivos y tareas

El primer paso es la definición de un modelo de objetivos de la organización. En esta propuesta se define el concepto de objetivo en base a los conceptos presentados en [9] donde se define un objetivo organizacional como un objetivo que afecta a las acciones de un sistema organizacional (es decir, el comportamiento de sus actores).

Asimismo, consideramos los objetivos funcionales como aquellos que puede refinarse hasta llegar a modelar objetivos operacionalizables que determinarán los procesos del negocio.

Se modela una jerarquía de objetivos de la organización cuyos niveles inferiores son los que dan lugar a los procesos del negocio, siguiendo un modelo clásico and/or de objetivos-tarea, como el que se propone en varias de las estrategias (Sección 4). Esta jerarquía representa objetivos a diferentes niveles de abstracción desde los objetivos estratégicos hasta llegar a los objetivos operacionales [28]. La estrategia se basa en un grafo AND/OR de objetivos generales que representan lo objetivos de la organización y se termina en nodos terminales que representan tareas; de esta manera se logra operacionalizar todos los objetivos [15]. Básicamente, la descomposición permite modelar niveles de abstracciones [13] siendo los niveles superiores intencionales y los inferiores operacionales. Un objetivo a un determinado nivel describe QUE necesita ser realizado. Al mismo tiempo este objetivo puede ser considerado como un fin (POR QUE) para otro objetivo de más bajo nivel, así como también como un medio (COMO) para otro objetivo de más alto nivel. Los objetivos que están en el último nivel son los operacionales, en algunas estrategias, estos objetivos ya son considerados como tareas que pueden ser asignadas a un determinado rol dentro de la organización [10,13, 29]. En nuestro modelo vamos a caracterizarlos como tareas. De esta manera quedarán explícitamente alienados los procesos que llevan a cabo estar tareas con los objetivos de la organización que se pretenden cumplir. La Fig.1 muestra el modelo utilizado, que adopta la descomposición y el nivel de abstracción presentado en [14] e incorpora la componente tarea como en [10, 13, 29].

3.1.1 Elicitación de los objetivos

En este apartado, seleccionamos algunas estrategias para la construcción de nuestro modelo de objetivos dentro del contexto de desarrollo To_Be y dependiendo del tamaño y complejidad del problema:

- Se puede aplicar el método de generación de ideas, estrategia que involucra activamente a los stakeholder utilizando métodos estilo brainstorming que permite descubrir nuevos sub-objetivos a partir de los objetivos [18].
- Si la organización no es compleja, la misma estructura jerárquica del grafo permite una estrategia básica de elicitación Top Down o Bottom Up a través de la preguntas de ¿Por que? ¿Qué? y ¿Cómo? [13, 14, 15].
- En organizaciones complejas, se puede utilizar la estrategia presentada en [28], que propone guías para elicitar objetivos en los diferentes niveles de la organización: estratégico (la visión de la empresa, los objetivos de mas alto nivel), tácticos (son los objetivos que soportan al nivel estratégico) y finalmente los objetivos operacionales que son los que se pueden materializar en los procesos del negocio. Si bien nuestro grafo no sigue esta clasificación de objetivos, el nivel de abstracción de los momos y compatible con la misma. Entonces, estas guías pueden ser utilizadas porque responde a la visión de un modelo general de objetivos representado con un grafo extendido AND/OR en donde la idea de descomposición es un proceso gradual que abarca transformaciones causales de objetivos de alto nivel de abstracción en uno o mas sub-

objetivos hasta poder describir tareas que se pueden operacionalizar a través de un proceso de negocio definido razonablemente.

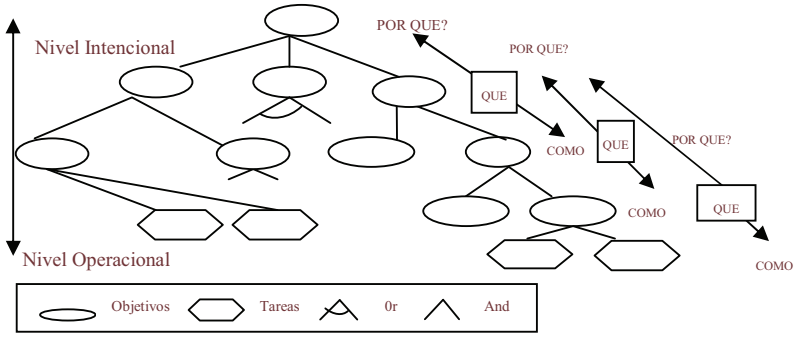


Fig. 1. Modelo de Objetivos

El modelo propuesto es un modelo general que cubre una especificación básica de objetivos. Dependiendo del nivel de complejidad de la organización, este modelo puede ser reemplazado o incrementado por modelos más sofisticados que permitirán realizar un análisis más exhaustivo de los objetivos (Sección 4).

3.1.2 El Modelo de Objetivos aplicado a un Caso de Estudio

Ejemplificaremos la propuesta con el siguiente caso: Una pequeña empresa de tambos desea comenzar un nuevo emprendimiento de venta de productos lácteos creando una empresa "Tradición Tandilense" que pretende ser una pequeña firma de productos gourmet(quesos especiales y dulce de leche). Al principio, tercerizarán la producción (producción a fason[1]). En cuanto a la comercialización, la empresa abastecerá de manera exclusiva a un importante comercio de la zona (y sus sucursales) por lo que el excedente deberá ser comercializado a particulares y no a otros comercios. Desean convertirse en la principal abastecedora de su principal Comprador. Si bien es una empresa pequeña y no tiene muchas variabilidad en sus operaciones, describir los objetivos de manera explícita y sencilla ayuda a tener en claro que se quiere ser y de que manera organizar el trabajo para poder cumplirlos. La fig. 2 muestra un modelo parcial de objetivos, se puede observar que el objetivo "Vender el excedente a particulares" por el momento no ha seguido siendo explotado, ya que al ser una empresa que recién comienza y tiene un comprador muy importante que comprará casi el total de las producciones iniciales, se priorizan los otros objetivos, concentrándose en los aspectos de la producción y abastecimiento a este cliente. También aparece como objetivo Disponer de un empleado exclusivo para cada sucursal que no se sigue analizando ya que este trabajo se centra en las heurísticas de definición de procesos a partir de objetivos funcionales y no en el análisis de alternativas de objetivos.

[1] Término utilizado en el ámbito industrial para señalar la manufactura de un producto por mandato de un tercero, dueño de una marca, en el cual éste puede proveer de las materias primas e insumos que son necesarias para la misma.

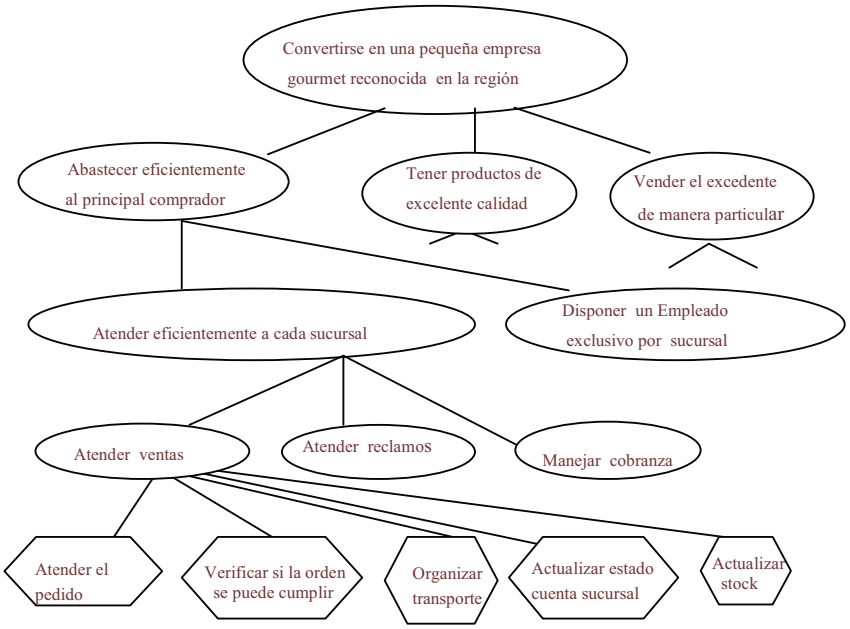


Fig. 2. Modelo parcial de objetivos de la empresa Tradición

3.2 Definición de los Actores

En CA, un actor organizacional es todo actor cuyo comportamiento es significativo para la organización y que, bien por pertenecer a la propia organización o a su sistema de interés, queremos modelarlo (normalmente en forma de roles) para evidenciar su participación en los sucesos comunicacionales. En [1] se propone una taxonomía de stakeholders (onion model) en donde se define los tipos de stakeholders en los diferentes niveles que involucran al sistema (representados en capas comenzando con el núcleo que representa el producto o servicio requerido hacia el nivel que representa el contexto que absorbe a la organización en donde se instalará el producto). Esta taxonomía concuerda con la visión de los sistemas que tiene CA (sistema de información, sistema de información computarizado, sistema organizacional), con lo cual puede ser utilizada para descubrir y definir los actores en nuestra propuesta.

3.2.1 El Modelo de Actores aplicado a un Caso de Estudio

En el caso del ejemplo, se definen los actores de la organización. La empresa cuenta con tres personas que cumplen distintos roles: **Vendedor**, **Encargado de cámara**, **Transportista**, (estos dos últimos a cargo de una misma persona), **Jefe general** y **Controlador de la producción** (estos dos últimos, por el momento a cargo de una misma persona). Otro actor fundamental son las sucursales, ya que cada una de ellas

se maneja como un negocio independiente a cargo de un encargado, de esta manera consideramos un rol **Encargado de sucursal** para la interacción de la empresa.

3.3 Modelado de Tareas

El último nivel del grafo representan las tareas, que en esta etapa deben especificarse con más detalle. Las distintas estrategias estudiadas complementan la información a través de diversos enfoques (Sección 4). En nuestra propuesta definimos un listado separado con las especificaciones detalladas de cada tarea que complemente el ultimo nivel del grafo, siguiendo la idea de i*[10] y [15, 16]. De esta manera, ambos modelos (objetivos y tareas) quedan más sencillos y fáciles de comprender. Por cada **tarea** se define información complementaria:
Responsables: ¿Qué roles son los encargado de llevar a cabo la tarea? Esta idea está basada en la asignación de tareas a roles [15], la asignación de un agente a un objetivo operacional de KAOS [29] y la modelización de la dependencia Actor/Tarea en el Modelo de la Razón Estratégica (SR de su original en inglés) de i*. Pueden ser varios responsables: uno cuyo rol será recepcionar los datos de entrada (rol de interfaz en CA) y otro(s) para ejecutar cada una de las acciones de la tarea (rol de soporte).

Datos de Entrada: ¿Qué datos o información necesita la tarea? ¿Son todos obligatorios? ¿Hay opcionales? La idea de datos de entrada está considerada en [15,16], y en i* con la relación de dependencia hacia un recurso.
Responsable de la entrada: ¿Qué rol es el encargado de brindar estos datos? Ó ¿Qué otra tarea produce estos datos?
¿Quien desencadena la tarea? ¿Que rol desencadena o solicita o inicia la tarea?
Salida: ¿Qué datos (que resultado) produce?, la misma idea de [15,16], y similar al concepto de dependencias del modelo SR de i*.
Receptor de la salida: ¿Quien es el receptor de esa salida? Este dato es similar a la idea de dependencias entre actores del Modelo de Dependencias Estratégicas (SD de su nombre en inglés) de i*.
Acciones: una tarea está compuesta por acciones (indivisibles) que ejecutadas en un determinado orden logran llevar a cabo la tarea.

En este listado pueden aparecer tareas nuevas, que no fueron modeladas en el grafo de objetivos por ser de más bajo nivel o ser de soporte a otras tareas. En los casos en que la organización es simple, el realizar las preguntas de cada ítem ayuda en la elicitación de la información. En [15,28] existen guías que ayudan a la asignación de roles a determinados objetivos/tareas cuando la organización es de mayor complejidad.

3.3.1 Modelado de Tareas aplicado al caso de estudio.

Continuando con el ejemplo, una vez establecido el (o parte del) modelo de objetivos, es momento de especificar las tareas representadas en el nivel mas bajo del mismo. Por cuestiones de espacio, la Fig. 2 sólo muestra parte del refinamiento del objetivo "Atender ventas", que al ser refinado, aparecieron tareas concretas para satisfacerlo:

Tarea: Atender el pedido
Responsables: Vendedor
Datos de Entrada: pedido
Responsable de la entrada: Encargado de Sucursal
¿Quien desencadena la tarea? Encargado de Sucursal
Salida: orden ingresada
Receptor de la salida: Vendedor
Acciones: recibir el pedido vía e-mail, teléfono o personalmente.
Genera una orden con el pedido, el número de sucursal y número de orden.

Tarea: Verificar si la orden se puede cumplir
Responsables: Encargado de Cámara
Datos de Entrada: orden ingresada
Responsable de la entrada: Vendedor
¿Quien desencadena la tarea? Vendedor
Salida: la orden ingresada se puede cumplir (orden verificada) o No hay en stocks algunos de los productos de la orden ingresada (orden no verificada)
Receptor de la salida: Vendedor o Encargado de Sucursal
Acciones: el encargado de cámara verifica si hay suficiente stock

Tarea: Organizar transporte
Responsables: Transportista
Datos de Entrada: orden verificada
Responsable de la entrada: Vendedor
¿Quien desencadena la tarea? Vendedor
Salida: fecha de traslado de orden
Receptor de la salida: Vendedor o Encargado de Sucursal
Acciones: verificar días libres o un viaje por zona de sucursal con capacidad para transportar la orden verificada, asignar la orden al viaje en ese día, informar

Tarea: Actualizar cuenta sucursal
……..

Tarea: Actualizar stock
…...

3.4 Definición de procesos

Un proceso describe una secuencia de tareas dentro de la organización que son realizadas para alcanzar un objetivo del negocio. Dentro de nuestra propuesta, consideramos al proceso desde un punto de vista comunicacional, es decir se define al proceso mediante un CED. Un evento comunicacional es un conjunto de acciones relacionadas a la información (adquirir, almacenar, procesar, retornar, distribuir) que son llevadas a cabo de una manera completa e ininterrumpidamente [6]. Por esta razón, dentro de nuestra definición de procesos, debemos poner énfasis en tareas definidas en el punto anterior para detectar la información que éstas producen o consumen. En

un contexto de un sistema To_Be, es difícil establecer un criterio para definir que conjunto de tareas formarán un proceso. Definimos las siguientes heurísticas:

Heurística 1: Definición de un proceso

Por cada subárbol del grafo de objetivos que tenga como raíz un nodo que tenga al menos una tarea como descendiente, se define un CED para satisfacer a ese objetivo (Fig. 3). El nombre del proceso será el nombre del objetivo.

Sub-Heurística 1.1: las ramas que contengan objetivos, podrán ser consideradas un subproceso (en este trabajo en particular, como se modela un CED, no se tiene en cuenta esta abstracción, sin embargo es útil mencionarla para una futura generalización de la propuesta)

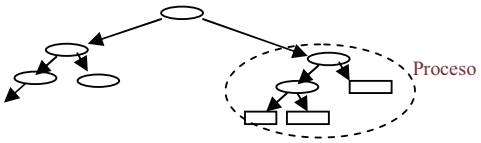


Fig. 3. Determinación de un proceso

En el modelo parcial de la figura 2 que representan parte de los objetivos de la organización en estudio, el objetivo operacional "Atender ventas", ya pudo ser refinado en tareas concretas para satisfacerlo. Este objetivo se convierte en un Proceso que será especificado con un diagrama de eventos cuyo nombre será Atender Ventas.

Heurística 2: Definición de eventos comunicacionales

Se analizan todas las hojas del subárbol que representan las tareas par definir los eventos comunicacionales. Se procede a detectar los eventos comunicacionales a partir de estas tareas. Se debe analizar cada tarea desde una perspectiva comunicacional, verificando si cumplen con el "criterio de unidad" (ver sección 2) que les permitirá ser definidas como eventos. Si cumplen, entonces:

- la tarea se convierte en un evento,
- el actor desencadenante de la tarea se convierte en el actor primario
- si hubiera un actor receptor de la tarea, éste se convierte en el receptor del evento
- las entradas y salidas de la tarea provistas por los actores correspondientes se convierten en las comunicaciones de entrada y salida del CED.
- en el caso que hubiera sido detectado un único responsable de la tarea, éste será el actor interfaz del evento. Si hay varios responsables es porque la tarea tiene una serie de acciones que la describen, se detecta el responsable de la acción que recepciona los datos del actor primario que será el actor interfaz (el resto de los actores se modelarán como actores de soporte cuando se especifique el evento).

Heurística 3: Definición del Diagrama de Eventos Comunicacionales

Finalmente se establece un orden de dependencias entre los eventos detectados y se define el CED relativo al proceso detectado. La relación de dependencia se basa en los datos que cada evento necesita. Según [15], los recursos de una tarea deben estar provistos por otro, esto significa que si como resultado de una tarea se produjo un determinado recurso y otra tarea la consume, hay un claro nivel de precedencias entre am-

bas. Es esta dependencia de información la que marcará las dependencias de precedencia entre los eventos. De esta forma los CED surgen como consecuencia de querer cumplir los objetivos de la organización que se fueron refinando hasta llegar a objetivos operacionales.

3.4.1 Modelado de Procesos aplicado al Caso de Estudio

Continuando con el ejemplo, la tarea "Atender el pedido" cumple con el criterio unidad: Existe una actor que desencadena la acción (criterio de unidad de trigger) que es el Encargado de sucursal, se crea nueva información para el sistema, a partir de un pedido de productos, ya que se va a crear una orden que incluye el pedido, el cliente y un número de orden (unidad de comunicación), por último esta tarea involucra varias acciones sincrónicas (un pedido telefónico, vía mail o de manera personal y luego la creación de la orden para ese pedido). Se crear un Evento comunicacional "Atender pedido" cuyo actor primario es "Encargado de sucursal". En este caso como hay un único actor responsable (Vendedor), este será el actor interfaz del evento. (Cuando se especifique en detalle los eventos, este actor también será el actor soporte). En el caso de la tarea Verificar orden, cumple con el criterio (tiene un actor desencadenante, una reacción y un conjunto de acciones sincrónicas) y dado que existe dos posible reacciones, el evento va a tener dos variantes: "orden ingresa verificada" y "orden ingresada no verificada". En este último caso se finaliza el proceso (puede haber otras opciones). Luego de detectado los eventos, se procede a establecer el orden aplicando la heurística 3. En el ejemplo se puede observar que el evento "Atender pedido" necesita el pedido que es ingresado por el actor primario. Este evento es considerado el primero en el orden de precedencias del diagrama. El evento "Verificar orden" es desencadenado por el encargado con la orden ingresada, por lo cual se puede llevar a cabo una vez que el pedido haya sido almacenado como orden. En este caso, hay dos alternativas, una que da por finalizado el proceso por no poder cumplir la orden y otra que sigue el curso normal de la venta. Finalmente los eventos: Organizar transporte, Actualizar stock y Actualizar cuenta sucursal sólo tienen una relación de dependencia con el evento "Verificar orden". La Fig. 4 muestra parte del diagrama generado.

4 Discusión y análisis de la propuesta

Las estrategias de ingeniería de procesos orientadas a objetivos son en general el resultado de la integración o extensión de estrategias de modelado de objetivos y de estrategias de modelado de procesos de negocios [11,17]. En esta sección analizamos y fundamentamos nuestra propuesta en relación a las principales estrategias existentes.

El modelo de objetivos de nuestra propuesta es un grafo And/Or compatible con los modelos presentados en [5, 28,29]; y similares a [13, 14,15] aunque estos últimos no distinguen ramas And/Or. Asimismo, puede ser considerado como una simplificación del modelo SD de i* tomando a la Organización como un Actor, y modelando las intenciones a través de objetivos y tareas. Este modelo propuesto es un modelo general que cubre una especificación básica de objetivos. Dependiendo del nivel de complejidad de la organización, este modelo puede ser reemplazado o incrementado por mo-

delos más sofisticados que permitirán realizar un análisis exhaustivo de los objetivos. Por ejemplo, se pueden utilizar los modelos SD y SR de i*, si lo que se necesita es profundizar en el análisis de las intenciones y dependencias de los actores. Si se quiere hacer una análisis de alternativas viables se puede extender el grafo con aspectos no funcionales representados por *softgoal* (objetivos *soft*) como se propone en i* o KAOS. Sin embargo, no siempre es beneficioso contar con modelos más complejos y completos pero que requieren mas esfuerzo tanto en su construcción como en su comprensión [24]. El análisis de requisitos es esencial, pero cuesta tiempo y dinero. Un proyecto de desarrollo obtiene el mejor valor para su inversión cuando se aplica el suficiente y justo esfuerzo en cada elemento de requisito [2]. Asimismo, nuestra propuesta por ahora sólo contempla la dimensión funcional de los objetivos, concentrándose en aquellos que serán utilizados par la definición de los procesos. Sin embargo existen diferentes clasificaciones y taxonomías [3, 29].

El último nivel de nuestro modelo representa tareas. Este concepto está presente en varias estrategias: en [15] se crea una tabla de entradas y salidas para cada una de las tareas detectadas en el grafo. En [16] el modelo i* es extendido con notaciones de flujo de control y de datos. En el modelo SR de i* se detalla una tarea con links la dependencias y descomposiciones que ésta posee [10]. En KAOS, la descomposición de los objetivos llega hasta lograr un objetivo operacionable que se asigna a un agente. En nuestra propuesta, se crea un listado de especificación de tareas, permitiendo que ambos modelos (objetivos y tareas) estén relacionados entre si pero a la vez sean sencillos y fáciles de comprender.

En [13, 14, 15, 16, 26,27] cada modelo de objetivos representa a un proceso en particular, no teniendo un modelo de objetivos de la organización. Los nodos de bajo nivel del modelo serán las tareas de ese proceso. La propuesta de [5] modela los objetivos generales y propone heurísticas para detectar los procesos. Nuestra propuesta sigue este último lineamiento partiendo de modelos generales y proveyendo de una estrategia de derivación de procesos. En cuanto a la definición de procesos a partir de objetivos, se ha detectado que la mayoría de de los métodos que aplican estrategias orientadas a objetivos en el modelado de procesos generalmente carecen de un mapeo sistemático entre los conceptos de objetivos y los de procesos y no describen detalladamente las transformaciones entre ellos [4]. Todas las estrategias mencionadas en el artículo comienzan el modelado de objetivos desde un alto nivel de abstracción hasta llegar a objetivos operacionables o tareas que pueden ser asignadas a roles, proveyendo racionalidad de porqué el trabajo debe ser realizado y cómo se termina llevando a cabo mediante los procesos. Nuestra propuesta sigue esta línea. En [16] el modelo i* que representa a un proceso ya establecido es extendido con notaciones de flujo de control y de datos para luego derivar las componentes de ese proceso en BPEL [22]. Al modelarlo con BPEL, no se necesita la asignación tarea /actor. En nuestra propuesta, al derivar a un CED, se concreta una asignación de actores a los eventos. En la propuesta de [5] se propone una guía sistemática para derivar un modelo de procesos definidos en BPMN [21] a partir de modelos de objetivos, proponiendo como un proceso de negocios a un subárbol que contenga solo tareas. No se define cómo es la asignación de las tareas a los pools del proceso, los cuales se detectan a partir de las entidades de un dominio de interés asociado al modelo de objetivos. En nuestra propuesta

cada tarea es especificada y detallada y esta información sirve para detectar los actores responsables de cada evento comunicacional.

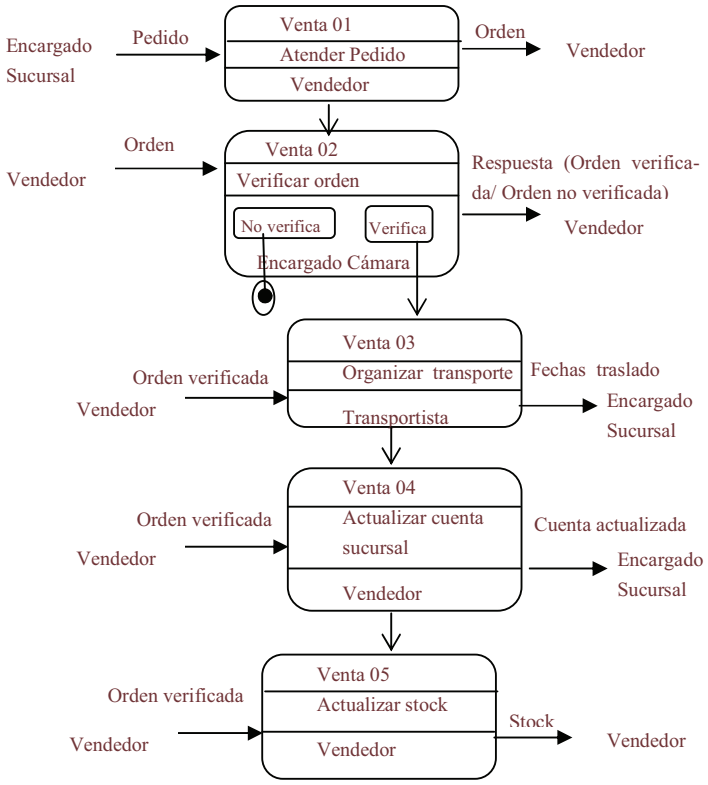


Fig. 4. CED "Atender Ventas"

5 Conclusiones y futuros trabajos

En un contexto de desarrollo To_Be, donde no existe un sistema previo donde poder realizar Análisis comunicacional, se intenta disponer de una guía para derivar sistemáticamente procesos de negocio a partir de un modelo de objetivos. El modelo de objetivos comienza con los objetivos generales de la organización que luego son refinados hasta alcanzar objetivos operacionales. A partir de ese modelo, se proponen heurísticas para definir los procesos que satisfagan los objetivos. Los diagramas resultantes son considerados un punto inicial y necesariamente deben ser refinados por los analistas para poder definir procesos con calidad, alineados a los objetivos. Este trabajo no pretende definir un nuevo modelo de objetivos sino adaptar existentes, proponiendo un modelo simple que puede ser extendido según la necesidad o la complejidad del con-

texto. Se pretende brindar un punto inicial a CA en el contexto de un desarrollo To_Be que permita detectar y especificar los procesos a partir de los objetivos.

El desarrollo de este trabajo está en su fase inicial. Sería razonable estudiar la posibilidad de aplicar el criterio de unidad comunicacional en el momento que se están identificando tareas. En esta primera versión, las tareas surgen como resultado del refinamiento de un modelo de objetivos usando alguna estrategia ya existente, y luego se aplica el criterio de unidad a la hora de definir los eventos. Se debe estudiar que ocurre cuando una tarea tiene más de un padre en el grafo (tareas soporte). Seria interesante contar con guías para definir procesos de soporte a los procesos del negocio. También se debe estudiar si existe un orden de precedencia entre procesos, o entre eventos de un proceso hacia eventos de otros procesos. Esta precedencia surgirá de la información que está descripta en las tareas que les dieron origen. Finalmente, para integrar el modelo de objetivos con CA en un contexto MDA, las heurísticas evolucionarán a reglas de transformación entre un modelo de objetivos y los modelos de CA, para lo cual se deberán extender los metamodelos existentes de CA.

6 Referencias

1. Alexander, I., Beus-Dukic, L. Discovering Requirements: How to Specify Products and Services. Wiley, Chichester. (2009)
2. Alexander I. "GORE, SORE, or What?", IEEE Software, pp.8-10. (2011)
3. Cardoso E., Guizzardi R. and Almeida J.P . Aligning Objectives and Business Process Models: A Case Study in the Health Care Industry. International Journal of Business Process Integration and Management, 5(2): 144-158(2011)
4. Decreus K, Snoeck M, Poels G. Practical Challenges for Methods Transforming i* Goal Models into Business Process Models. 17th IEEE International Requirements Engineering Conference. Pp15-23 (2009)
5. Decreus K, Poels G . A Goal-Oriented Requirements Engineering Method for Business Processes. CAiSE Forum: 29-43 (2010)
6. España, S., A. González and Ó. Pastor. Communication Analysis: a requirements engineering method for information systems. 21st International Conference on Advanced Information Systems (CAiSE9). The Netherlands, Springer LNCS 5565: 530-545. (2009)
7. España Sergio, Arturo González, Óscar Pastor, Marcela Ruiz, Integration of Communication Analysis and the OO Method: Manual derivation of the Conceptual Model. The SuperStationery Co. lab demo Technical Report PROS-TR-2011-01(2011)
8. España Sergio. Methodological integration of Communication Analisys Into a Model Driven software development framework. Tesis doctoral. Universidad politécnica de Valencia. www.dsic.upv.es/docs/bib-dig/tesis/etd-02012012-24944/Espana%20PhD%20draft%20v1.03lo.pdf (2011)
9. González, A., S. España and Ó. Pastor. Unity criteria for Business Process Modelling: A theoretical argumentation for a Software Engineering recurrent problem. IEEE Third International Conference on Research Challenges in Information Science. 173-182. (2009)
10. Giorgini P . Tropos: basics. http://www.troposproject.org/files/8-Tropos-Basics.pdf (2009)
11. Guizzardi, R. S. S., G. Guizzardi, J. P. A. Almeida and E. C. S. Cardoso. Bridging the gap between goals, agents and business processes. 4th International i* Workshop. J. Castro, X. Franch, J. Mylopoulos and E. Yu. Hammamet, Tunisia, (2010)
12. Gordijn Jaap, Akkermans Hans Value Based Requirements Engineering:Exploring Innovative e -Commerce Ideas. Requir. Eng. 8(2): 114-134 (2003)

13. Kavakli, E., Loucopoulos, P. Experiences With Goal-Oriented Modeling of Organizational Change. IEEE Transactions on Systems, Man, and Cybernetics. Applications and Reviews 36(2): 221-235. (2006)
14. Kavakli, E., Loucopoulos, P. Goal-Driven Business Process Analysis Application in Electricity Deregulation. Inf. Syst. 24(3): 187-207 (1999)
15. Kueng, P. and P. Kawalek. Goal□based business process models: creation and evaluation. Business Process Management Journal 3(1): 17-38. (1997)
16. Lapouchnian A, Yu Y., and Mylopoulos J. Requirements-Driven Design and Configuration. Management of Business Processes . G. Alonso, P. Dadam, and M. Rosemann (Eds.): BPM 2007, LNCS 4714, pp. 246–261, Springer-Verlag, 2007
17. Neiger D, Churilov L . Goal-Oriented Business Process Engineering Revisited: a Unifying Perspective. Proceedings of the 1st International Workshop on Computer Supported Activity Coordination, CSAC INSTICC PRESS. Pp 149-163 (2004)
18. Oshiro K., Watahiki K., Saeki M. Integrating an Idea Generation Method into a Goal-Oriented Analysis Method for Requirements Elicitation. APSEC: 113-121 (2005)
19. Object Management Group/ Model Driven Architecture (MDA). www.omg.org/mda/ Acceso Noviembre 2012
20. Object Management Group/ Business Motivation Model, version 1.1 http://www.omg.org/spec/BMM/1.1/ Acceso Noviembre 2012
21. Object Management group/Business Process Modeling Notation http://www.omg.org/bpmn/. Acceso Noviembre 2012
22. OASIS: Web Services Business Process Execution Language Version 2.0 www.oasis-open.org/committees/documents.php?wg_abbrev=wsbpel . Acceso Noviembre 2012
23. Pastor O. and J. C. Molina, Model-Driven Architecture in practice:a software production environment based on conceptual modeling. New York: Springer (2007)
24. Pohl, K. Requirements Engineering: Fundamentals, Principles, and Techniques. Springer, Heidelberg. (2010)
25. Regev, G., Wegmann, A. Where do Goals Come from: the Uncerlying Principles of Goal-Oriented Requirements Engineering. In: 13th IEEE International Conference on Requirements Engineering (RE'05), pp 353-362. (2005)
26. Rolland C., Nurcan S. Business Process Lines to deal with the Variability Proceedings of the 43rd Hawaii International Conference on System Sciences (2010)
27. Rolland, C., Salinesi, C. Modeling Goals and Reasoning with Them. Engineering and Managing Software Requirements (EMSR), Aurum, Wohlin (ed) pp-189-217. Springer Verlag (2005)
28. Singh, S.N., Woo, C. A Methodology for Discovering Goals at Different Organizational Levels. In: 3rd International Workshop on Business/IT Alignment (2008)
29. van Lamsweerde, A. Goal-oriented requirements engineering: a guided tour. In: 5th IEEE International Symposium on Requirements Engineering (RE'01), pp 249-262. (2001)

Buenas prácticas en la especificación del dominio de una aplicación

Leandro Antonelli[1], Gustavo Rossi[1],
Julio Cesar Sampaio do Prado Leite[2], Alejandro Oliveros[3]

[1] Lifia, Fac. de Informática, UNLP, calle 50 esq 120, La Plata, Bs As, Argentina
{lanto, gustavo}@lifia.info.unlp.edu.ar
[2] Dep. Informática, PUC-Rio, Rua Marquês de Sâo Vicente 255, Gávea, RJ, Brasil
www.inf.puc-rio.br/~julio
[3] INTEC – UADE, Bs As, Argentina
aoliveros@uade.edu.ar

Abstract. La ingeniería de requerimientos es un área crítica en el desarrollo de software y particularmente lo es la especificación de requerimientos. Se pueden construir productos tecnológicamente adecuados pero si ellos no cumplen con los requerimientos carecen de utilidad. Sin embargo, no es una tarea sencilla el construir una especificación correcta que describa y logre transmitir el conocimiento, necesidades y deseos de los stakeholders. Por este motivo, es necesario contar con guías que faciliten la tarea de escribir especificaciones de requerimientos. En este artículo, se presentan una serie de guías que tienen por objetivo mejorar la descripción del Léxico Extendido del Lenguaje (LEL). La identificación de las guías se realizó durante un ejercicio de escritura del LEL en donde se recurrió a la literatura para identificar las guías a aplicar como buenas prácticas, y finalmente se realizó un experimento para verificar la interpretación de las especificaciones realizadas utilizando las buenas prácticas propuestas.

Keywords. Dominio de la aplicación, especificación de requerimientos, buenas prácticas, Léxico Extendido del Lenguaje

1 Introducción

La etapa de relevamiento de requerimientos es un área clave para el éxito de un proyecto. En particular, el construir una especificación de requerimientos de software (ERS) adecuada es esencial, ya que constituyen un medio de comunicación entre los miembros del equipo de desarrollo con el resto de los stakeholders, y es necesario que el equipo de desarrollo posea los requerimientos correctos para construir el producto.

El problema que se presenta al construir el documento de especificación de requerimientos es que los stakeholders manejan un lenguaje distinto del lenguaje del equipo de desarrollo. Los stakeholders manejan el lenguaje natural, mientras que los miembros del equipo de desarrollo necesitan descripciones más precisas y formales. Es muy difícil lograr especificaciones que sean adecuadas y útiles para las dos partes,

porque el lenguaje natural carece de la precisión y formalidad necesaria, mientras que los usuarios no están familiarizados con los lenguajes formales.

El Léxico Extendido del Lenguaje, (Language Extended Lexicon, LEL) es un glosario que permite describir el lenguaje del dominio de la aplicación utilizando el lenguaje natural. El LEL no brinda una descripción de requerimientos, sino que sólo permite describir el lenguaje del dominio, a pesar de que hay trabajos que muestran cómo identificar requerimientos a partir del LEL [1].

El LEL es una herramienta muy conveniente para expertos sin habilidades técnicas, sin embargo, las personas con tales habilidades pueden obtener un mayor beneficio con su uso. La conveniencia de la utilización de LEL proviene de 3 características significativas: es fácil de aprender, es fácil de usar y posee buena expresividad, como lo validan experiencias en dominios complejos. Gil et al. [13] indican que: “la experiencia de construir un LEL de una aplicación completamente desconocida para los ingenieros de requerimientos y con un lenguaje altamente complejo, puede ser considerada exitosa, desde el momento en que los usuarios fueron los que notaron que los ingenieros de requerimientos habían desarrollado un gran conocimiento sobre la aplicación”. Por su parte, Cysneiros et al. [8] indican que: “el uso del LEL fue muy bien aceptado y comprendido por los stakeholders. Dado que los stakeholders no eran expertos en los dominios tan complejos y específicos en los que trabajaron, los autores creen que el LEL pude ser adecuado para utilizarse en muchos dominios”.

Con el fin de obtener más beneficios del LEL, se realizó una experiencia en donde se involucró a 70 personas en un ejercicio de construcción y lectura de LEL con el objetivo de identificar "buenas practicas". Las mismas apuntan a la expresividad, simpleza y modificabilidad del LEL.

La experiencia concreta consistió en solicitarle a 50 alumnos de un curso de posgrado (con experiencia en diferentes roles en el desarrollo de software y variada antigüedad) que escriban un LEL. Se trabajó sobre un único dominio, un dominio administrativo. Durante la construcción, a los participantes se les presentaron inquietudes sobre cómo describir ciertas situaciones. Por lo cual, se estudió la literatura de requerimientos y se estableció una forma de describir esas situaciones.

Esta descripción adoptada fue validada en un pseudo experimento que consistió en presentar un LEL específico a un grupo de 20 personas y validar si los sujetos comprendían la misma idea que se quería transmitir. Luego de esta validación se identificaron las buenas prácticas de descripción del LEL que se describen en el presente artículo.

El resto del artículo está organizado de la siguiente manera. La sección 2 describe la esencia del Léxico Extendido del Lenguaje. La sección 3 describe el mecanismo con el cual se identificaron las buenas prácticas. La sección 4 describe las buenas prácticas. Finalmente, la sección 5 describe los trabajos relacionados.

2 Léxico Extendido del Lenguaje

El Léxico Extendido del Lenguaje (LEL) [18] es un glosario que tiene por finalidad describir el lenguaje del contexto de la aplicación. Su objetivo es el describir ciertas

palabras o frases peculiares a la aplicación y que su comprensión son vitales para poder comprender el contexto de la misma. LEL es anclado en una idea bien simple "entender el lenguaje del problema sin preocuparse por entender el problema". De esta forma, luego de comprender el lenguaje, el analista podrá escribir requerimientos utilizando como base el conocimiento adquirido a través del lenguaje capturado.

El LEL es un glosario en el cual se definen símbolos (términos o frases). A diferencia del diccionario tradicional que posee sólo un tipo de definición por término (puede haber muchas acepciones, sin embargo, todas son significados del término que se está describiendo). En el LEL cada símbolo se define a través de dos atributos: la noción (notion) y los impactos (behavioural responses). La noción describe la denotación, es decir, describe las características intrínsecas y substanciales del símbolo. Por su parte, los impactos describen la connotación, es decir, un valor secundario que adopta por asociación con un significado estricto.

Existen dos principios que se deben seguir al describir símbolos: el principio de circularidad (también llamado principio de cierre o clausura) y el principio de vocabulario mínimo. El principio de circularidad establece que durante la descripción de los símbolos se debe maximizar el uso de otros símbolos descriptos en el LEL. Por su parte, el principio de vocabulario mínimo complementa el principio de circularidad y establece que en las descripciones se debe minimizar el uso de símbolos externos al LEL y los que se utilicen deben tener una definición clara, unívoca y no ambigua.

Ambos principios son vitalmente importantes para obtener un LEL autocontenido y altamente conectado. Estas características redundan en beneficios para comprender el lenguaje de la aplicación, y también hacen que el LEL pueda ser visto como un grafo, el cual, al contener nodos con información textual determina que el mismo sea un hipertexto.

Los símbolos se deben categorizar en una de cuatro categorías básicas con el fin de especializar la descripción de los atributos. En principio hay 3 ontologías básicas que permiten modelar el mundo [4]. Ellas son: entidades, actividades y afirmaciones. Con estos 3 elementos se puede describir al mundo, puesto que permiten modelar las cosas, las actividades con las que interactúan las cosas y finalmente las afirmaciones o verdades que las cosas o actividades poseen o persiguen. Para el LEL se determinan cuatro categorías básicas que son una extensión de las tres ontologías. Las cuatro categorías básicas son: sujeto, objeto, verbo y estado. Tanto los sujetos como los objetos son una especialización de las entidades. Luego, las actividades se corresponden con los verbos. Y finalmente las afirmaciones se corresponden con estados. Los sujetos se corresponden con elementos activos dentro del contexto de la aplicación, mientras que los objetos se corresponden con elementos pasivos. Por su parte, los verbos son las acciones que realizan los sujetos utilizando los objetos. Finalmente, los estados representan las situaciones en las que se pueden encontrar los sujetos o los objetos previo y luego de realizar las acciones.

3 Generación de buenas prácticas

La consolidación de las buenas prácticas, consistió en dos etapas: identificación y validación. Para ello, se llevó a cabo el siguiente proceso. Primero se convocó a un

grupo de 50 personas para escriban un LEL sobre un mismo dominio de aplicación concreto a la vez que identificaban puntos débiles de las descripciones. Luego se analizó la literatura para definir criterios a adoptar con el fin de mejorar las descripciones. Finalmente se validaron las guías definidas. Las primeras dos actividades se corresponden con la etapa de identificación, mientras que la última actividad se corresponde con la etapa de validación.

3.1 Etapa 1: confección del LEL

Se les solicitó a 50 personas que escriban un LEL sobre un dominio de aplicación administrativo. La experiencia se desarrolló con alumnos de posgrado de Argentina de dos cohortes consecutivas: 2010 y 2011. Los participantes eran de distintas ciudades de la Argentina, e incluso algunos de ellos eran de Colombia y Brasil. Tenían una experiencia muy variada, principalmente en desarrollo, desempeñándose en diferentes roles y con variada antigüedad. Algunos de ellos también tenían experiencia en docencia e investigación.

A los alumnos se les entregó una especificación y debían construir el LEL en 3 meses. Los trabajos eran supervisados periódicamente y a los participantes se les presentaron inquietudes sobre como describir ciertas situaciones. Estas inquietudes fueron las que motivaron que se estudie la literatura de requerimientos y se acordara una forma de describir esas situaciones, determinando así las buenas prácticas descriptas en este artículo.

3.2 Etapa 2: definición de buenas prácticas

Se estudiaron dos fuentes de literatura. Por un lado la bibliografía de especificación de requerimientos y por otro lado la bibliografía de análisis y diseño orientado a objetos.

La bibliografía de requerimientos se utilizó para ajustar las descripciones textuales del LEL, ya que se aprovechó las sugerencias de cómo redactar requerimientos textuales. Sin embargo, el LEL tiene una estructura que permite hacer una analogía entre las descripciones del LEL y la de un diseño orientado a objetos [19]. Es por ello, que también se utilizó la literatura de objetos para definir ciertas prácticas.

3.3 Etapa 3: validación de buenas prácticas

Con el fin de validar las buenas prácticas definidas se realizó un pseudo experimento en el que participaron 20 personas. El mismo consistió en presentar a los participantes la definición de algunos símbolos de un LEL de un dominio específico junto con un cuestionario sobre ciertos elementos del dominio para el cual se escribió el LEL.

Los participantes debían responder a cada pregunta basándose en la definición de los símbolos. Las preguntas se enfocaban en aspectos del dominio que estaban descriptos haciendo uso de las reglas propuestas. Por lo tanto, a partir de las respuestas de los participantes se pudo determinar si las reglas fueron adecuadas o no, ya que se verificaba si los participantes comprendían la misma idea que el LEL con el uso de las

buenas prácticas se quería transmitir. Luego de esta validación, se realizaron ajustes y se estableció la versión final de buenas prácticas.

4 Buenas prácticas

Esta sección describe cada una de las 10 buenas prácticas que fueron identificadas. Las mismas se describe con el siguiente formato: (i) nombre de la guía, (ii) descripción de la misma, (iii) ejemplo y (iv) justificación. Cabe señalar que si bien las guías fueron validadas a través de un pseudo experimento, la justificación descripta en (iv) se corresponde con el análisis de la literatura que se realizó.

4.1 Guía 1: información a incluir en la noción y en los impactos

Descripción: la descripción de la noción y los impactos se debe ajustar a la categoría del símbolo que se debe describir. Para los *sujetos*, la noción debe describir las características o condiciones que debe satisfacer para poder considerarse como tal. Por su parte, los impactos, deben describir las acciones que realizan. Para los *objetos*, la noción debe describir los atributos o características constitutivas del *objeto*, mientras que los impactos deben describir las acciones que se realizan sobre ellos. Para los *verbos*, la noción debe describir el objetivo o fin que persiguen, mientras que los impactos deben describir los pasos necesarios para cumplir con la acción. Finalmente, la noción de los estados debe describir la situación que representa, mientras que los impactos deben describir las acciones que se pueden realizar a partir de ese estado y el nuevo estado que se puede acceder. La siguiente tabla resume la información.

Tabla 1. Descripción de cada categoría de símbolo.

Categoría	Noción	Impactos
Sujeto	Características y condiciones que el sujeto debe satisfacer	Acciones que el sujeto realiza
Objeto	Atributos o características constitutivas del objeto	Acciones que se realizan sobre los objetos
Verbo	Objetivo o fin que el verbo persigue	Pasos necesarios para cumplir con el objetivo
Estado	Situación que el verbo representa	Acciones posibles de realizar desde el estado y nuevo estado al que se arriba.

Ejemplo: las siguientes figuras muestran la descripción de un símbolo de cada categoría.

Sujeto: cliente
Noción
Persona que opera con una cuenta.
Impactos
El cliente abre una cuenta.
El cliente deposita dinero en su cuenta.
El cliente extrae dinero de su cuenta.
El cliente consulta su balance de cuenta.
El cliente cierra una cuenta.

Fig. 1. Descripción del símbolo cliente.

Objeto: cuenta
Noción
La cuenta posee un balance.
Impactos
El cliente abre una cuenta.
El cliente deposita dinero en su cuenta.
El cliente extrae dinero de su cuenta.
El cliente consulta su balance de cuenta.
El cliente cierra una cuenta.
El banco audita cada cuenta.
El cliente cierra una cuenta.

Fig. 2. Descripción del símbolo cuenta.

Verbo: extraer
Noción
Acción de tomar dinero de su cuenta.
Impactos
El banco revisa que la cuenta tenga suficiente dinero para realizar la extracción.
El banco revisa que el propietario de la cuenta no haya extraído más veces del límite permitido.
El banco revisa que el propietario de la cuenta no posea deudas en su tarjeta de crédito.
El banco reduce el balance de la cuenta.

Fig. 3. Descripción del símbolo extraer.

Estado: activada
Noción
Situación donde el cliente puede operar con su cuenta.
Impactos
El cliente cierra la cuenta y el cliente es propietario de una cuenta cerrada.

Fig. 4. Descripción del símbolo activada.

Justificación: la descripción de cada categoría está basada en el análisis y diseño orientado a objetos propuesto por Wirfs-Brock [27]. Las categorías sujeto y objeto, se corresponden con objetos (clases) es por ello que la noción caracteriza a estos elementos, mientras que los impactos describen las acciones vinculadas con cada uno de ellos. Por su parte, los verbos se corresponden con el comportamiento de los objetos (métodos), es por ello que la noción describe conceptualmente la acción, mientras que los impactos detallan la misma. Finalmente, la categoría estado, guarda estrecha relación con el patrón estado [12], por lo cual, su descripción tiene como finalidad describir una máquina de estados.

4.2 Guía: formato que deben cumplir cada una de las expresiones de los impactos de los sujetos, objetos y verbos

Descripción: las acciones descriptas en los impactos de sujetos, objetos y verbos deben cumplir la estructura: *sujeto + verbo + objeto*. Es importante destacar que el objeto al final de la oración queda condicionado por el verbo. Es decir, si el verbo es transitivo, se debe indicar un objeto. Caso contrario, no es necesario.

Ejemplo: las Fig. 1y Fig. 2 muestran en los impactos la estructura *sujeto + verbo + objeto* para símbolos sujeto y objeto. Dado que los verbos son transitivos, se incluye la descripción de los *objetos*. Por su parte, la Fig. 3 ilustra la descripción de un verbo. Cabe señalar que los verbos utilizados no están definidos en LEL, ya que son verbos que poseen el significado trivial.

Justificación: los impactos describen acciones, que pueden considerarse como funciones o requerimientos de un sistema. El estándar IEEE 830 [15] determina que los requerimientos deben describirse de la forma: "El sistema debe *acción*". Luego, Kovitz [16] especializa esta descripción, incorporando el rol: "El *rol* debe *acción*". La guía presentada incorpora además el objeto.

4.3 Guía: formato que deben cumplir cada una de las expresiones de los impactos de los estados.

Descripción: las acciones descriptas en los impactos de los estados deben cumplir la estructura: *sujeto + verbo + objeto + nuevo estado*. Es importante destacar que el objeto queda condicionado por el verbo, ya que si el verbo no es transitivo el objeto no se debe indicar.

Ejemplo: la Fig. 4 muestra los impactos de un estado conforme al formato *sujeto + verbo + objeto + nuevo estado.* El único elemento a considerar es que se incluye dos veces el símbolo *cuenta* y el símbolo *cliente*, pero ello no desvirtúa que la descripción se ajuste al formato indicado, ya que al final de la expresión se referencia al nuevo estado.

Justificación: Chelimsky [6] propone especificar comportamiento en desarrollo ágil con el formato: *given* (situación inicial) *when* (acción) *then* (situación final). Básica-

mente describe una máquina de estados con la transición necesaria para pasar de un estado a otro. Para la regla propuesta, el símbolo estado que se está describiendo es la situación inicial, luego, las transiciones (acciones) se describen con: *sujeto + verbo + objeto*, mientras que el estado final con: *nuevo estado*.

4.4 Guía: simplicidad en los impactos.

Descripción: los impactos deben ajustarse al formato indicado en las guías 4.2 y 4.3, y si fuera necesario agregar información adicional, debe definirse alguno de los símbolos involucrados y agregar la información adicional en la descripción de ese símbolo.

Ejemplo: la Fig. 5 muestra la descripción del símbolo cliente que incluye en los impactos información de cómo realizar la acción de *abrir* una cuenta. La información adicional (*indicando nombre, apellido, etc…*) es propia de la acción *abrir*, por lo cual, es necesario definir el símbolo *abrir* con esa descripción, como se muestra en la Fig. 6.

Sujeto: cliente
Noción
…
Impactos
El cliente abre una cuenta indicando nombre, apellido, …

Fig. 5. Descripción del símbolo cliente con un impacto con demasiada información

Sujeto: cliente
Noción
…
Impactos
El cliente abre una cuenta

Sujeto: abrir
Noción
…
Impactos
El cliente indica nombre, apellido, ...

Fig. 6. Descripción del símbolo cliente con información factorizada

Justificación: el estándar IEEE 830 [15] establece que una especificación de requerimientos debe ser modificable, y para ello la modularidad es una herramienta indispensable. La misma estrategia se aplica al LEL.

4.5 Guía: auto referencias explícitas en los impactos de los sujetos y objetos.

Descripción: en la descripción de los impactos de un sujeto y de un objeto, cuando se debe referenciar al sujeto y objeto, se lo debe hacer explícitamente y no utilizar un pronombre.

Ejemplo: las Fig. 1 y Fig. 2 muestran que los impactos poseen referencias al mismo símbolo que se esta describiendo. Por ejemplo, se indica "El cliente abre una cuenta.". A pesar de que es una descripción del cliente, al mismo se lo debe mencionar explícitamente en lugar de indicar "Él abre una cuenta" o simplemente "abre una cuenta."

Justificación: La ambigüedad es uno de los problemas que se intentan evitar en la descripción de requerimientos [16] [26], es por ello que se evita utilizar el pronombre.

4.6 Guía: no utilizar auto referencias en los impactos de verbos o en la noción de cualquier tipo de símbolo

Descripción: no se debe describir la noción de ninguna categoría de símbolos usando el mismo símbolo que se está describiendo. Tampoco se deben describir los impactos de los verbos usando el mismo verbo.

Ejemplo: en la Fig. 1 puede observarse que en la noción de cliente, se describe que es "persona que opera…" para evitar utilizar nuevamente el término *cliente*. Por otra parte, la Fig. 3 define el término *extraer*. En los impactos, no se utiliza el término *extraer*, sino que se indican las acciones necesarias para realizar la extracción, ya sean las verificaciones como la reducción concreta del saldo.

Justificación: no es correcto definir el símbolo en términos de sí mismo o tampoco es posible descomponer la tarea en términos de sí misma.

4.7 Guía: no se deben utilizar frases débiles

Descripción: no se deben utilizar frases débiles tales como: puede, podría, etc.

Ejemplo: la siguiente figura muestra un ejemplo de lo que no debe hacerse. Como impacto se incluye la expresión "El cliente puede abrir una cuenta". La frase incluye la expresión débil "puede" que debe evitarse.

Objeto: cuenta
Noción
…
Impactos
El cliente puede abrir una cuenta.
…

Fig. 7. Descripción parcial del símbolo cuenta con una frase débil.

Justificación: las frases débiles dan a lugar cierta subjetiva sobre la descripción, por lo cual, no deben ser utilizadas [24].

4.8 Guía: relación "es un" ("is a")

Descripción: dados dos símbolos en donde uno es una especialización del otro, no se debe repetir información en ambos. En la noción del más específico se debe colocar la expresión *"es un" + símbolo genérico*.

Ejemplo: la siguiente figura muestra el objeto *cuenta corriente*, que utiliza la relación "*es un"* en la descripción de la noción. Por otra parte, dado que la *cuenta corriente* es una *cuenta* y todos los impactos de *cuenta* aplican para *cuenta corriente*, *cuenta corriente* sólo define los impactos específicos (*El cliente deposita cheques*).

Objeto: cuenta
Noción
…
Impactos
…

Objeto: cuenta corriente
Noción
Es una cuenta.
Impactos
El cliente deposita cheques.

Fig. 8. Utilización de la relación "*es un"*.

Justificación: esta relación es utilizada en representación del conocimiento, modelos conceptuales y diseño y programación orientada a objetos [14]. En particular, en programación orientada a objetos existe el principio de sustitución denominado principio de sustitución Liskov [20].

4.9 Guía: relación "desempeña el rol"

Descripción: dados símbolos sujetos, en donde hay características que no son propias de los sujetos, sino del rol que desempeñan, se debe definir un símbolo con las características de los sujetos y la descripción propia del rol. Luego, en los sujetos que desempeñan ese rol, se deben enriquecer la noción con la expresión *"desempeña el rol" + rol*.

Ejemplo: en la siguiente figura puede observarse como el cliente puede desempeñar el rol de moroso.

Sujeto: cliente
Noción
El cliente puede desempeñar el rol de moroso
Impactos
El cliente abre una cuenta

Sujeto: moroso
Noción
…
Impactos
El moroso debe pagar deudas

Fig. 9. Utilización de la relación "*desempeña el rol"*.

Justificación: en diseño orientado a objetos existe el patrón *the role object pattern* que permite adaptar un objeto a diferentes necesidades de sus objetos clientes adosándole distintos roles [3].

4.10 Guía: relación "tiene un"

Descripción: dados dos símbolos de categoría objeto, en donde uno es una parte del otro, en la noción del objeto contenedor se debe indicar la expresión *"tiene un" + objeto contenido.*

Ejemplo: la siguiente figura muestra el objeto *cuenta*, que utiliza la relación "*tiene* un" en la descripción de la noción, para indicar la composición con el símbolo objeto *balance.*

Objeto: cuenta
Noción
La cuenta tiene un balance.
Impactos
…

Objeto: balance
Noción
…
Impactos
…

Fig. 10. Utilización de la relación "*tiene un"*.

Justificación: esta relación es utilizada en diseño de base de datos como así también en, modelos conceptuales, y diseño y programación orientada a objetos [14].

5 Trabajos relacionados

Rost [25] plantea la dificultad de escribir una especificación que se ajuste a los distintos estándares (por ejemplo [15]). Es por ello que hay varios trabajos que se ocupan de describir las mejores prácticas en los procesos de la ingeniería de requerimientos con el fin de lograr un buen producto final aplicando tales procesos [2] [23] [17]. Young et al. [28] por su parte plantean un marco de cómo mejorar los procesos que se realizan en lugar de proponer nuevos procesos que podrían resultar difíciles de implementar.

Además de buenas prácticas y mejoras en los procesos, existen trabajos que proponen guías identificadas a partir de la experiencia y otros trabajos realizan experimentos con el fin de identificar las buenas prácticas. Forsgren et al. [11] proponen guías para especificar requerimientos en sistemas de control industrial. Si bien el trabajo se ocupa de un dominio específico, en el mismo se discute el balance entre requerimientos detallados versus requerimientos generales, como así también de requerimientos funcionales versus requerimientos que describen la implementación técnica. Por su parte, Firesmith [10] provee un conjunto muy amplio de guías para utilizar con el modelado de Casos de Uso sobre cuestiones a tener en cuenta en los procesos de construcción y sobre la especificación concreta.

En relación a trabajos que realizan estudios empíricos Carew et al. [5] presentan una comparación sobre la comprensibilidad de dos especificaciones: una informal y otra informal (o semi formal). Los resultados muestran que la especificación informal fue mejor comprendida que la especificación formal. Por su parte, España et al. [9] evaluaron la calidad de especificaciones funcionales, enfocándose en completitud y granularidad. Realizaron un experimento de laboratorio con alumnos de posgrado para comparar Casos de Uso y Análisis de comunicación. Los resultados muestran que se obtuvo mayor calidad (en términos de completitud y granularidad) cuando se siguieron guías de análisis de comunicación.

La importancia de la comunicación también queda claro en el trabajo de Marnewick et al. [22]. Ellos recabaron evidencia empírica durante 5 años de dos casos de estudios. Se enfocaron en los procesos de ingeniería de requerimientos para obtener una comprensión en profundidad sobre que procesos permiten obtener requerimientos de buena o mala calidad. Los resultados sugieren que los factores humanos durante los procesos de comunicación juegan un rol significativo para lograr la calidad de los requerimientos.

Por otra parte, Cox et al. [7] muestran la importancia de contar con glosarios. En su trabajo se muestra el estudio de las prácticas de las 7 áreas claves de la guía de buenas prácticas de la ingeniería de requerimientos a través de entrevistas en profundidad en 10 organizaciones de desarrollo de software de Australia. De entre todas las conclusiones a las que se llegan, la importancia de contar con un glosario es una de ella.

Finalmente, Maiden [21] expone que existen pocos estudios sobre las prácticas actuales en ingeniería de requerimientos y que existe poco conocimiento sobre como se desarrolla la disciplina. Por lo que propone estudiar los procesos cognitivos que se requieren para realizar buenos trabajos en requerimientos.

6 Conclusiones y trabajos futuros

Este artículo presentó una serie de guías a modo de buenas prácticas para construir el Léxico Extendido del Lenguaje. Estas buenas prácticas fueron identificadas en forma empírica con el apoyo de la literatura. Las buenas prácticas establecidas fueron validadas en un curso de posgrado. Producto de esa validación se realizaron ajustes y se estableció el conjunto definitivo que se presenta en este trabajo. El LEL ha demostrado ser útil en dominios complejos, en donde se notó como los involucrados lograron un conocimiento del dominio. Con estas buenas prácticas, se busca mejorar los resultados al utilizar el LEL, enfocando la precisión en las descripciones, la organización de la información para facilitar la construcción y la modificación. Existen trabajos que permiten obtener distintos productos a partir del LEL: ontologías, requerimientos y características transversales entre otros. Si bien se seguirá trabajando en el proceso de construcción del LEL y la expresividad del mismo a partir de las guías propuestas, otra línea de investigación consiste en estudiar como varía la calidad de los productos que se construyen a partir del LEL con la utilización de las guías propuestas.

References

1. Antonelli, L., Rossi G., Leite, J.C.S.P., Oliveros, O.: Deriving requirements specifications from the application domain language captured by Language Extended Lexicon. In proceedings of the Workshop in Requirements Engineering (WER), Buenos Aires, Argentina, April 24 – 27 (2012).
2. Haron, A., Harun, M., Naz'ri Mahrim, M., Sahibuddin, S., Zakaria, N. H., Abdul Rahman, N.: Understanding the Requirement Engineering for organization: The challenges, In Proceedings of the 8th International Conference on Computing Technology and Information Management (ICCM), Volume: 2, pp 561 – 567 (2012).
3. Bäumer, D., Riehle, D., Siberski, W., Wulf, M.: The Role Object Pattern In Proceedings of PLOP 97, Monticello, Illinois, USA (1997).
4. Breitman K.K., Leite J.C.S.P.: Ontology as a Requirements Engineering Product, In Proceedings of the 11th IEEE International Conference on Requirements Engineering (RE), IEEE Computer Society, Monterey Bay, California, USA, ISBN 0-7695-1980-6 (2003).
5. Carew, D., Exton, C., Buckley, J.: An Empirical Investigation of the Comprehensibility of Requirements Specifications, In Proc of the International Symposium on Empirical Software Engineering , IEEE (2005).
6. Chelimsky, D., Astels, D., Helmkamp, B., North, D., Dennis, Z., Hellesoy, A.: The RSpec Book: Behaviour Driven Development with Rspec, Cucumber, and Friends, Pragmatic Bookshelf, (2010).
7. Cox, K., Niazi, M., Verner, J.: Empirical study of Sommerville and Sawyer's requirements engineering practices, IET Software Volume: 3 , Issue: 5 pp 339 – 355 (2009).
8. Cysneiros L.M., Leite J.C.S.P.: Using the Language Extended Lexicon to Support Non-Functional Requirements Elicitation, in proceedings of the Workshops de Engenharia de Requisitos, Wer'01, Buenos Aires, Argentina (2001).
9. España, S., Condori-Fernandez, N., González, A., Pastor, O.: Evaluating the Completeness and Granularity of Functional Requirements Specifications: A Controlled Experiment, In Proceedings of the 17th IEEE International Requirements Engineering Conference (2009).

10. Firesmith, D. G.: Use Case Modeling Guidelines, Technology of Object-Oriented Languages and Systems - TOOLS, pp. 184-193 (1999).
11. Forsgren, P., Rahkonen, T.: Specification of Customer and User Requirements in Industrial Control System Procurement Projects, In Proceedings of the Second IEEE International Symposium on Requirements Engineering, IEEE, pp 81-88 (1995).
12. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns CD: Elements of Reusable Object-Oriented Software, Addison-Wesley Professional (1994).
13. Gil D., Figueroa D. A., Oliveros A.: Producción del LEL en un Dominio Técnico.Informe de un caso, In proceedings of the Workshops de Engenharia de Requisitos, Wer'00, Rio de Janeiro, Brazil (2000).
14. Gunter, K. A., Mitchell, J. C.: Theoretical Aspects of Object-Oriented Programming, The MIT Press (1994).
15. IEEE Recommended Practice for Software Requirements Specifications, IEEE Computer Society, IEEE Std 830-1998 (1998)
16. Kovitz, B. L.: Practical software requirements: A manual of content and style, ISBN 1884777597, Manning, Greenwich (1999).
17. Laplante, P.A., Neill, C.J. , Jacobs, C.: in Proceedings of the Software Engineering Workshop, 27th Annual NASA Goddard/IEEE, pp 121 – 128 (2002).
18. Leite J.C.S.P., Franco A.P.M.: A Strategy for Conceptual Model Acquisition, In Proceedings of the First IEEE International Symposium on Requirements Engineering, San Diego, California, IEEE Computer Society Press, pp 243-246 (1993).
19. Leonardi, C., Leite J.C., Rossi G.: Una Estrategia de Modelado Conceptual de Objetos basada en Modelos de Requisitos en Lenguaje Natural, Tesis de maestría, http://www-di.inf.puc-rio.br/~julio/teses.htm, Facultad de informática, Universidad Nacional de La Plata, Argentina (2001).
20. Liskov, B., Wing, J.: A behavioral notion of subtyping, ACM Transactions on Programming Languages and Systems (TOPLAS), Volume 16, Issue 6, pp. 1811 – 1841 (1994).
21. Maiden, N.: Exactly How Are Requirements Written?, IEEE Software Volume: 29 , Issue: 1, pp 26 – 27 (2012).
22. Marnewick, A., Pretorius, J. H., Pretorius, L.: A perspective on human factors contributing to quality requirements: A cross-case analysis, In Proceedings of the International Conference on Industrial Engineering and Engineering Management (IEEM), IEEE, pp. 389 – 393 (2011).
23. Plotka, M.A., Syty, P.: Good practices in requirements, project and risk management in educational IT projects, In Proceedings of the Federated Conference on Computer Science and Information Systems (FedCSIS), pp 1017 – 1021 (2012).
24. Rosenberg, L.: Methodology for Writing High Quality Requirement Specifications and for Evaluating Existing Ones, Software Assurance Technology Center, NASA Goddard Space Flight Center Greenbelt, MD, September 24 (1998)
25. Rost, J. A.: "Best Practices" Requirements Documents a Myth?, IEEE Software, Volume: 23 , Issue: 3 pp 104 (2006).
26. Wiegers, K.: Software requirements, Miscrosoft Press (1999).
27. Wirfs-Brock, R., Wilkerson, B., Wiener, L.: Designing Object-Oriented Software, Prentice Hall (1990).
28. Young, R.: Effective Requirements Practices, Addison-Wesley, Boston (2001).

Uso de Sinônimos na Identificação de Atributos de Transparência

Joás Weslei Baía, José Luis Braga

Universidade Federal de Viçosa
Campus UFV, 36570-000 Viçosa, MG, Brasil.
joas.baia@gmail.com; zeluis@dpi.ufv.br;

Resumo. A transparência é um princípio democrático que permite aos cidadãos buscarem informações sobre fatos e processos. Nesse contexto, ocorre o aumento da demanda por transparência nas relações humanas. A transparência de *software* é um requisito não funcional que engenheiros de *software* precisarão demonstrar à medida que a sociedade exigir transparência em suas relações, pois essas relações são automatizadas pelos sistemas de *software*. Diante da importância de demonstrar a transparência nos processos de informação, propõe-se nesse trabalho verificar a presença de atributos de transparência em modelos de requisitos de *software* representados com o *Framework* iStar, utilizando os termos que representam os requisitos e também seus sinônimos mais conhecidos. O modelo de requisitos é analisado utilizando uma base de dados contendo sinônimos dos atributos de transparência, em adição aos conceitos do iStar para identificar esses atributos.

Palavras-chave: Framework iStar, transparência de *software*, verificação de modelos de requisitos.

1 Introdução

A transparência é um princípio democrático onde cidadãos desejam obter informações sobre fatos e processos. As organizações necessitam obter informações sobre o panorama mundial. Os investidores precisam de informações sobre oportunidades e riscos de novos mercados para decidirem sobre oportunidades de investimento. As organizações de proteção à saúde necessitam de fontes de informações sobre epidemias, e outras questões relativas ao bem estar dos seres humanos. Organismos internacionais necessitam de informações válidas sobre as intenções e estratégias políticas entre os países. Nesse contexto, ocorre o aumento da demanda por transparência nas relações humanas [1].

A transparência de software é um requisito não funcional que engenheiros de software precisarão demonstrar cada vez mais à medida que a sociedade exigir transparência nas relações com seus representantes, nas relações comerciais, sociais, enfim, nas relações humanas, pois essas relações são automatizadas pelos sistemas de

software [2]. Considerando os processos organizacionais a transparência é conceituada como a existência de processos que permitem aos indivíduos obterem informações sobre a organização através do acesso, uso, apresentação, entendimento e auditabilidade [3].

Diante da importância de demonstrar a transparência nos processos de informação, propõe-se nesse trabalho verificar a presença de requisitos de transparência a partir de modelos de requisitos de *software* representados com o Framework iStar [4]. Os requisitos são analisados através de uma base de sinônimos de atributos de transparência, que serão indicativos indiretos da presença dos atributos.

A Figura 1 apresenta o modelo de razões estratégicas do cenário *adicionar foto* em um dispositivo móvel, adaptado de [11]. O usuário deseja manipular mídias, nesse sentido ao manipular uma foto ele informa o local dessa foto, atribui seu nome e indica o álbum ao qual ela pertencerá. O sistema do dispositivo é o ator responsável pelo gerenciamento das mídias manipuladas pelo usuário. Para alcançar esse objetivo o sistema inicializa o aplicativo, disponibiliza a lista de álbuns, salva e armazena a fotos. Nessa especificação foi acrescentado ao modelo original o requisito *Ease* para ilustrar o uso de sinônimos na identificação de atributos de transparência. A tarefa *Save Automatically* contribui parcialmente para que o objetivo *Ease,* facilidade para manipular mídias, seja alcançado. Na seção 4 discutiremos o papel de sinônimos na identificação de atributos de transparência.

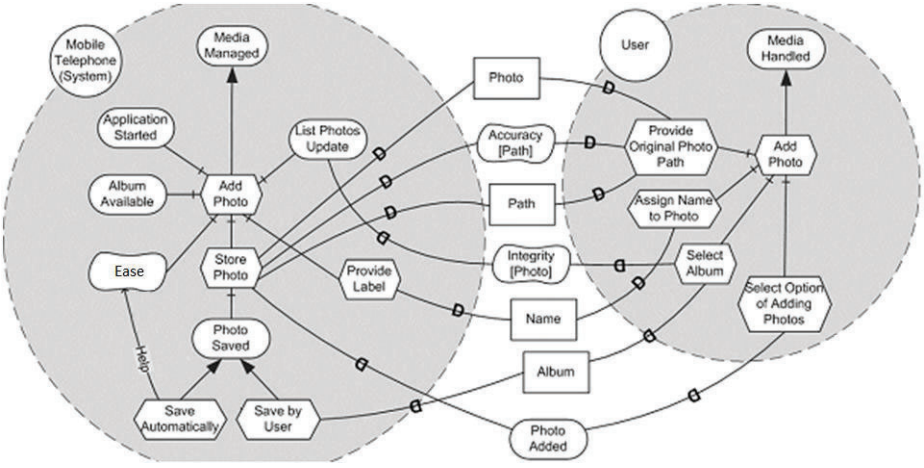


Fig. 1. Modelo de Razões Estratégicas da especificação de requsito (Mobile Telephone System). Adaptado de [11].

Nesse texto será apresentada a abordagem para analisar modelos de requisitos sob o ponto de vista da transparência. A estratégia apresentada para verificar a presença de atributos de transparência em modelos de requisitos apresentada nesse trabalho fornece:

- Uma base de dados de sinônimos dos atributos de transparência.
- Um mecanismo para transformar o modelo de requisitos representado com o Framework iStar em uma base de fatos (base de dados).

- Uma base de conhecimento para análise desses modelos de requisitos composta por regras de produção.
- A separação entre o mecanismo de controle (base de conhecimento) e a descrição do problema (especificação de requisitos) permite um sistema flexível e expansível para realizar a verificação de requisitos.

Este trabalho é um desdobramento do artigo [10], apresentado no WER2012 em Buenos Aires. A estrutura da solução baseada em Sistemas de Conhecimento está descrita naquele artigo, e será apenas referenciada neste artigo.

Esse texto está organizado em seis seções. Na segunda seção é apresentada a relação entre os atributos de transparência e as características do Framework iStar. A terceira seção apresenta a base de dados contendo os sinônimos dos atributos de transparência. A quarta seção discute a estratégia escolhida para a verificação da presença de atributos de transparência em especificações iStar. Na quinta seção discutimos os trabalhos correlacionados. Na sexta seção finalizamos com as conclusões.

2 A Transparência de Software e sua Relação com o Framework iStar

A transparência de *software* é um requisito não funcional, pois é caracterizada por atributos de qualidade, podendo ser julgada de forma diferente de acordo com a visão do indivíduo, dessa forma ela é considerada uma meta-flexível, ou um *softgoal*, ou seja, o *software* pode ser considerado transparente do ponto de vista de uma pessoa, enquanto pode não ser transparente ou ainda menos transparente para outra [5]. O *software* transparente deve possuir os atributos de *acessibilidade*, *usabilidade*, *informatividade*, *entendibilidade* e *auditabilidade*, pois nos processos organizacionais transparentes esses atributos devem estar presentes, e considerando que os sistemas de *software* automatizam tais processos, eles devem incorporá-los [6][7].

A Tabela 1 apresenta os atributos de transparência relacionados com as características do Framework iStar. Esse relacionamento entre os atributos que conceituam a transparência e o iStar é utilizado na implementação de regras de produção para verificar se uma especificação de requisitos é aderente aos conceitos da Transparência. A coluna Relação expressa a aderência das características do iStar identificados pelos algarismos de 1 a 9 na coluna ID, aos atributos de transparência. *Verificabilidade*, por exemplo, está relacionada com as características 1, 5.2, e 7 do Framework iStar.

Analisando a Tabela 1, observa-se que existem alguns atributos de Transparência que não têm relação com o Framework iStar. Por exemplo, o atributo *publicidade* não está relacionado a nenhuma característica do iStar. O mesmo ocorre com os atributos *portabilidade*, *uniformidade*, *simplicidade*, e outros nove atributos listados na primeira coluna da Tabela 1.

A partir dessa lacuna entre alguns atributos de Transparência e as características do Framework iStar foi adota a estratégia de utilizar sinônimos desses atributos para sinalizar sua presença em especificações de requisitos. Na próxima seção é apresentada essa lista de sinônimos.

Table 1. Relação entre as características do iStar e os atributos de transparência [6] [7].

Atributos de Transparência	Relação	ID	Característica iStar
Portabilidade	Ausente	1	Intencionalidade (*actor* e *goal*)
Disponibilidade	6	2	*Softgoal*
Publicidade	Ausente	3	Intencionalidade detalhada (decomposição de tarefas)
Uniformidade	Ausente	4	Alternativas de operacionalização (conectores *means-end*)
Simplicidade	Ausente		
Operabilidade	5.1	5	Dependência estratégica
Intuitividade	Ausente	5.1	Dependência de recurso
Desempenho	8	5.2	Dependência de meta
Adaptabilidade	Ausente	5.3	Dependência de meta flexível
Amigabilidade	Ausente	6	Classificação de atores (Associação entre atores INS e ISA)
Clareza	2, 5.2, 9	7	Estrutura organizacional (posições)
Completeza	2	8	Responsabilidades (papeis)
Corretude	Ausente	9	Contribuição positiva/negativas (*link contribution*)
Atualidade	Ausente		
Comparabilidade	Ausente		
Consistência	Ausente		
Integridade	4		
Acurácia	2		
Concisão	Ausente		
Compositividade	3, 5.3		
Divisibilidade	3, 8		
Detalhamento	4		
Dependência	5, 9		
Validável	4		
Controlabilidade	Ausente		
Verificabilidade	1, 5.2, 7		
Rastreabilidade	1, 5.2, 9		
Explicável	Ausente		

3 Sinônimos dos Atributos de Transparência

A partir da lista de atributos de transparência apresentados na Tabela 1 é possível observar que existem atributos que não possuem relação com os conceitos do iStar, inviabilizando sua identificação pelas regras de produção que exploram esse relacionamento. A partir desses atributos de transparência foram identificados seus sinônimos através da consulta à base de dados Wordnet [8]. A Tabela 2 contém a lista desses sinônimos e a partir dela foi criada a base de fatos contendo os atributos de transparência e seus respectivos sinônimos, exibida na Figura 2.

Table 2. Sinônimos dos atributos de Transparência, extraído da base Wordnet [8].

Atributos	Atributos	Sinônimo
Portável	portable	portable, portability
Disponível	available	availability, usable, useable, uncommitted
Publicidade	disclosure	revelation, revealing
Uniforme	uniform	consistent, undifferentiated, unvarying
Simples	simplicity	ease, easiness, simpleness
Operável	operable	operable, practicable, functional, usable, useable, operability, in working order
Intuitivo	intuitive	intuitive , nonrational, visceral
Desempenho	performance	execution, operation, functioning, carrying out, carrying into action
Adaptável	adaptable	adaptability, adaptable
Amigável	friendly	friendliness, friendly
Clareza	clarity	lucidity, lucidness, pellucidity, clearness, limpidity
Completa	complete	fill out, fill in, make out, completeness, full
Correto	correct	right, correctness, rightness
Atualizada	updated	update
Comparável	comparable	corresponding, like, comparison, compare, equivalence, comparability
Consistência	consistency	consistence
Integridade	integrity	unity, wholeness
Acurácia	accuracy	truth
Concisão	conciseness	concision, pithiness, succinctness
Compositividade	compound	compound
Divisível	divisible	divisibility
Detalhamento	detail	item, point, particular
Dependência	dependence	dependance, dependency
Validável	authenticate	authenticate
Controlável	controllable	controllable, controllability, governable
Verificável	verifiable	confirmable

Rastreável	traceable	trackable, traced
Explicável	explainable	understandable

A Figura 2 mostra parcialmente a base de dados de sinônimos dos atributos de transparência representada na sintaxe da linguagem CLIPS *C Language Integrated Production System* [9]. Através dessa base de dados, o mecanismo de inferência do sistema pode aplicar a regra de produção *ruleDenoteSynonym*, também exibida na Figura 2, para sinalizar uma possível ocorrência de sinônimos dos atributos de transparência no modelo de requisitos. Por exemplo, quando for citada a palavra *revelation* na especificação, a regra de produção será disparada informando que há o relacionamento com o atributo de transparência *disclosure.*

```
(deffacts words
      (word (id 1)    (word "portable") )
      (word (id 2)    (word "available") )
      (word (id 3)    (word "disclosure") )
      (word (id 4)    (word "uniform") )
      (word (id 5)    (word "simplicity") )
      (word (id 6)    (word "operable") )
      (word (id 7)    (word "intuitive") )
      (word (id 8)    (word "performance") )
      (word (id 23)   (word "dependence") )
      (word (id 24)   (word "authenticate") )
      (word (id 25)   (word "controllable") )
      (word (id 26)   (word "verifiable") )
      (word (id 27)   (word "traceable") )
      (word (id 28)   (word "explainable") )

      (word (id 30)  (word "portability")  (idAttribute 1))
      (word (id 31)  (word "availability") (idAttribute 2))
      (word (id 32)  (word "usable")       (idAttribute 2))
      (word (id 33)  (word "useable")      (idAttribute 2))


(defrule ruleDenoteSynonym
      (element (id ?id) (name ?name) (type ?type) ) ; o elemento
      (word (id ?idAttribute) (word ?nameAttribute) ) ; o atributo
      (word (id ?idSim) (word ?name) (idAttribute ?idAttribute)) ;o sinônimo
      =>
      (printout t ?name " denote " ?nameAttribute crlf)
)
```

Fig. 2. Parte da base de dados composta de sinônimos.

Conforme mostra a Figura 2, a palavra *portability* está relacionada com o atributo *portable*. Quando a regra de produção *ruleDenoteSynonym* for aplicada ao modelo de requisitos ela sinalizará essa ocorrência. O funcionamento dessa regra de produção consiste na utilização de três informações na base de dados. Para cada elemento do modelo de requisitos é verificada sua relação com os sinônimos dos atributos, e a partir dessa relação é recuperada a relação do sinônimo com o atributo de transparência. Se essa condição for satisfeita será sinalizado a ocorrência de um sinônimo de atributo de transparência na especificação de requisitos.

A abordagem utilizando a base de sinônimos dos atributos de transparência é uma extensão do trabalho apresentado em [10]. Conforme relatado na Tabela 1, há

atributos de transparência que não possuem relação com os conceitos do iStar, inviabilizando a identificação desses atributos.

4 Verificação de Atributos de Transparência em Especificações de Requisitos Representadas em iStar

A estratégia para verificar a aderência de uma especificação de requisitos representada com o *Framework* iStar aos atributos de transparência é realizada seguindo os seguintes passos:

1. Representar a especificação de requisitos no formato de fatos CLIPS [9], formando assim a base de dados do sistema de verificação.
2. Inspecionar essa base de dados com regras de produção que implementam a *relação* entre os conceitos do iStar e os atributos de transparência, conforme exibido na Tabela 1.
3. Inspecionar essa base de dados com a regra de produção *ruleDenoteSynonym* para identificar a ocorrência de sinônimos dos atributos de transparência nessa especificação de requisitos.

No primeiro passo, a especificação de requisitos construida com o Framework iStar é representada no formato de fatos, formando assim a base de dados do sistema de verificação. No trabalho [10] foram apresentadas duas abordagens para formar a base de dados. Por exemplo, a especificação de requisitos apresentada na Figura 1, os elementos *User* e *Media handled* são representados nos seguintes fatos:

```
(element ( id “01”)   (name “User” )            (type “actor” ))
(element ( id “02”)   (name “Media handled” )  (type “goal” )   (idActor “01”) )
```

A partir da especificação de requisitos descrita na base de dados, é possível verificar sua aderência aos atributos de transparência utilizando a base de conhecimento apresentada em [10]. Essa base de conhecimento é formada pela relação entre as características do iStar com os atributos de transparência apresentada na Tabela 1. Por exemplo, a característica do iStar *intencionalidade* está relacionada com os atributos de transparência *rastreabilidade* e *verificabilidade,* portando na base de conhecimento há uma regra que indica essa ocorrência. Na especificação da Figura 1, o elemento *Media handled* é uma meta do ator *User*, essa relação indica a intencionalidade desse ator, portanto será sinalizada a presença dos atributos *rastreabilidade* e *verificabilidade* nesse parte da especificação.

Uma vez realizada a segunda etapa, a base de dados é verificada à procura de sinônimos dos atributos de transparência. Nesse sentido a regra *ruleDenoteSynonym* verifica a ocorrência desses sinônimos na especificação de requisitos. Por exemplo, na especificação de requisitos exibida na Figura 1 o elemento *Ease* é um sinônimo do atributo *simplicity*, portanto essa regra de produção sinalizará a ocorrência desse atributo nessa especificação, “*Ease denote simplicity*”.

5 Trabalhos correlacionados

No trabalho [10] foi apresentada a abordagem para verificar a aderência de uma especificação de requisitos aos atributos de transparências a partir do relacionamento entre as características do iStar e os atributos de transparência. Nesse trabalho, [10], foram discutidas duas abordagens para formar a base de dados a partir de uma especificação de requisitos em iStar, além de apresentar a base de conhecimento que implementa a relação entre os atributos de transparência e as características do Framework iStar. Porém, a partir da análise da Tabela 1, observa-se que existem atributos de transparência que não têm relação com as características do iStar, consequentemente a base de conhecimento apresentada nesse trabalho não pode identificar sua presença. A partir dessa necessidade foi apresentada no presente texto uma base de dados composta por sinônimos dos atributos de transparência.

No trabalho desenvolvido em [12] foi apresentada uma abordagem para tornar o *software* mais transparente, onde foi discutida uma forma de documentação de sistemas de *software* fundamentada em aspectos técnicos e de uso. As informações técnicas são direcionadas aos engenheiros de *software*, tais como: arquitetura e linguagem de programação. As informações de uso são destinadas aos usuários finais do sistema, por exemplo, visão geral do *software*, suas funcionalidades e também os requisitos que não foram implementados. Essa abordagem recebeu o nome de *BuS* (Bula de Software) inspirada na bula de remédios que agrega informações sobre medicamentos.

Para organizar as informações sobre o *software* foi utilizada a tecnologia XML (*eXtensible Markup Language*). As informações técnicas e de uso são estruturadas em tópicos através da utilização de *tags*. A *tag* <*CABECALHOSOFTWARE/*> descreve a seção do documento contendo o nome do *software*, sua composição e seus objetivos. Na *tag* <*INFORMACAOGERALCIDADAO/*> ficam estruturadas as informações sobre o uso do sistema e sobre o seu escopo. Por fim, na *tag* <*INFORMACAOTECNICA/*> são disponibilizadas informações sobre a tecnologia envolvida no desenvolvimento do *software* e que tem como público alvo os engenheiros de *software*.

A documentação de *software BuS* visa potencializar a transparência contribuindo para que o *software* seja aderente aos atributos de transparência *uniformidade, amigabilidade, simplicidade, intuitividade, operabilidade* e *clareza. A* abordagem apresentada nesse texto procura indícios da presença de atributos de transpârencia em especificações de requisitos iStar. As duas estratégias possuem objetivos semelhantes, ambas buscam tornar o *software* mais transparente.

O trabalho desenvolvido por [13] aborda a verificação de requisitos a partir de processamento de linguagem natural com o objetivo de identificar ambiguidades em especificações de requisitos. Os requisitos representados em linguagem natural são submetidos ao procedimento de medição de ambiguidades, onde as sentenças potencialmente ambíguas são verificadas. Além disso, é identificado o motivo que tornaria a sentença ambígua auxiliando o trabalho do engenheiro de *software* na remoção de tais ambiguidades. A abordagem de verificação de atributos de transparência desenvolvida nesse trabalho não utiliza processamento de linguagem

natural, porém procura identificar sinônimos desses atributos de transparência nos modelos de requisitos.

No trabalho de [14] a metodologia Tropos é empregada para verificar os aspectos dinâmicos de dependência entre atores estratégicos. O modelo de especificação de requisitos iStar é representado na linguagem *Formal Tropos* onde atores, metas, tarefas, recursos e dependências são declarados juntamente com as restrições usadas para verificar o modelo. O método de verificação de atributos de transparência difere dessa abordagem, pois o modelo a ser verificado é organizado em uma base de dados enquanto o conhecimento usado para a verificação é implementado nas regras de produção. Há dois módulos distintos para realizar a verificação: a base de fatos e a base de regras. A base de fatos é composta pela especificação de requisitos, enquanto a base de regras é formada pelas regras de produção que são aplicadas à base de fatos para inferir a presença ou ausência dos atributos de transparência.

A estratégia utilizada no trabalho de [15] emprega a tecnologia XML/XSLT para verificar especificações de requisitos sob os aspectos de ambiguidade, completude e rastreabilidade. A verificação dos requisitos é realizada através da aplicação de folhas de estilos sobre o modelo de requisitos representado em XML. Enquanto a abordagem aqui apresentada utiliza regras de produção para verificar se na especificação de requisitos representada com o Framework iStar há indícios de atributos de transparência.

A diferença entre a estratégia utilizada nesse trabalho e as abordagens utilizadas por [13], [14] e [15] está no objetivo da verificação. Ela busca identificar atributos de transparência em especificações de requisitos, enquanto as demais procuram verificar se essas especificações estão em conformidade com as necessidades e interesses dos envolvidos no ambiente organizacional onde o sistema irá operar. A transparência de *software* será mais uma entre as expectativas dos *stakeholders* à medida que ela for exigida pelos diversos segmentos da sociedade. Além disso o uso de uma base de sinônimos dos atributos de transparência procura expandir a abordagem empregada em [10].

6 Conclusões

A abordagem baseada em sistema de inferência permitiu a obtenção de um sistema flexível e expansível, que proporciona seu uso para verificar especificações de requisitos e para sugerir a inclusão de novos atributos em especificações analisadas pelo sistema. Essa estratégia permite também o uso do sistema como uma plataforma de ensino e disseminação das ideias de transparência e sua inclusão em sistemas de informação organizacionais. A representação do modelo de requisitos iStar em uma base de dados permite que outros tipos de verificação sejam realizadas. Por exemplo, a verificação sintática do modelo segundo as regras do Framework iStar e pode também ser integrada às ferramentas de modelagem iStar.

A separação entre a base de dados contendo a especificação de requisitos e as regras de produção proporcionou a obtenção de um sistema de verificação composto por módulos independentes. Novas regras de produção podem ser adicionadas à medida

que outras relações entre os atributos de transparência e o iStar forem propostas. A base de conhecimento desenvolvida pode ser utilizada por um agente inteligente baseado em conhecimento.

O sinônimos dos atributos de transparência utilizados para indicar a presença de palavras na especificação de requisitos que indicam os atributos de transparência é uma proposta inicial que pode ser melhorada a partir de sua utilização em modelos de requisitos disponíveis na literatura.

Referências Bibliográficas

1. HOLZNER B.; HOLZNER L. **Transparency in Global Change:** The Value of Open society. 1. ed. Pittsburgh, PA, EUA: Universidad of Pittsburgh Press, 2006.
2. CAPPELLI, C.; OLIVEIRA, A. PADUA; LEITE, J. C. S. P. **Exploring Business Process Transparency Concepts**, RE 207, IEEE Computer Society Press, pp. 389-390, 207.
3. CAPPELLI, C.; LEITE, J. C. S. P. **Transparência de Processos Organizacionais.** In: Simpósio Internacional de Transparência nos Negócios, 2008, Rio de Janeiro, Brasil. **Anais**... Rio de Janeiro: Universidade Federal Fluminense, LATEC, 2008. Disponível em: <http://www.latec.uff.br/transparencia/anais.pt-br.php>. Acesso em: 25 jan. 2011.
4. YU, Eric. **Modelling Strategic Relationships for Process Reengineering.** 1995. 124 f. Tese de Doutorado – Dept. of Computer Science, University of Toronto, Toronto, 1995.
5. 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 (XIV CibSE - Congresso Ibero-Americano em Engenharia de Software), 2011, Rio de Janeiro. **Proceedings** XIV CibSE. Rio de Janeiro, 2011. v. XIV. p. 321-332. Disponível em: <http://wer.inf.puc-rio.br/WERpapers/papers_by_conference.lp?conference=WER11. Acesso em: 20 jan. 2012.
6. LEITE, J.C.S.P, CAPPELLI, C., Exploring i* Characteristics that Support Software Transparency. In Proceedings of the 3rd International i* Workshop, CEUR Workshop Proceedings, Vol. 322, 208, pp. 51-54. Disponível em: <http://CEUR-WS.org/Vol-322/>. Acesso em 24 set. 2011.
7. OLIVEIRA, A. P. A.; CAPPELLI, C.; CUNHA, H. S.; LEITE, J. C. P.; WERNECK, V. M. B. **Engenharia de software Intencional: Tornando o Software Mais Transparente.** In: Simpósio Brasileiro de Engenharia de Software, 2007, Rio de Janeiro. Tutorial apresentado no XXI Simpósio Brasileiro de Engenharia de Software, Rio de Janeiro, 2007. Disponível em: <http://www.sbbd-sbes2007.ufpb.br/ tuto3.pdf>. Acesso em: 22 jan. 2011.
8. WORDNET: A lexical database for English. Disponível em: <http://wordnet.princeton. edu/wordnet/>. Acesso em: 02 maio 2012.
9. GIARRATANO, J. C. **CLIPS User's Guide**. Version 6.0. NASA Lyndon B. Johnson Space Center, Software Technology Branch, Houston, Texas, EUA, 1993.
10. BAIA, J. W.; BRAGA, J. L.; CARVALHO, L. F. **Verificação de Requisitos de Transparência em Modelos iStar.** In: XV WER - Workshop em Engenharia de Requisitos (XIV CibSE - Congresso Ibero-Americano em Engenharia de Software), 2012, Bueno Aires . **Proceedings** XV CibSE. Buenos Aires, 2012. v. XV. Disponível em:

<http://wer.inf.puc-rio.br/WERpapers/papers_by_conference.lp?conference=WER12>. Acesso em: 20 jul. 2012.

11. SILVA, C.; BORBA, C.; CASTRO, J. **A Goal Oriented Approach to Identify and Configure Feature Models for Software Product Lines**. In: Workshop em Engenharia de Requisitos – WER11, (XIV CibSE - Congresso Ibero-Americano em Engenharia de Software), 2011, Rio de Janeiro. **Proceedings** XIV CibSE. Rio de Janeiro, Brasil, 2011. v. XIV. p. 321-332. Disponível em: <http://wer.inf.puc-rio.br/WERpapers/papers_by_conference.lp?conference=WER11>. Acesso em: 20 jan. 2012.
12. LEAL, A. L. de C.; ALMENTERO, Eduardo; CUNHA, Herbert; SOUSA, H. P.; LEITE, J. C. S. P. **Bula de Software:** Uma Estrutura Definida para Promover a Melhoria da Transparência em Software**.** In: XV Workshop on Requirements Engineering - WER2012, 2012, Buenos Aires, Argentina. **Anais** ... Buenos Aires, Argentina: UNLaM - Universidad Nacional de La Matanza, 2012. v. 15. Disponível em: <http://wer.inf.puc-rio.br/WERpapers/ papers_by_ conference.lp?conference=WER12>. Acesso em 11 jun. 2012.
13. KIYAVITSKAYA, Nadzeya; ZENI, Nicola; MICH, Luisa; BERRY, Daniel M. **Requirements for Tools for Ambiguity Identification and Measurement in Natural Language Requirements Specifications**. In: Workshop em Engenharia de Requisitos - WER07, 2007, Toronto. **Proceedings** .... Toronto: York University, 2007. p. 197 - 206. Disponível em: <http://wer.inf.puc-rio.br/WERpapers/papers_by_conference.lp?conference=WER07>. Acesso em: 23 dez. 2011.
14. FUXMAN, A.; MYLOPOULOS, J.; PISTORE, M.; TRAVERSO, P. **Model Checking Early Requirements Specifications in Tropos**. In: RE-2001, 9th IEEE International Requirements Engineering Conference, 2001, Toronto, Canada. **Proceedings** on Fifth IEEE International Symposium. Toronto, Canada:[s.n.], 2001.
15. DURAN, A.; BERNARDEZ, B.; RUIZ, A.; TORO, M. **An XML-based approach for the automatic verification of software requirements specifications**. In: Fourth Workshop on Requirements Engineering, 2001, Buenos Aires, Argentina. **Proceedings** ... Buenos Aires: Universidad Tecnológica Nacional, 2001. p. 181-194. Disponível em: <http://www.inf.puc-rio.br/wer01/>. Acesso em 23 dez. 2011.

Correcciones semánticas en métodos de estimación de completitud de modelos en lenguaje natural

Claudia S. Litvak[1,2], Graciela D. S. Hadad[1,2], Jorge H. Doorn[2,3]

[1]Facultad de Ingeniería y Tecnología Informática, Universidad de Belgrano, Argentina
[2]DIIT, Universidad Nacional de La Matanza, Argentina
[3] Fac. Ciencias Exactas, Universidad Nacional del Centro de la Provincia de Buenos Aires

{claudia.litvak,graciela.hadad}@comunidad.ub.edu.ar,
jdoorn@exa.unicen.edu.ar

Abstract. La Ingeniería de Requisitos tiene como objetivo producir requisitos de alta calidad, poniendo especial interés en la completitud de los modelos elaborados. Establecer si la información elicitada y modelada es suficiente para construir un software que cubra las necesidades del cliente es una cuestión de difícil respuesta. Existen algunos intentos realizados en ese sentido, tal como una adaptación del método de captura-recaptura para estimar el tamaño de modelos de requisitos escritos en lenguaje natural. Basados en dicho trabajo y considerando la naturaleza de estos modelos, proponemos introducir un análisis semántico previo a estimar el tamaño, dado que el método predictivo solo contempla aspectos formales y cuantitativos. Dicho análisis semántico estudia la relevancia, pertenencia, sinonimia y homonimia del contenido textual del modelo. Comparando los resultados estadísticos de trabajos precedentes contra los obtenidos realizando correcciones semánticas, concluimos que estas son beneficiosas para la estimación de la completitud de modelos en lenguaje natural.

Keywords. modelado de requisitos, completitud de modelos, método de captura-recaptura, modelos en lenguaje natural.

1 Introducción

La completitud de los requisitos no solo es uno de los criterios de calidad pregonados en la literatura, sino en el ámbito práctico del desarrollo de software. El estándar IEEE 830-1998 reafirmado en 2009 [1] la considera una de las 8 propiedades que debe cumplir una buena especificación de requisitos de software.

Con lo cual, la incompletitud es uno de los principales problemas que afronta la Ingeniería de Requisitos [2-4]. La dificultad radica en poder determinar si se ha logrado elicitar y modelar toda la información necesaria para desarrollar un sistema de software que cubra las expectativas y necesidades de los clientes y usuarios. Según Leite [5], el proceso de definición de requisitos es intrínsecamente incompleto debido a la complejidad del mundo real. Esto significa que no hay ninguna posibilidad de

alcanzar la completitud de un modelo, con lo cual el principio conductor debe ser elaborar un modelo lo más completo posible.

La incompletitud es un problema común a otras actividades del proceso de desarrollo de software. En el diseño, es lógico cuestionarse si se han identificado todas las clases, todos los métodos y todos los atributos. En las inspecciones de código o de algún otro artefacto permanece siempre la duda si se han detectado todos los defectos. La misma incógnita se plantea en la etapa de testing. Es decir, pueden existir cuestiones no descubiertas en diversos artefactos o actividades a lo largo de todo el desarrollo de software y persistir estas sin ser detectadas, con la consecuente degradación en la calidad del software e inevitable insatisfacción del cliente. Aún cuando la completitud es un criterio de calidad pregonado en la literatura y exigido por muchos estándares, su cumplimiento es con mucha frecuencia parcialmente alcanzado.

Entonces, el problema de la incompletitud está directamente relacionado con un tipo específico de defecto: las omisiones. Las técnicas de verificación, en algún grado, y de validación, menos aún, ayudan a mitigar este problema, así como el uso de técnicas de elicitación apropiadas permiten lograr una adquisición de conocimiento más completa. Sin embargo, estas técnicas por sí solas no permiten eliminar el problema ni siquiera estimar el grado de completitud alcanzado. Esto significa que, si no se puede asegurar la completitud, al menos se debiera ser consciente del nivel aproximado de completitud alcanzado en los modelos para decidir cuándo avanzar hacia otras etapas del proceso de desarrollo o continuar elaborándolos.

El método de captura-recaptura [6] es un método de predicción del tamaño de poblaciones cerradas que se ha utilizado en Ingeniería de Software para estimar el número de defectos aún no descubiertos en artefactos de código [7-9] y en documentos de requisitos [10]. Posteriormente, este método se aplicó en la Ingeniería de Requisitos para predecir el número de elementos faltantes en un modelo de requisitos [11-12]. En el trabajo de Walia y Carver [10] se realizan varias inspecciones independientes sobre el mismo modelo de requisitos para estimar defectos en dicho modelo, mientras que Doorn y Ridao [11-12] realizaron la estimación de elementos no elicitados en un dominio de aplicación a partir de varios ejemplares de un modelo, construidos independientemente.

La adaptación del método de captura-recaptura realizado por Wohlin y Runeson [7], llamado Detection Profile Method (DPM), fue aplicado por Doorn y Ridao [11-12] para estimar el tamaño de dos modelos escritos en lenguaje natural: el Léxico Extendido del Lenguaje y Escenarios. El método predictivo requiere tener dos o más muestras de la misma población, por lo cual Doorn y Ridao utilizaron nueve muestras de ambos modelos. Cada muestra fue producida por diferentes grupos de elicitadores. De acuerdo con los resultados que obtuvieron, el número de elementos omitidos en cada muestra era de significativa importancia. El grupo de elicitadores que descubrió el mayor número de elementos sólo alcanzó un nivel de completitud del 51%, medido en referencia a la cantidad de elementos presentes en la unión de todos los valores muestrales obtenidos por todos los grupos. A primera vista, estos niveles indicarían que el modelo construido por cada grupo de elicitadores es bastante pobre. Esta sola experiencia muestra claramente que el problema de la incompletitud es efectivamente mucho más serio que lo que en principio se presupone.

En base a esto, hemos conjeturado acerca de la posibilidad de que no se hayan identificado los elementos adecuados en cada una de las muestras al aplicar el método de captura-recaptura. Por lo que, hipotetizamos que se debería realizar un análisis previo de los elementos de cada muestra antes de establecer la estimación del tamaño del modelo, con el fin de mejorar la calidad de dicha estimación. Los resultados iniciales que obtuvimos confirman una mejora en la estimación del tamaño para un modelo específico escrito en lenguaje natural.

En la siguiente sección se presentan los trabajos anteriores y la motivación del presente trabajo. En la sección 3 se describe la hipótesis planteada y el trabajo desarrollado para comprobarla, detallando el análisis semántico realizado. En la sección 4 se detallan observaciones del estudio semántico-estadístico realizado y, finalmente, se exponen conclusiones y trabajos futuros.

2 Marco de Trabajo

Doorn y Ridao en [11] han estudiado experimentalmente el uso de DPM [7] para estimar el número de términos faltantes en un glosario, llamado Léxico Extendido del Lenguaje (LEL) [13]. El LEL es un glosario de términos (símbolos) que se utilizan en el universo de discurso. Cada término se describe por uno o más nombres (sinónimos), una noción (denotación), un impacto (connotación) y un tipo (clasificación del símbolo) que guía la definición del término.

En dicho trabajo [11], concluyeron que aumentando el número de elicitadores (ingenieros en requisitos), la brecha entre el número de elementos elicitados y el número de elementos estimados disminuye. Cuando se refieren al número de elicitadores, esto implica que se dispone de más muestras de la población, es decir, más modelos-muestra.

Se analizaron los resultados que obtuvieron Doorn y Ridao aplicando DPM al modelo LEL. El número estimado de elementos faltantes en el LEL fue 9 de los 118 términos elicitados en conjunto por todos los grupos. En principio, el número de elementos faltantes parece ser un número bajo, sin embargo, esto no es así cuando se estudia la muestra de cada grupo de elicitadores independientemente. Si observamos los resultados obtenidos por dos grupos cualesquiera, por ejemplo el Grupo 1 y el Grupo 5, ellos coinciden en 34 elementos, pero difieren en 50. Esta diferencia se manifiesta en que al Grupo 1 le faltó encontrar 32 elementos que sí fueron detectados por el Grupo 5, y al Grupo 5 le faltaron 18 elementos del Grupo 1. Luego, se observa que las diferencias superan ampliamente las coincidencias. Cabe destacar que estas diferencias señalan omisiones en las respectivas muestras del modelo. Esta situación se agrava si comparamos un grupo respecto a todos los otros, por ejemplo, al Grupo 1 le faltó encontrar en total 66 elementos, mientras que el Grupo 5 omitió 53 elementos. Aquí entonces, la cantidad de omisiones es más notoria aún. El hecho de que dos grupos difieran en alrededor de 50 términos requiere estudios más profundos para explicar estas diferencias.

Se debe tener en claro que la experiencia reportada en [11] y revisada en este artículo no es un trabajo profesional viable ya que se multiplica el esfuerzo necesario por

un factor de 9. Este estudio sólo es posible en el contexto de un proyecto de investigación que pretende ponderar de manera cuantitativa la magnitud de la falta de completitud. Por otro lado, ninguna de estas 9 muestras del modelo sufrió un proceso de validación. Se debe notar que en general una validación con clientes permite detectar básicamente datos erróneos y contradicciones, mientras que la identificación de omisiones tiene una baja incidencia durante una validación debido a que los clientes habitualmente se centran en analizar lo que está presente en el modelo. En otras palabras, en el mejor de los casos, las personas provenientes de la organización receptora del sistema tienen por objetivo colaborar hasta donde les sea posible con la redacción de un documento, por ejemplo un glosario, cuyo propósito no les es propio. Está fuera de su intención y habitualmente de su capacidad darse cuenta de qué palabras o frases, ellas usan frecuentemente y que son importantes para la Ingeniería de Requisitos. Por estos motivos, ellas se darán cuenta fácilmente de errores en la definición de un término y, en el mejor de los casos, de debilidades, pero seguramente estarán lejos de poder contribuir espontáneamente con nuevos términos.

Basándonos en el estudio de Doorn y Ridao del modelo LEL y considerando todas las omisiones observadas anteriormente, realizamos un nuevo estudio bajo la hipótesis de que los grupos de elicitadores estaban observando diferentes universos de discurso [14-15]. En dicho trabajo previo, dividimos el universo bajo estudio en cinco sub-áreas para facilitar la realización del análisis estadístico. Confirmamos estadísticamente, que en cuatro sub-áreas los grupos habían observado universos con límites diferentes, mientras que en sólo una sub-área, todos estaban observando el mismo universo. La Fig. 1 presenta las tres visiones diferentes de la sub-área Administración de Bien Tipo. Esto significa que 4 grupos observaron el mismo universo en esta sub-área, 3 grupos observaron otro universo y otros 2 grupos tuvieron otra visión del universo. Es de destacar que estos tres universos fueron detectados y confirmados utilizando técnicas estadísticas, habiéndose logrado un grado de confianza muy alto en el resultado.

Estas conclusiones nos llevaron a formular una nueva hipótesis bajo la cual debemos estudiar el contenido textual de las muestras internamente (intra-análisis semántico) y entre muestras (inter-análisis semántico) para unificar los diferentes límites observados del mismo universo y poder estimar adecuadamente el tamaño del modelo.

Es de destacar que tanto los estudios originales [11-12] como esta revisión, se basan en datos no influenciados por el trabajo de investigación realizado, ya que en el momento de construcción de los 9 modelos léxicos no existía ninguna intención de realizar un estudio comparativo de los mismos, siendo que este estudio fue realizado varios años después, dado lo cual no hubo contaminación por parte de los investigadores. Por otro lado, debemos mencionar que estos modelos se basaron en un caso real detallado en forma escrita, evitando la contaminación cruzada entre grupos de elicitadores. Debe notarse, además, que el número 9 de muestras es un valor importante para aplicar métodos formales de estimación.

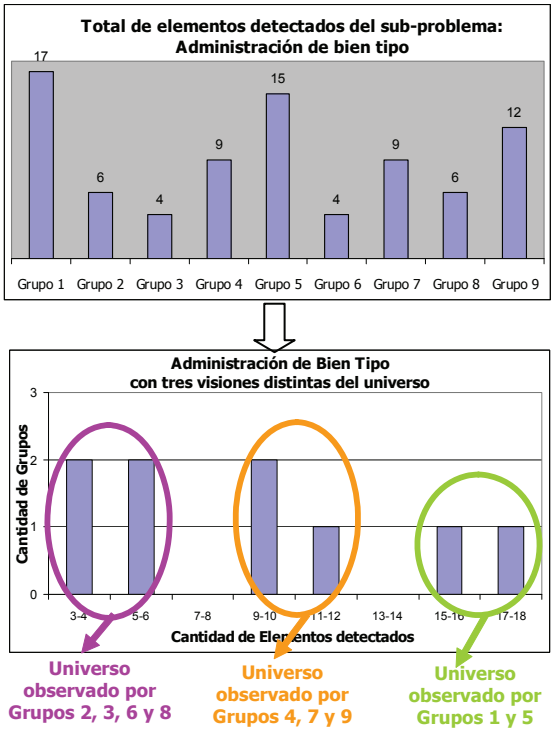


Fig. 1. Visiones de los elicitadores en una sub-área: Administración de Bien Tipo [14]

3 Desarrollo del trabajo

3.1 Hipótesis de Trabajo

En base al análisis de los resultados presentados en un artículo preliminar [15], planteamos la siguiente hipótesis de trabajo: *"Se requiere un análisis semántico sobre los elementos del modelo en lenguaje natural antes de aplicar métodos formales de estimación de tamaño del modelo"*.

Para comprobarlo, partimos de las mismas muestras utilizadas por Doorn y Ridao en [11], es decir, dispusimos de 9 muestras del modelo LEL generadas por 9 grupos de ingenieros (alumnos de posgrado y profesores) que estudiaron el mismo universo: el sistema de Planes de Ahorro Previo para la Adquisición de Vehículos 0km.

Comenzamos realizando un estudio estadístico para estimar el tamaño del modelo LEL sin ningún ajuste en los elementos de las muestras (símbolos del léxico generado por cada grupo de elicitadores). Luego, analizamos semánticamente cada elemento de la muestra, para finalmente volver a realizar la estimación del tamaño del modelo con las correcciones semánticas necesarias. En las siguientes sub-secciones detallamos estas actividades.

3.2 Trabajo Estadístico Inicial

Para realizar el estudio estadístico aplicando DPM sobre las 9 muestras, se requirió primero identificar los elementos comunes (símbolos) elicitados por todos los grupos, luego identificar aquellos elicitados por solo 8 grupos, y así sucesivamente hasta obtener los elementos elicitados por un solo grupo. Es decir, se determinó la unión de las 9 muestras, registrando la frecuencia de aparición de cada elemento (en cuantos grupos apareció el mismo elemento, ver Tabla 1).

La identificación de elementos comunes entre distintas muestras se realizó por simple sinonimia de nombre. Esto significa que dos símbolos de distinto grupo representan un único elemento del universo si ambos tienen el mismo nombre, o coinciden en al menos un nombre (caso que el símbolo tenga sinónimos). Cabe aclarar que la comparación de nombre consideró variantes gramaticales, tales como flexiones nominales, flexiones verbales y formas sustantivas de verbos. Este fue un criterio similar al aplicado en [11] con menos refinamiento, para identificar el conjunto de símbolos distintos detectados por todos los grupos.

Tabla 1. Frecuencia de aparición de cada símbolo en el conjunto de muestras

SÍMBOLOS	G1	G2	G3	G4	G5	G6	G7	G8	G9	Frecuencia
Solicitud de adhesión / Solicitud / Pla	1	1	1	1	1	1	1	1	1	9
Solicitante	1	1	1	1	1	1	1	1	1	9
Adherente / Miembro de grupo / Titula	1	1	1	1	1	1	1	1	1	9
Administradora / Sociedad Administra	1	1	1	1	1	1	1	1	1	9
Sorteo / Sorteo mensual / Sortear / Ac	1	1	1	1	1	1	1	1	1	9
Bien tipo / Automóvil / Bien / Vehículo	1	1	1	1	1	1	1	1	1	9
Fabricante / Importador / Proveedor	1	1	1	1	1	1	1	1	1	9
Plan de ahorro / Plan / Sistema de Ah	1	1	1	1	1	1	1	1	1	9
Grupo / Grupo de Adherentes	1	1	1	1	1	1	1	1	1	9
Comunicación fehaciente / Notificació	1	1		1	1	1	1	1	1	8
Adjudicatario / Adjudicado	1	1	1	1	1		1	1	1	8
Licitación / Adjudicar por licitación / A	1	1		1	1		1	1	1	7
Adjudicación / Acto de adjudicación /	1	1	1		1	1	1		1	7
Cuota pura	1	1	1	1	1			1	1	7
….. …. …. ….. ….. …..										
Cesionario / Adherente Cesionario					1				1	2
Incumplimiento imputable al grupo					1				1	2
Notificacion de rechazo	1									1
Aceptación / Aceptado			1							1
Desistir de Solicitud					1					1
Rechazo de solicitud					1					1
Solicitante aceptado							1			1
Cantidad de símbolos: 132	**54**	**29**	**31**	**35**	**67**	**27**	**46**	**24**	**55**	**368**

El método DPM [7] aplicado se basa en ajustar la curva que representa los datos referidos a la cantidad de grupos que elicitaron cada símbolo (obtenidos de la planilla confeccionada cuyo extracto se muestra en la Tabla 1), utilizando una curva teórica simple. En este caso se usó una función exponencial decreciente, ecuación (1), cuyos parámetros incluyen las variaciones de probabilidad de detección de un símbolo y de la habilidad de elicitación del grupo.

Entonces, la ecuación que representa la curva de ajuste exponencial es:

$$m_x = A \times e^{-bx} \quad (1)$$

siendo

m_x : cantidad total de grupos que detectaron el símbolo x
b : decrecimiento de la función exponencial
A : constante

Utilizando técnicas de regresión se estimaron los parámetros A y b, obteniendo A = 7,0514 y b = 0,0191, con un coeficiente de determinación $R^2 = 0,8529$. Este coeficiente de determinación es un parámetro para evaluar la distancia entre la estimación y los datos experimentales ($0 \leq R^2 \leq 1$). La cantidad total de símbolos estimados se calcula por el mayor valor de x para el cual la ecuación (1) produce un valor mayor o igual que 0,5, es decir, haciendo $m_x = y$ para y = 0,5. Esto da como resultado el valor estimado 138. La Fig. 2 representa esta curva de aproximación.

Entonces, entre todos los grupos encontraron 132 símbolos (sin repetición) de los 138 símbolos estimados aplicando DPM, es decir, les ha faltado, en conjunto, elicitar 6 símbolos.

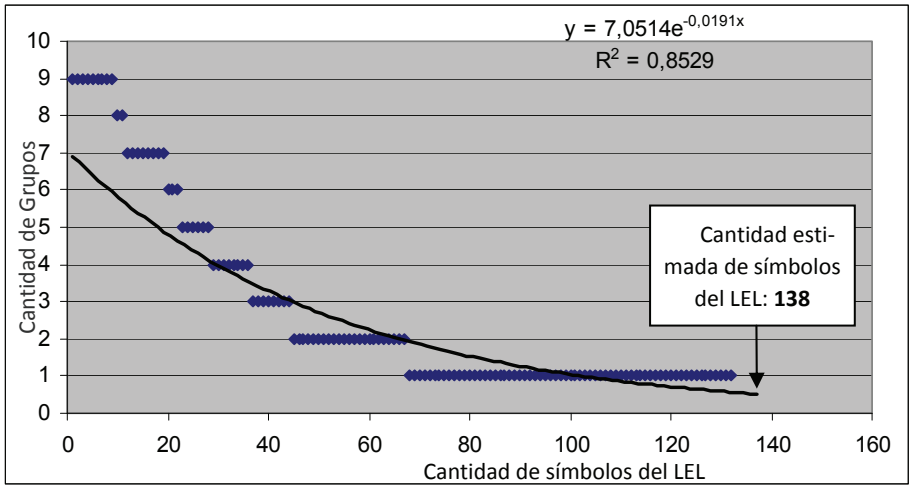


Fig. 2. Aplicación de DPM antes de la corrección semántica

3.3 Análisis Semántico

Se realizó un estudio de la semántica de todos los términos del LEL definidos por los 9 grupos de elicitadores. Ahora, además de considerar los nombres de los símbolos, se analizó la denotación y connotación de los mismos.

Primero, se hizo un análisis semántico dentro de cada muestra, independiente de las restantes, donde se estableció la pertenencia y la relevancia del símbolo en el universo de discurso. Posteriormente, se realizó un análisis semántico entre símbolos de distintas muestras para establecer sinónimos y homónimos. Tanto en el intra-análisis semántico como en el inter-análisis, el estudio comenzó por aquellos símbolos de

menor frecuencia de aparición en el conjunto de todas las muestras. Una baja frecuencia implica que un único grupo o unos pocos grupos elicitaron el símbolo.

Para establecer la pertenencia, primero se validó la existencia del nombre del símbolo en dicho universo. Luego, se analizó su contenido (denotación y connotación) para establecer si el símbolo estaba fuera de los límites del universo bajo estudio o no. En el caso de que el contenido del símbolo tuviese información del universo pero el nombre fuese inválido (creado artificialmente por los elicitadores), podía existir otro nombre en el universo no detectado por el grupo elicitador. En este punto, el símbolo quedó marcado para ser tenido en cuenta en el inter-análisis semántico para establecer sinónimos con símbolos de otros grupos.

La relevancia se analizó en base a la denotación y connotación del símbolo. La relevancia de un símbolo se determinó por el aporte de conocimiento al modelo LEL. Esto es, se consideraron no relevantes aquellos símbolos con exceso en el nivel de detalle, innecesario para un glosario, o con información parcial ya contenida en otro símbolo.

El inter-análisis semántico comenzó por confirmar los sinónimos ya establecidos para el estudio estadístico inicial (presentado en la anterior sub-sección). Ahora, ya no alcanzó tener nombres de símbolos idénticos o nombres con variaciones gramaticales, se debían tener contenidos similares. Por lo tanto, se confirmaron sinónimos estudiando la denotación y connotación entre cada par de símbolos. En el caso de diferencias, se descartó la relación semántica de sinonimia, y se analizó la posibilidad de homónimos (homógrafos). Luego, este análisis semántico continuó estableciendo nuevos sinónimos entre símbolos de distintas muestras cuyos nombres no eran idénticos (ni las flexiones gramaticales). Para ello, se compararon las denotaciones y connotaciones de cada símbolo.

En resumen, como consecuencia del intra-análisis semántico, hubo símbolos que se descartaron, fueron aquellos que cumplían con algunas de las siguientes características: a) con información fuera del límite del universo de discurso; b) con información con un exceso de nivel de detalle, innecesaria para un glosario, que podía ser mencionada dentro de otro símbolo; y c) con información que ya estaba contenida en otros símbolos.

Como consecuencia del inter-análisis semántico, se ajustó la cantidad de símbolos sin repetición del conjunto de muestras, en base a los siguientes aspectos: a) detección de sinónimos entre muestras, y b) detección de homónimos entre muestras.

La Fig. 3 muestra los 9 grupos con los símbolos del LEL detectados por cada uno para la sub-área Adhesión. En esta figura se identifican los símbolos descartados: 8 símbolos de un total de 21 símbolos sin repetición, siendo el conjunto muestral de 60 símbolos. Las intersecciones representan los sinónimos. Se puede observar visualmente que hubo 4 símbolos compartidos por todos los grupos para esta sub-área, y que a su vez coincidieron con el conjunto muestral de los Grupos 2 y 8.

Los resultados del ajuste semántico se resumen en la Tabla 2. Antes del estudio semántico se habían detectado un total de 132 símbolos sin repetición, elicitados en conjunto por los 9 grupos, mientras que concluido el estudio semántico se descartaron 35 símbolos (esta diferencia está dada por no existentes, no relevantes, no identifica-

dos como sinónimos y desglose de homónimos inicialmente considerados como sinónimos), quedando entonces un conjunto de 97 símbolos.

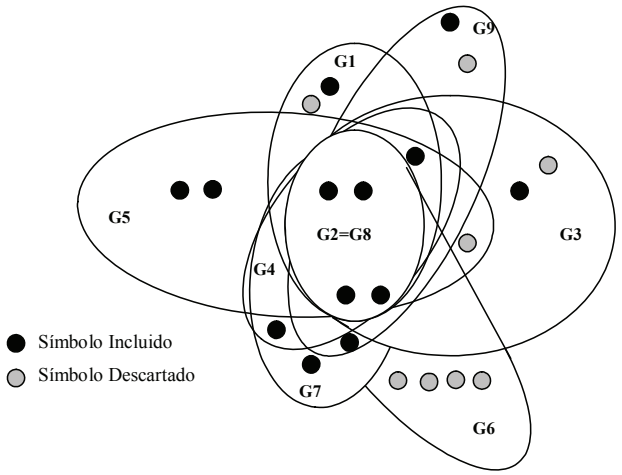


Fig. 3. Distribución visual de símbolos por grupos para sub-área Adhesión

Tabla 2. Resultados del análisis semántico

	PRE Análisis Semántico	POST Análisis Semántico
Grupo 1	**54**	**46**
Grupo 2	**29**	**28**
Grupo 3	**31**	**24**
Grupo 4	**35**	**35**
Grupo 5	**67**	**62**
Grupo 6	**27**	**14**
Grupo 7	**46**	**44**
Grupo 8	**24**	**22**
Grupo 9	**55**	**51**
Total de símbolos elicitados	**368**	**326**
Total de símbolos sin repetición	**132**	**97**

Símbolos Descartados	
Nombre no existe y no es relevante	9
Nombre existe pero símbolo no es relevante	23
Sinónimo no detectado en el grupo	3
Homónimo no detectado	(3)
Total de Símbolos Sin Repetición Descartados	**35**

3.4 Estimación Estadística con Ajuste Semántico

Una vez realizado el ajuste semántico se tuvo una nueva frecuencia de aparición de cada símbolo en el conjunto de las muestras. Se aplicó nuevamente DPM a las muestras corregidas y se obtuvo la curva de ajuste presentada en la Fig. 4, donde se estima-

ron los nuevos valores de A y b, obteniéndose A = 9,561 y b = 0,0279, con un coeficiente de determinación R^2 = 0,9394. Ahora, la cantidad de símbolos estimados fue de 105, es decir, han faltado encontrar 8 símbolos del conjunto corregido semánticamente.

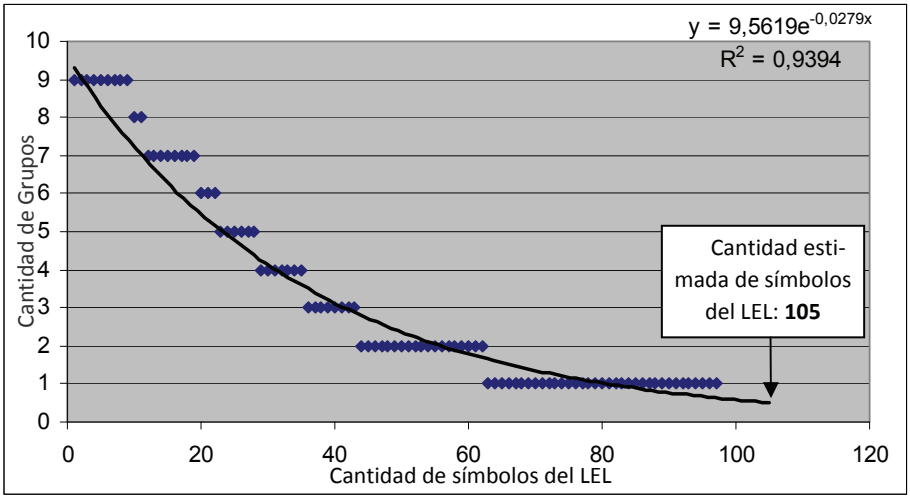


Fig. 4. Aplicación de DPM después de la corrección semántica

Los coeficientes de determinación permiten comparar ambos gráficos, Fig. 2 y 4. En el primer caso, sin las consideraciones semánticas (Fig. 2), el coeficiente de determinación obtenido fue R^2 = 0,8529. Al aplicar la corrección semántica (Fig. 4), el valor fue R^2 = 0,9394. Se observa entonces que este último valor está mucho más cerca de 1, lo que implica que la curva tiene estadísticamente una mejor calidad que la inicial.

Tabla 3. Comparación de algunos Resultados

		PRE Análisis Semántico	POST Análisis Semántico
Total de Grupos	Cantidad Elicitada	132	97
	Tamaño Estimado	138	105
	Omisiones Estimadas	6	8
Grupo 5	Cantidad Elicitada	67	62
	Nivel de Completitud	49%	59%

Si analizamos los valores obtenidos en la segunda estimación respecto de la primera, aún cuando ha crecido levemente la cantidad de omisiones, el nivel de completitud alcanzado individualmente por cada grupo ha mejorado. Como se muestra en la Tabla 3, el Grupo 5 que había elicitado la mayor cantidad de símbolos solo alcanzaba un

49% de completitud, y una vez realizadas las correcciones semánticas ha mejorado al 59%.

4 Observaciones sobre el estudio semántico-estadístico

En base al análisis semántico y posterior estudio estadístico, se puede puntualizar lo siguiente:

- Desde una perspectiva semántica, se puede afirmar que el Grupo 6 observó débilmente al universo, pues del total de 27 símbolos que elicitó, solo 14 símbolos fueron considerados relevantes. Además, identificó 7 símbolos cuyos nombres no existían en el universo de discurso. Es decir, en general tuvo una visión diferente del universo dada una captura de símbolos muy distorsionada frente al resto de los grupos.
- En el otro extremo, el Grupo 5 fue el que identificó más símbolos relevantes. Del total de 67 símbolos identificados por este grupo, solo 5 fueron descartados. Considerando el ajuste semántico, este grupo encontró el 64% de símbolos del total elicitados entre todos los grupos (97), aunque el 59% del total estimado (105).
- Considerando el conjunto de símbolos ajustados semánticamente (97), hubo 9 símbolos elicitados por todos los grupos y 10 símbolos detectados en común por 7 u 8 grupos diferentes. Es decir, hubo casi un 20% de símbolos comunes elicitados entre 7 y 9 grupos, valor que puede considerarse relativamente bajo.
- Se comprobó semánticamente la existencia de 16 símbolos cuyos nombres no pertenecían al universo de discurso, aunque el significado (denotación y connotación) de 7 de ellos era relevante, es decir, dicha información no debería omitirse. Efectivamente, esa información estaba contenida en otros símbolos de otros grupos, por lo tanto ellos no fueron desestimados. Además, debe notarse que había más nombres inexistentes pero tenían sinónimos dentro del grupo, por lo tanto, el nombre inexistente se descartó, sin alterar las estadísticas.
- En resumen, se identificaron en total 35 símbolos no necesarios de un total de 132 símbolos distintos identificados entre todos los grupos. Es decir, se descartó un 26% de los símbolos hallados por los grupos para realizar el estudio estadístico.

Todo esto confirma lo expuesto en la sección 2, en el sentido de que el problema de completitud es mucho más importante que lo que se percibe a primera vista.

5 Conclusiones

En base a trabajos anteriores realizados sobre la estimación del tamaño de modelos escritos en lenguaje natural aplicando un método de captura-recaptura, aún cuando parecía encontrarse un camino promisorio para encarar la incompletitud, el análisis de los resultados estadísticos no parecían ajustarse a la realidad. Partiendo de este análisis previo, se decidió mejorar las estimaciones ajustando los valores experimentales de cada muestra. Es decir, se resolvió determinar correctamente la cantidad de símbolos léxicos elicitados por cada grupo respecto de los otros, pues al manejar elementos

descriptos en lenguaje natural no suele ser tan fácil establecer si dos elementos de dos muestras distintas se refieren al mismo elemento o no.

De los resultados estadísticos presentados en este trabajo, se puede establecer que realizar un análisis semántico hace más confiable las estimaciones de completitud del modelo. Esta mejora en las estimaciones hacen más sólidas las conclusiones del trabajo original, en el sentido que los problemas de completitud son usualmente muy subestimados.

Las 9 muestras del modelo LEL fueron construidas en 1997 cuando la creación del glosario tenía guías básicas a seguir [16]. Dados los resultados, estas muestras no lograron asegurar que los elicitadores tuviesen una visión unificada del universo de discurso. El próximo paso será elaborar más muestras del modelo LEL para el mismo universo de discurso basados en un proceso de creación del LEL afianzado [17]. Esto permitirá establecer si disponer de guías más detalladas colabora en una actividad de elicitación cuyo resultado sea independiente del elicitador, permitiendo alcanzar un grado de completitud mayor. Por otro lado, el análisis semántico realizado en el presente trabajo será estudiado para elaborar nuevas guías en la creación de glosarios o refinar algunas presentes en [17]. Consideramos que, a partir de este trabajo, es necesario modificar el perfil de la heurística de construcción del LEL, la que hasta este momento es notoriamente sintáctica, debiéndosele incorporar más aspectos semánticos.

Una cuestión pendiente es la transferencia de esta experiencia a otros modelos que utilicen el lenguaje natural con estructuras que presenten distintos grados de formalidad.

Adicionalmente, se profundizará el análisis de la influencia del nivel de completitud de un modelo sobre otro modelo derivado de él. Este es el caso de estudiar el nivel de completitud de un conjunto de Escenarios Actuales construidos a partir de información derivada del modelo léxico. También se podría estudiar la completitud del conjunto de Escenarios Futuros partiendo de información proveniente de Escenarios Actuales.

Referencias

1. IEEE Std 830-1998 (R2009), IEEE Recommended Practice for Software Requirements Specifications (ANSI), IEEE, Nueva York, (1998) (R2009)
2. Kotonya, G., Sommerville, I.: Requirements Engineering: Process and Techniques. John Wiley & Sons (1998)
3. Loucopoulos, P., Karakostas, V.: System Requirements Engineering. McGraw-Hill, Londres (1995)
4. Firesmith, D.: Are Your Requirements Complete? Journal of Object Technology, vol. 4, nº 1, pp. 27- 43 (2005)
5. Leite, J.C.S.P.: Gerenciando a Qualidade de Software com Base em Requisitos. Qualidade de Software: Teoria e Prática. Prentice-Hall, Rocha A, Maldonado J, Weber K (eds), cap. 17, pp. 238-246 (2001)
6. Otis DL, Burnham KP, White GC, Anderson DR: Statistical inference from Capture on Closed Animal Populations. Wildlife Monograph, 62 (1978)

7. Wohlin C, Runeson P: Defect content estimations from Review Data. En: 20th International Conference on Software Engineering, pp. 400-409, Japón (1998)
8. Briand, L., El Emam, K., Freimut, B., Laitenberger, O.: A Comprehensive Evaluation of Capture-Recapture Models for Estimating software Defects Contents. IEEE TSE, Vol 26, Nº 6, pp. 518-540 (2000)
9. Petersson, H., Thelin, T., Runeson, P., Wohlin, C.: Capture-Recapture in Software Inspections after 10 Years Research - Theory, Evaluation and Application. The Journal of Software and Systems, vol. 72, pp. 249-264 (2003)
10. Walia, G.S., Carver, J.C.: Evaluation of Capture-Recapture Models for Estimating the Abundance of Naturally Occurring Defects. En: 2nd ACM-IEEE Intl Symposium of Empirical Software Engineering and Measurement, ISBN: 978-1-59593-971-5, pp.158-167, Alemania (2008)
11. Doorn, J.H., Ridao, M.: Completitud de Glosarios: Un Estudio Experimental. En: VI Workshop on Requirements Engineering, pp. 317-328, Brasil (2003)
12. Ridao M, Doorn JH: Estimación de Completitud en Modelos de Requisitos Basados en Lenguaje Natural. En: IX Workshop on Requirements Engineering, pp. 151-158. ISSN: 1413-9014, Brasil (2006)
13. Leite, J.C.S.P., Franco, A.P.M.: A Strategy for Conceptual Model Acquisition. En: IEEE 1st Intl Symposium on Requirements Engineering, IEEE Computer Society Press, pp 243-246, EEUU (1993)
14. Hadad, G,D,S., Litvak, C.S., Doorn, J.H.: Estudio semántico de modelos construidos por elicitadores independientes observando el mismo problema, Proyecto: Completitud de Modelos de Requisitos, Serie Documentos de Trabajo, Nº 279, Departamento de Investigaciones, Universidad de Belgrano, Buenos Aires, ISSN: 1850-2512, 25 páginas (2012)
15. Litvak C.S., Hadad, G.D.S., Doorn, J.H.: Un abordaje al problema de completitud en requisitos de software XVIII Congreso Argentino de Ciencias de la Computación, ISBN: 978-987-1648-34-4, pp. 827-836, Bahía Blanca, Argentina (2012)
16. Hadad, G.D.S, Kaplan, G.N., Oliveros, A., Leite, J.C.S.P.: Integración de escenarios con el léxico extendido del lenguaje en la elicitación de requerimientos: aplicación a un caso real, Revista de Informática Teórica y Aplicada (RITA), ISSN 0103-4308, Vol.6, Nº1, pp.77-103, Porto Alegre, Brasil (1999)
17. Hadad, G.D.S., Doorn, J.H., Kaplan, G.N.: Creating Software System Context Glossaries, en Encyclopedia of Information Science and Technology. Editorial: IGI Global, Mehdi Khosrow-Pour (ed), Information Science Reference, ISBN: 978-1-60566-026-4, 2º edición, Vol. II, pp. 789-794, Hershey, EEUU (2008)

25 years of Requirements Engineering in Brazil: a systematic mapping

Karolyne Oliveira, João Pimentel, Emanuel Santos, Diego Dermeval, Gabriela Guedes, Cleice Souza, Monique Soares, Jaelson Castro, Fernanda Alencar, Carla Silva

Universidade Federal de Pernambuco—UFPE, Recife, PE 50 740-560, Brazil
{kmao, jhcp, ebs, ddmcm, ggs, ctns, mcs4, jbc, ctlls}@cin.ufpe.br, fernanda.ralencar@ufpe.br

Abstract. The celebration of 25th anniversary of the Brazilian Symposium of Software Engineering (SBES) as well as the forthcoming Requirements Engineering Conference to be held in Brazil for the first time, has led us to have a closer look at the local Requirements Engineering (RE) Community. A systematic mapping was performed in order to find out the main Brazilian research groups, authors as well as their topics of interest and publications with greatest impact. This information may be useful for those that do not know well the local requirements engineering community, such as local newcomers or foreign researchers. It may also help to identify potential groups for collaboration. Similarly, it may provide valuable information to assist local agencies when granting research funds.

Keywords: requirements engineering; systematic mapping

1 Introduction

The activities related to software requirements are some of the most important steps in software development, since the requirements describe what will be provided in a software system in order to fulfill the stakeholders' needs. The process of eliciting, analyzing, specifying, validating and maintaining requirements is known as Requirement Engineering (RE).

The RE research community is constantly investigating methods and techniques to tackle and overcome open issues and challenges that may compromise the quality of RE activities [1–3]. The ACM taxonomy has often been used to classify the types of RE work performed [4].

The Brazilian Software Engineering community has recently celebrated its 25th anniversary. It includes several RE groups that have national and international participation in events such as Brazilian Symposium on Software Engineering- SBES, Workshop on Requirements Engineering - WER, IEEE International Requirements Engineering Conference (RE), etc. As an evidence of its growing importance, it was successful in bidding to host the flagship IEEE International Requirements

Engineering Conference which will be held in Rio de Janeiro in 2013. Hence, we felt that it timely to perform an investigation about the contributions made by the Brazilian Requirements Engineering research community to the RE area and their impacts. Thus, this work presents a systematic mapping of the literature related the RE scientific works carried out and published by the Brazilian community during the last 25 years [5]. The purpose of this mapping study is to identify the main research groups, authors and their publications as well as the topics of interest and the scientific works with greatest impact. With the results obtained from this study, a newcomer will be able to identify main groups, key researchers and the work already developed as well as topics that have not deserved much attention by the local community. This kind of information will also be useful for setting up possible collaborative network as well to identify priorities for the allocation of further research funds.

It is worth noting that we started our systematic review examining the main local event of the Brazilian Software Engineering community (i.e. SBES) together with the regional specialized Requirements Engineering event (i.e. WER). The research reported is primarily a statistical analysis of search results for Brazilian papers published in SBES and WER. These events served as the basis for the initial identification of the most productive RE groups. Then, we widened our search using specialized search engines (such as ACM Digital Library, IEEE digital Library, Scopus, Elsevieer) to identify their relevant publications in the RE field, which enabled us to identify the most popular research topics as well as the publications with the greatest impact.

This paper is structured as follows. Section 2 describes our research protocol, including the search strategy and studies selection. In Section 3, we describe the results of our mapping. Section 4 includes the analysis and discussion of the mapping results. Section 5 addresses the threats to the validity of this research' results. Lastly, the paper is concluded in Section 6.

2 Research Protocol

To achieve the purpose of this paper, we performed a systematic mapping of the literature, which provides an overview of the RE research area to assess the amount of existing evidence on a topic of interest and trends to guide future studies [6]. In the sequel we define the research scope; the research planning of the search process; the inclusion and exclusion criteria of the data gathering and its analysis.

The Systematic Mapping protocol has been designed and executed by three pair of researchers and three additional researchers that revised this protocol, conducted the inclusion and exclusion of papers, and discussed the results of the review.

2.1 Scope

The scope of this research is the analysis of the major Brazilian research groups and their published papers in Requirements Engineering area. In order to identify them we

examined both regional and international events. First, we examined the proceedings of the 25 editions of SBES (the most prestigious Brazilian venue for publication of works in Software Engineering) as well as 14 editions of WER (the regional specialized forum on Requirements Engineering). In doing so we will be able to identify the groups that have published most papers at SBES and WER. Later, we will use some search engines (ACM, IEEE, SCOPUS and Elsevier) to investigate publications written in English in other international forums that are indexed by them. Unfortunately, given the complexity of our query the Springer search engine could not be used.

2.2 Research questions

The definition of the research questions is the most important part of a systematic review [6]. Therefore, the research questions that we intend to answer in this Systematic Mapping are the following:

RQ1. *What are the main Brazilian RE research groups, authors and publications published at SBES and WER?*

RQ2. *What are the main research topics published by the Brazilian RE community?*

RQ3. *What are the publications of the Brazilian RE community with the greatest impact?*

2.3 Search process

We performed our systematic mapping in two cycles. In the first cycle we examined the SBES and WER events to find out the requirements engineering contributions of the Brazilian community. Since not all SBES Editions were available in a digital library we had to manually search the proceedings of all 25 editions. We also considered the 14 editions of WER. As a result we identified 07 major Brazilian research groups.

In the second cycle we widened our search using automatic engines to check the publication of these 07 groups in international forums. The following scientific search engines were used: ACM (Advanced Search); IEEE (Advanced Search); Scopus; Elsevier (Science Direct Search). In order to identify and select the studies, we created a search string that was defined in three parts: first, the keywords related to the different RE areas and their variants; second, the term Requirement Engineering, in order to associate the keywords to our area of interest; last, the affiliations related to the major Requirements Engineering research groups in Brazil (according to the results of the first cycle). The string was structured as follows:

(("Analysis" OR "Analyzing" OR "Model" OR "Modeling" OR "Modelling") OR ("Elicitation" OR "Elicit" OR "Eliciting") OR ("Language" OR "Notation") OR ("Management" OR "Evolving" OR "Traceability" OR "Maintaining") OR ("Methodology" OR "Method") OR ("Process") OR ("Specification" OR "Specifying" OR "Communication" OR "Communicating") OR ("Tool") OR ("Validation" OR "Validating" OR "Agreeing" OR "Verification" OR "Verifying")) AND

("Requirements Engineering") AND (Affiliation:"*major Brazilian groups in SBES and WER*")

The specific syntax of this specialized search string was adapted to each digital library previously mentioned. In order to provide more thorough results, we searched each affiliation using its abbreviation, its full name in Portuguese and its full name in English.

In the second cycle, we also manually searched the Springer Requirement Engineering journal (*REJ*) to find works with Brazilian affiliations. This is the most important international journal on Requirements Engineering. Thus it is imperative to consider it when analyzing the impact of researches in this field. The other important international conferences are indexed in the digital libraries used.

2.4 Inclusion and exclusion criteria

As it is usual in systematic mappings, we defined inclusion and exclusion criteria in both cycles in order to guide the selection of the papers that would be considered in our analysis. The inclusion criteria used for the selection of papers in this systematic review were the following:

- The selected papers should be related to Requirements Engineering and contain at least one Brazilian affiliate on its authors list;
- The papers must be available on-line or made available by the author.

On the other hand, the defined exclusion criteria were:

- Documents that are not full papers (e.g., PowerPoint presentations and Short papers);
- Informal research (e.g., Wikis, Blog posts, etc);
- Papers that do not mention Requirements Engineering;
- Duplicated papers.

Three pairs of researchers read the title and abstract and skimmed the paper in order to determine the inclusion or exclusion of the papers listed by the search engines. When there was some divergence, a third researcher would be assigned to make the inclusion/exclusion decision.

2.5 Quality assessment

The assessment of the quality of a research is a quite debatable matter because it is a subjective task, i.e. different reviewers may value different aspects. As systematic reviews and systematic mappings are key tools for enabling evidence-based practice, we have decided assess if each publication had followed basic principles suggested by the Empirical Software Engineering Community [17]. Based on [7] we proposed the

following questions to assess the quality of the individual primary studies as well as the overall strength of the body of evidence used:

- Is the study objective clearly mentioned? (Yes = 1 / No = 0);
- Are the measures used in this study completely defined? (Yes = 1 / Measures are cited, but are not defined = 0,5 / No = 0);
- Are the methods to collect data well described? (Yes = 1 / Methods are cited, but its application is not described = 0,5 / No = 0);
- Are the study results reported in a clear and unambiguous way? (Yes = 1 / No = 0);
- Are the results reported based on evidence? (Yes = 1/ Yes, but not in the paper = 0.5 / No = 0);
- Are the threats to the study validity discussed? (Yes = 1 / Possible threats are cited, but its effects are not discussed = 0,5 / No = 0);

The sum of the answers to each question will provide an indicator (ranging from 0 to 6) of the quality of empirical study. Of course, some other well-known indicators could have been used to assess the quality of the primary study (eg. citation impact that was analyzed separately).

2.6 Data extraction

After the search and the selection processes, we performed a data extraction process by reading each one of the selected papers. In order to guide this data extraction, we adapted a data collection form from Biolchini *et al*. [8], containing the following fields:

- Paper Information: Source; Year; Source Type (Journal or Conference); Brazilian affiliations; Authors list; Title
- Citation (according to the source); Impact Google Scholar; Impact Scopus
- ACM Classification
- Requirements Structure (Models [Standard and Non-Standard]; Natural Language, Textual or Graphical Representation)
- Initial Model (Goal-Oriented Models; Business Models; Stories, Scenarios and Use Case Template; Use Case Diagram; UML; User Interface [Prototype]; Not Specific)
- Method (KAOS; RESCUE; GGBRAM; Any UML-based Method; RUP; CREWS-SAVRE; Not Specific)
- Empirical Study Type (Controlled experiments; Quasi-experiments; Case Study; Survey; Ethnography; Action Research; Illustration; Empty[1])
- Empirical Study Classification
- Context (Industry and Academia)
- Use any tool? (Yes; No)
- Subjective results extraction

[1] We just classified the papers as empirical studies but did not restrict it to this (eg. Surveys do not have empirical validation).

2.7 Data synthesis

In order to facilitate the analysis of the research questions the data collected were tabulated to show:

- The identifier assigned to the study, its authors, affiliations, source and year of publication;
- The amount of publications of each author (concerning RQ1);
- The classification of the study following the taxonomy proposed (concerning RQ2);
- The citations numbers and the quality score of each published work (concerning RQ3).

3 Results

This section presents the results of this systematic mapping in which each criterion is commented.

3.1 Search results

The first part of this research was the mapping of all Brazilian studies published since the first edition of SBES (1987) and WER (1998). After a careful analysis of all papers from these events, the number of works selected in this process was 63 papers at SBES and 154 at WER.

As a result we obtained the top five Brazilian institutions with more RE papers published at each event (see Table 1). Merging the results from both events, we found the major RE Brazilian groups in terms of number of publications are the following: Universidade Federal da Paraíba (UFPE), Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), Universidade Metodista de Piracicaba (UNIMEP), Univesidade Estadual do Rio de Janeiro (UERJ), Universidade Federal de São Carlos (UFSCar), Universidade Federal do Rio Grande do Sul (UFRGS), Universidade Federal do Rio de Janeiro (UFRJ).

Table 1. The top Brazilian affiliations at SBES and WER by number of publications.

SBES		WER	
Affiliation	Publications	Affiliation	Publications
UFPE	22	UFPE	45
PUC-Rio	11	PUC-Rio	33
UFRGS	6	UNIMEP	11
UFSCar	5	UERJ	9
UFRJ	4	UFSCar	8
Total	48	Total	106

In the second part of our research, we used the digital libraries engines to broaden the search for RE publications of those 7 Brazilian institutions. We also conducted a manual search of Requirements Engineering journal (published by Springer) to identify papers of authored by researchers of those Brazilian institutions.

The number of papers found on each scientific search engine, plus the Requirements Engineering journal is summarized in Table 2.It also presents the number of papers marked as candidates and those ultimately selected according to the inclusion and exclusion criteria.

Table 2. Number of studies found, candidate and selected studies, by source.

Source	Studies found	Candidate studies	Selected studies
ACM	49	22	19
IEEE	125	13	13
SCOPUS	7	3	3
Science Direct	13	6	6
REJ	9	9	9
Total	203	53	50

3.2 Synthesis of the findings

The findings of this mapping were separated in two main groups: first we discuss the SBES and WER studies, related to RQ1; then, the proposals published in other venues, found in automatic and manual search, that answer RQ2 and RQ3.

3.2.1 What are the main Brazilian RE research groups, authors and publications published at SBES and WER? (Answer to RQ1)

The distribution of papers per year can be seen in Fig. 1. Considering the Brazilian RE studies representation, we identified that at SBES and WER these studies had an average participation rate of 12% and 60%. Note that in 1998 the WER (Workshop on Requirements Engineering) was established with the specific goal to become a forum for the Ibero-American Requirements Engineering community.

In Table 1, we presented the top five Brazilian RE research groups at each event. They represent approximately 77% of the total of Brazilian RE papers published at SBES and 70% at WER. This result suggests that a substantial amount of RE Brazilian papers comes from these institutions. Of these total, UFPE and PUC-Rio are the most important groups and together are responsible for around 70% of the Brazilian RE papers at SBES and 73% at WER.

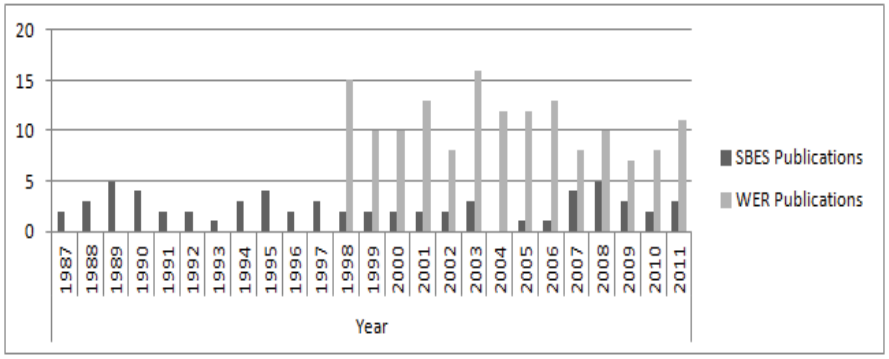


Fig. 1. Number of SBES and WER publications per years

In Table 3 we present the top five authors by number of RE publications at SBES and WER. These researchers are recognized as leaders in the RE community with a history of project management and supervision that explains their participation in several papers.

Table 3 Top RE Brazilian Authors by number of publications at SBES and WER

SBES			WER		
Author	Organization	Number of publications	Author	Organization	Number of publications
Jaelson Castro	UFPE	11	Jaelson Castro	UFPE	38
Julio Leite	PUC-Rio	9	Julio Leite	PUC-Rio	31
Silvio Meira	UFPE	7	Fernanda Alencar	UFPE	13
Fernanda Alencar	UFPE	4	Luiz Martins	UNIMEP	9
Paulo Cunha	UFPE	3	Carla Silva	UFPE/UFPB	8
Rosângela Penteado	UFSCar	3	Vera Werneck	UERJ	8

There are two RE publications at SBES that have more than 20 citations [9] [10]. The paper titled "CRE: A Systematic Method for COTS Components Selection" [9] has more than 90 citations and is the second most cited SBES paper considering all SBES editions. Regarding WER, there are four publications with more than 20 citations [11] [12] [13] [14]. The study "From Early Requirements Modeled by the i* Technique to Later Requirements Modeled in Precise UML" [11], with 35 citation, is the second most cited WER paper considering all WER editions.

3.2.2 What are the main research topics published by Brazilian RE community? (Answer to RQ2)

After the mapping of the RE area published at SBES and WER, we searched the digital libraries and the Requirements Engineering journal to obtain the internationals venues that have had Brazilian RE publications. These selected papers are listed in [18], which presents the identification and ACM Classification of each selected study. The studies were assessed in order to catalog the main research topics in an international scenario. The result is summarized in Table 4.

The results for the ACM Classification suggest that the majority of works (38%) is focused on Methodologies, followed by Specification with 14%. Together, they account for 52% of the total. The other topics correspond at most to 10% of the selected papers. We analyzed these topics by combining them as follows: (i) Methodologies and Specification (52 %); Elicitations methods, management and process (30%); and Analysis, tools, assessment, frameworks software architectures and validation (18%). In the following subsections we will discuss the results on these topics.

Table 4. Number of studies by ACM Classification

ACM Classification	Number of publications	Percentage
Methodologies	19	38%
Specification	7	14%
Elicitation methods	5	10%
Management	5	10%
Process	5	10%
Analysis	3	6%
Tools	2	4%
Assessment	1	2%
Frameworks	1	2%
Software Architectures	1	2%
Validation	1	2%

Methodologies and specification

The majority of the Brazilian RE groups have been working mainly in areas related to Methodologies (19 publications) and Specification (7 publications), corresponding to 26 of all selected studies. Analyzing Methodologies area, we observed some interesting points. According to the Requirements Structure criterion, 63% of the studies are associated to Models and 37% to Natural Languages. Considering the type of models, the most used Initial Models in these methodologies are Goal-Oriented Models with 42% of the total. As Empirical Study Types, Illustrations and Case Studies are the most used. The Validation Context of these studies is mainly

Academics (89%). With respect to the use of tools, we realized that 63% of the papers in Methodologies area were not supported by tools, i.e. only 37% indicated the availability of some tool.

Some important works rely on Goal-Oriented Models and more specifically i* (iStar) approaches. In one of them, the authors present a methodology for information systems development based on i* modeling language. The authors of another work argued that the Use Cases development can be improved by using i* organizational models. Another study describes a set of guidelines for the integration of early and late requirements specifications. They proposed the GOOD tool (Goals into Object Oriented Development) that consists of an extension of the Rational Rose tool which integrates with Organization Modeling Environment (OME), an i* supporting tool. There are also three more works related to i*. One important publication related to models proposes a systematic approach to assure that conceptual models will reflect the elicited NFRs. The authors use OORNF tool and work on a UML based approach.

There are 7 studies in Specification area. One study is a survey and as such fell outside the Requirements Structure criteria, the others are considered either as Models (43%) or Natural Languages (43%). Considering these 6 studies, the most used Initial Models are Stories, Scenarios and Use Case Template while 1 study is based on Business Models. Regarding Study Types, the majority of the works rely on Illustrations and Case studies (72%) with few (28%) using other types. On the other hand, the Validation Context of these studies is Academic. With respect to the use of tools, none of the Specification studies benefited from tool support.

In the Specification area, one important paper describes a method to improve requirements specification using scenario and the LEL. Another study presents a way for eliciting and describing business rules and states and how they are related to requirements. A third study describes an approach to support use case scenario variability management, enabling separation of concerns between languages used to manage variability and languages used to specify use case scenarios. All of them are related to Stories, Scenarios and Use Case Template unlike the Methodologies area that strongly uses Goal-Oriented Models

Elicitations methods, management and process

Observing the studies in Elicitation methods area, we identified that 60% of them rely on modeling techniques for requirements specification; the other studies (40%) use natural language or textual/graphical representation in their elicitation methods. With regard to modeling techniques we can highlight two studies that use traditional models in elicitation: one paper uses business models and the other applies scenarios in its requirements elicitation. Another study uses a non-traditional modeling technique applied to some kind of formalization in its elicitation method. Within the studies that use natural language or textual or graphical representation, we highlight the use of the viewpoint concept in one study and the use of studies of the future methods in order to anticipate requirements elicitation in other paper.

It is also important to note that 60% of the studies in Elicitation Methods area are validated through illustrative examples. Besides that, we found one paper with empirical

evaluation based on a case study and other paper that performed a controlled experiment. Another relevant aspect of the studies identified is that 100% of the studies were conducted in an academic context. With respect to the use of tools, we identified that 60% of the studies in requirements elicitation area propose some tool support.

Considering the management area, we identified 5 publications, which represent 10% of the total number of selected studies. Analyzing these studies we can highlight some interesting points. The use of modeling techniques constitute 40% of the studies in this area, 20% of the studies rely on natural language or textual/graphical representation and 40% of the studies do not use any kind of requirements structure. Both studies that use models apply Goal Oriented techniques. Moreover, the studies that do not use requirements structure have different characteristics, one study is a survey and the other is a position paper that indicates some challenges in mobile game requirements management.

The distribution of empirical studies types in the requirements management area is: 2 papers with case studies, 1 survey, 1 lab demo and 1 illustrative example. Another interesting result observed in this area is their distribution in academic or industrial context. The academic studies constitute 80% of the publications while only 20% of the studies are performed in industrial context. With respect to automation, we identified two studies that provide some kind of tool support to their studies; the first tool extends a traditional goal-oriented tool (OpenOME) and the second presents the Scenario Evolution Tool (SET) to support the framework proposed in the study.

Regarding the studies of the requirements process area, the use of modeling techniques appears in 60% of the studies while the use of natural language and textual/graphic representation appears in 40% of the studies. Note that all papers that present some kind of modeling technique in their requirements process proposal use goal-oriented models, in particular the i* framework. With respect to the studies that include natural language or textual/graphic representation, one study proposes a scenario construction process and the other one uses natural language.

Concerning the classification of the empirical studies, we noticed that case studies are used in 60% of the papers in the requirements process area. In the remaining 40%, one paper is an Action Research and the other paper uses an illustrative example. Among the studies of this area, we can highlight that 80% of them were conducted in an academic context. The only one that was performed in an industry setting presents an experience during a technology transfer project to improve the requirements engineering process in market-driven companies. Finally, there is no tool support for the studies of this area.

Analysis, tools, assessment, frameworks software architectures and validation

There are 9 studies in this group, corresponding to 18% of the total. Analyzing the requirements structure, 5 of them use textual requirements (with natural or controlled language), while 3 use models; the last one consists of a test-bed, so this aspect can-not be evaluated. Considering the initial type of model used 33% relied on goal-oriented models, 11% depended on scenarios and the rest of them did not use a specific model.

From the 9 studies in this group only one was conducted in the industry scope, the other 8 were under an academic scope. As for the empirical study type, case studies are used by 3 studies, while 1 work used survey and another used action research. However, 2 studies preferred to use only an illustrative example and 2 other presented their proposals without corroborating their findings with any empirical evidence or illustrating them with an example.

As for the tool support, three studies of this group provide it: one study was classified as framework, other one was classified as validation and the last one was classified as tools. The other study under tools classification is actually a systematic review about tools, so it does not provide tool support for any approach; it analyzes the existing tools instead.

3.2.3 What are the publications of the Brazilian RE community with the greatest impact? (Answer to RQ3)

In the second part of our systematic mapping, we performed automated search on the main scientific search engines, considering the following Brazilian universities: UFPE, PUC-Rio, UNIMEP, UERJ, UFSCar, UFRGS, UFRJ. We also performed a manual search for papers that appeared at the Requirements Engineering journal published by Springer. This search resulted in a total of 203 papers. After applying the inclusion and exclusion criteria (see Table 2), we selected 50 papers for analysis. The results comprise 16 years of research publications, from 1996 to 2011. The distribution of these papers throughout the years is presented in Fig. 2. This distribution shows a slight increase of the number of publications in the last years.

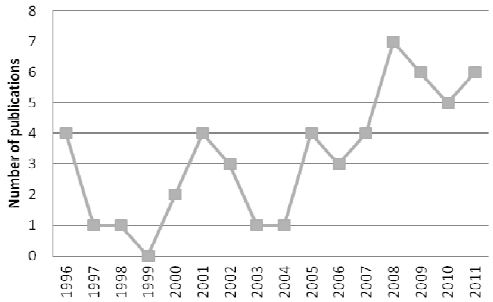


Fig. 2. Papers published by year

Table 5 shows the most cited papers found on our survey, based on the Google Scholar citation count. These papers represent the Requirements Engineering papers from the Brazilian community that had the greatest research impact, considering citation count. The citations of these ten papers amount to 74,77% of the total number of citations from the 50 selected papers.

Table 5. Most cited papers

Year	Title	# Citations
2002	Towards requirements-driven information systems engineering: the Tropos project	540
1997	Enhancing a requirements baseline with scenarios	186
2000	A Scenario Construction Process	134
2002	Requirements Engineering and Technology Transfer: Obstacles, Incentives and Improvement Agenda	90
2002	Deriving use cases from organizational modeling	84
2001	A Framework for Integrating Non-Functional Requirements into Conceptual Models	83
1998	Business Rules as Organizational Policies	47
2001	Using UML to reflect non-functional requirements	46
2008	Configuring features with stakeholder goals	36
2001	A framework for building non-functional software architectures	34

Regarding the publication venues of the 50 selected papers, we find the following distribution: 43,4% on international conferences; 35,85% on international journals; and 20,75% on international workshops or Brazilian symposiums. Regarding conference papers, 34,78% of them were published on the Requirements Engineering Conference. From the journal papers, 47,37% of them were published in the Requirements Engineering journal. These numbers suggest that the Brazilian papers have a small but already reasonable impact on the Requirements Engineering research.

Our quality assessment (Section 2.5), resulted in an average score of 2,55 out of 6,0. Given that in this paper we equated the quality to the use of sound Software Engineering Principles, we reach a conclusion that the Brazilian RE community, may be neglecting some Empirical Software Engineering principles. Nonetheless, we found no correlation between the quality score and the impact factor of these papers.

4 Discussion

As expected, there is a clear relationship between the organizations found in Table 1 and the authors found in Table 3. The leadership of these authors certainly reflects the consolidation of local groups of research and thus put their organizations in a prominent position. Some authors have presented pertinent contribution for both SBES and WER events showing their importance for the Brazilian RE community. For both conferences, the most cited publications were authored by the author with more publications selected in the respective conferences. This revels that the authors with highest number of publications also contributes with the more relevant ones.

The two major research topics identified in the selected studies were Methodologies and Specification. It reveals the preference of Brazilian researchers for these topics. In addition, the two more cited publications were grouped in these two topics. The first more cited publication and second one refer to Methodologies and

Specification, respectively. This does not mean that remaining topics are not researched; in fact all the topics cited in Table 4 have been represented at least by one publication. Hence, there is a wide thematic covered by the Brazilian RE research groups.

In general, there is no significant difference in the use of models or natural language to structure requirements. The publications point for both types of structure has the same level of relevance. On the other hand, when considering the use of CASE tools to support RE activities, no such balance is observed. About 68% of the studies are not supported by tools. When considering tools as the main topic of a paper, the difference is even higher with only a small percentage of the total papers (see Table 5). Since the use of tools has become essential to deal with complex cases, the development of CASE tools could be better addressed by this community.

Majority of the works are illustrative examples and are focused in the academic context. This tendency can be explained by several reasons. For instance, the innovative contribution for the state of art could be not mature enough to be applied in industrial environments. The descriptive nature of the studies also could be a reason to the small presence of controlled experiments. Nonetheless, it could be justified by the fact that the Brazilian RE community has recently started to adopt principles of Empirical Software Engineering.

Regarding the impact of Brazilian research on the Requirements Engineering field, it is important to note that this is not an extensive systematic mapping and, therefore, it does not represent the whole set of contributions generated by this community. We only considered the five top publishing Brazilian universities at SBES and WER, which resulted in a total of seven distinct universities: UFPE, PUC-Rio, UNIMEP, UERJ, UFSCar, UFRGS, UFRJ. Moreover, it is possible that some RE papers published by these universities have not been considered. This is the case because the 'search by affiliation' feature on the search engines is not completely accurate. In particular, we can observe that the list of results only starts at 1996, even though we are aware of works published on international venues at least as far as 1989. This kind of limitation is expected on systematic mappings that make use of automatic search mechanisms, as it is our case.

Nonetheless, we can draw some comments considering the sample we obtained, as follows. The chart on Fig. 2 shows a slight increasing trend on the number of RE papers that are published. This is a positive indicator, showing that the RE community in Brazil is growing stronger. Moreover, there is a reasonable amount of publications at the most important Requirements Engineering venues (the Requirements Engineering journal and the Requirements Engineering Conference). Thus, we may say that the research produced by the Brazilian community is not confined to Brazil, i.e., it has a worldwide reach.

From the total of 50 papers analyzed in cycle two, the top 10 cited papers amount for 74,77% of the total number of citations. This shows an uneven distribution of the impact of these papers. i.e., some papers have a very strong impact, while several papers have a weak impact. Moreover, all the top four most cited papers, which amount to 55,49% of the total number of citations, were published on journals. This indicates that papers published on this kind of venue are more likely to have a strong

impact. Thus, publishing more papers on high quality journals may be a good way of increasing the impact of the Brazilian RE community.

The findings also indicate (see section 3.2.3) that Empirical Software Engineering principles are not fully used by the RE Brazilian community.

5 Threats to Validity

We have validated our review protocol to ensure that the research was as correct, complete, and objective as possible. We identified possible limitations of this study in two moments of the review process: in the publication selection and in data extraction.

The search for publications was performed in two cycle but in three major steps: (i) manual search of the studies in the proceedings of SBES and WER; (ii) automatic search considering the major Brazilian research groups in four digital libraries (ACM, IEEEXplore, Scopus and Science Direct); and (iii) manual search in the Requirement Engineering journal to find works with Brazilian affiliations. In the first step, we did not have problems to find the papers of all editions of SBES and WER. The main limitations regarding the publication selection occurs in steps (ii) and (iii).

In step (ii), the search string associates the different RE areas (and synonyms) to the term "Requirements Engineering", which may constitute a limitation because studies that only refer to specific techniques in RE are not considered in the mapping. In addition to that, the synonyms of RE areas may be insufficient to capture all studies in each area. With regard to the automatic search in the digital libraries, there is a limitation in the review because we could not use our search string in Springerlink library, which possibly leads to a reduction in the considered studies. However, we used SCOPUS that is a large database that includes springer papers.

In the step (iii), we identified a limitation concerning the journal papers included in the mapping. We only performed the manual search in the Requirements Engineering Journal (published by Springer). We took for granted that the other RE studies published in other journals would have been captured through the automatic search performed in the previous step. However, given the acknowledged limitations of the search engines we cannot guarantee that all RE Brazilian papers published in other Software Engineering journals are included in the systematic mapping. Besides that, we did not consider grey literature (e.g. industrial papers, PhD thesis and books) or unpublished results.

With regard to the data extraction, the data collection spreadsheet was based on two papers [15] [16]. However, we could have explored a broader set of data in order to investigate other aspects of the papers included in the review. Moreover, it is possible that some kind of inaccuracy or misclassification may have occurred in the data extraction performed in this systematic mapping.

6 Conclusion

We have conducted a systematic mapping to find out (i) the most productive RE Brazilian research groups at SBES and WER, (ii) the main research topics published by these groups, and (iii) the publications of the Brazilian community of

Requirements Engineering with the greatest impact in the international community, according to their number of citation.

The information provided in this paper may be useful in different contexts. For example, a newcomer (eg. new research student) will be able to identify main groups, key researchers and the work already developed. Moreover, topics that have not deserved much attention by the local community may be identified and become the subject of new research projects. This kind of information will also be useful for setting up possible collaborative networks as well to identify priorities for the allocation of further research funds. The local industrial community may also benefit as they may be able to identify experts and groups that could help them if a specific RE need arises.

The systematic mapping was divided in two cycles: the first consisted of a manual search of the SBES and WER proceedings; the second cycle included an automatic search of four major digital libraries and a manual search on editions of the Requirements Engineering journal. The main goal of the first cycle was to answer the research question RQ1 and to answer RQ2 partially. The second cycle was used to answer RQ2 and RQ3. As a result, 177 papers were selected and assessed from an initial set of 367. Our findings allowed us to answer the three proposed research questions.

With the data extracted from the select studies, we were able to discover the Brazilian researchers that have extensively published at SBES and WER and relate them to the leadership of RE research groups in Brazil. With regard to the research topics, about 38% of the examined studies were about methodologies, either presenting new ones or improvements of existing ones. Moreover our findings reveal the better empirical validation maybe required.

This work is primarily a statistical analysis of search results identified in our SMS. It would be interesting in the future also to discuss the type of research conducted by the Brazilian research groups. We also plan to address some the weakness found as well as the topics that deserve further studies. Likewise we would like to examine the extend that the Brazilian institutions have contributed to the RE progress.

References

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). 1, (2010).
2. Nuseibeh, B., Easterbrook, S.: Requirements engineering: a roadmap. Proceedings of the Conference on The Future of Software Engineering. pp. 35–46. ACM, New York, NY, USA (2000).
3. Zave, P.: Classification of research efforts in requirements engineering, http://portal.acm.org/citation.cfm?doid=267580.267581, (1995).
4. ACM: The ACM computing Classification, http://www.computer.org/portal/web/publications/acmtaxonomy.
5. Brereton, P., Kitchenham, B.A., Budgen, D., Turner, M., Khalil, M.: Lessons from applying the systematic literature review process within the software engineering domain. J. Syst. Softw. 80, 571–583 (2007).

6.
7. Kitchenham, B., Charters, S.: Guidelines for performing Systematic Literature Reviews in Software Engineering. Engineering. 2, 1051 (2007).
8. Dyba, T., Dingsoyr, T.: Empirical studies of agile software development: A systematic review. Information and Software Technology. 50, 833–859 (2008).
9. Biolchini, J., Mian, P.G., Candida, A., Natali, C.: Systematic Review in Software Engineering. Engineering. 679, 165–176 (2005).
10. Alves, C., Castro, J.: CRE : A Systematic Method for COTS Components Selection. XV Simpósio Brasileiro de Engenharia de Software. pp. 193–207. , Rio de Janeiro (2001).
11. De Sousa, G.M.C., Da Silva, I.G.L., De Castro, J.B.: Adapting the NFR Framework to Aspect-Oriented Requirements Engineering. XVII Brazilian Symposium on Software Engineering. SBC Press (2003).
12. Alencar, F., Castro, J.: From early requirements modeled by the i* technique to later requirements modeled in precise UML. Anais do III Workshop em …. 92–109 (2000).
13. Pinheiro, F.: Formal and informal aspects of requirements tracing. Workshop of Requirements Engineering (WER) (2000).
14. Togneri, D.F., Falbo, R.D.A., Menezes, C.S. De: Supporting Cooperative Requirements Engineering with an Automated Tool. Workshop of Requirements Engineering (WER). , Valencia, Spain (2002).
15. Iribarne, L., Vallecillo, A., Alves, C., Castro, J.: A Non-Functional Approach for COTS Components. Workshop of Requirements Engineering (WER). pp. 124–138 (2001).
16. Loniewski, G., Insfran, E., Abrahão, S.: A Systematic Review of the Use of Requirements Engineering Techniques in Model-Driven Development. Model Driven Engineering Languages and Systems. 6395 LNCS, 213–227 (2010).
17. Nicolás, J., Toval, A.: On the generation of requirements specifications from software engineering models: A systematic literature review. Information and Software Technology. 51, 1291–1307 (2009).
18. Dyba, T., Dingsoyr, T.: Strength of evidence in systematic reviews in software engineering. Symposium on Empirical software engineering. 7465, 178-187 (2008).
19. Oliveira, K., Pimentel, J., Santos, E., Dermeval, D., Guedes, G., Souza, C., Soares, M., Castro, J., Alencar, F., Silva, C.: Technical Report of 25 years of Requirements Engineering in Brazil: A systematic mapping. (2012). Available in http://www.scribd.com/doc/117508494

Retrospective and Trends in Requirement Engineering through WER

Joselaine Valaski, Wilian Stancke, Sheila Reinehr, Andreia Malucelli

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

Abstract. This work refers to the review of 258 papers published in the WER throughout 15 editions. This review´s goal was to identify the most active research groups within this workshop, the most debated topics and the trends in the Requirements Engineering area. The results showed that Brazil, Argentina and Spain hold the most active groups. Moreover, the results pointed out the requirements modeling as one of the most discussed topics in this event.

Keywords. Retrospective, Trends, Requirements Engineering, WER

1 Introduction

The Requirements identification is an extremely important activity, since it is the basis for the planning, development follow up and acceptance of the software project results [1]. In order to support these results, Requirements Engineering provides appropriate mechanisms to understand the client´s needs, analyzing necessities, checking feasibilities, negotiating a reasonable solution, specifying a solution without ambiguities, validating a specification and managing the needs as they are transformed into a system [2].

Some of the main activities related to Requirements Engineering are as follows: elicitation, analysis, specification, validation and requirements management [3]. These activities main goal is to support understanding and to formalize the client´s main needs in a way that it decreases the problems throughout the software development.

However, despite the advances in the Requirements Engineering area, as software become much more complex and bigger, new problems emerge and new solutions are proposed [4]. Thus, it is important to follow the evolution of the topics related to this area. Some mappings and systematic reviews have been performed in order to provide a better idea of what has been produced in this area. These works have offered a better vision for specific topics from the Requirements Engineering area, such as elicitation techniques [5] [6], specification techniques [7] and requirements writing standards [8].

However, it is also important to consider providing a broad vision of the main discussed topics among research groups, as well as identifying these groups. This infor-

mation may bring several benefits to the scientific area, such as the identification of new information sources, establishment of new partnerships and the orientation of researches toward more relevant topics at a given time. The scientific works are a quite important source to find this information. The Requirements Engineering area has several means of publishing these works, such as specialized journals and international events.

In this context, this work had the goal of identifying some of this information and as an initial source of research, the Workshop on Requirements Engineering (WER) was chosen. The WER is a workshop that has been taking place since 1998 with the goal of consolidating the Iberoamerican Requirements Engineering research community. The event has predominantly happened in Argentina and Brazil. However, in the last few years, other countries from South, North America and Europe have hosted this event as seen in Table 1. Throughout 15 editions of the WER, 258 papers have been published. The published topics discuss issues involving the main activities of Requirements Engineering. The WER is considered a mature and consolidated event in the Requirements Engineering area and for this reason was considered a feasible source to obtain relevant answers in this area.

Table 1. Host cities of WER

Year	Country/City
1998	Brazil/Maringá
1999	Argentina/Buenos Aires
2000	Brazil/Rio de Janeiro
2001	Argentina/Buenos Aires
2002	Spain/Valência
2003	Brazil/Piracicaba
2004	Argentina/Tandil
2005	Portugal/Porto
2006	Brazil/Rio de Janeiro
2007	Canada/Toronto
2008	Spain/Barcelona
2009	Chile/Valparaiso
2010	Ecuador/Cuenca
2011	Brazil/Rio de Janeiro
2012	Argentina/Buenos Aires
2013	Uruguay/Montevideo

Section 2 describes the method used in this research, followed by Section 3, which details and discusses the results. Finally Section 4 presents the final considerations of this work.

2 Method

This section describes the method used in this research, including the questions and the steps performed in order to make the extractions and classification of the analyzed papers.

2.1 Development of the research questions

With the goal of identifying the most active countries and institutions at the WER, as well as the most discussed topics and trends in Requirements Engineering, the following questions have been made:

1. What are the main countries that published in WER?
2. What are the main institutions that published in WER?
3. What are the main topics discussed in WER?
4. Which institutions have been discussed the main topics?
5. Which topics have been discussed for the main institutions?
6. What are the trends in Requirements Engineering?

2.2 Extraction of information

In order to answer the previous questions, each one of the 258 papers has been analyzed with the goal of extracting the following information: institutions involved in research, home country of the institution and main discussed topics. The access to the papers was made through the website http://wer.inf.puc-rio.br.

The institution identification and home country was manually made in each one of the papers. More than one institution could have been related to the article through their researchers, including the ones from different countries.

In order to define the topics to be classified, first was analyzed the classification used in the previous WER´s editions. However, it was noted that in back 1998, 2000, 2002, 2006, 2007 and 2012, no classification was proposed by the event. Moreover, it was also noted that the topics had a variation in their nomenclature throughout the years, as well as the unification of topics, as it can be seen in Table 2. Considering these limitations, it has been decided to propose a classification based on the classifications already made unifying or subdividing determined topics. The result of this new classification is presented in Table 3.

After the topics classification definition, two students (one master and one doctor) read the papers with the goal of identifying the main discussed topics. It was defined that the paper must be related to at least one topic presented in Table 3 but new topics not listed in Table 3 could be included and related. The reading and identification of the topics was performed in an independent way between the students. For some of the papers, the reading of the abstracts was enough in order to identify their topics. In other cases, the papers had to be read thoroughly. After the individualized classification was performed, the students together made the consolidation of issues where there was divergence of classification.

Table 2. Topics of the WER

Year	Topics						
1999	Multidisc. approach and Education	Analysis	Modeling and Represent. Requirem.	Negotiation and Requirem. elicitation	Process e Management Requirem.		
2001	Requirements elicitation	Requirements modeling	Process and req. management	Quality Requirements	Non-functional requirements	Requirements reuse	
2003	Requirements elicitation	Specification and Requirements modeling	Requirements management and Experimental studies	Process, model, methods and tools	Quality requirements and Quality Assessment		
2004	Requirements elicitation	Experimental studies	Requirements management	Modeling and Specification	Process, Methods and Tools	Quality requirements	Requirements for agent-oriented paradigm
2005	Cognitive approaches and Reuse	Agents and objectives	Analysis and Modeling	Aspects and Composition	Requirements elicitation	Process	Quality and Metrics
2008	Analysis	Elicitation /Empirical Studies	Elicitation /Methodologies and Tools	Modeling	Process and Quality	Reuse and Traceability	
2009	Early requirements	Requirements elicitation and Management	Requirements specification and Management	Later Requirements and architecture	Traceability and Product Lines		
2010	Requirements elicitation	Req.specification and Modeling	Req. management and Traceability	Business process mod. and P. Families			
2011	Aligning req. with business objectives and process	Early requirements	Late requirements	Non-functional requirements	Reuse of requirements	Models transformation	

Table 3. Topics proposed for classification

Requirements Engineering topics
Cognitive approaches, educational and knowledge management
Requirements analysis and Requirements negotiating
Requirements elicitation
Requirements specification
Tools
Requirements management
Measurement/Metrics
Requirements modeling
Process/Method
Quality requirements
Requirements traceability
Non-functional requirements
Agent-oriented paradigm
Reuse
Requirements Validation

The obtained results after the papers' classification are presented and discussed below.

3 Results and Discussion

The results will be presented and organized according to the research questions presented in Section 3.

3.1 What are the main countries that published in WER?

In all, 20 countries had at least one publication at WER through 121 institutions. In Table 4 it is shown the amount of institutions involved, the number of published papers and the percentage of publications considering the total amount of papers published up to 2012.

As it has been pointed out in Table 4, the countries with the most publications at WER are Brazil, Argentina, Spain and Canada. Brazil has had an 80% participation ratio, as far as publication is concerned. Once the identification of the most active countries was done, this was also applied to the institutions (of education or not) with the most publications at WER.

Table 4. Numbers of papers per country

Country	Number of institutions	Number of papers	Percentage (of 258 papers)
Brazil	52	208	80.62%
Argentina	14	61	23.64%
Spain	13	60	23.26%
Canada	10	29	11.24%
Mexico	2	7	2.71%
Chile	6	6	2.33%
Portugal	3	5	1.94%
Italy	4	5	1.94%
United Kingdom	4	4	1.55%
Netherlands	1	4	1.55%
Switzerland	1	3	1,16%
Malaysia	1	2	0,78%
Venezuela	1	2	0,78%
Cuba	2	2	0,78%
China	2	2	0,78%
Costa Rica	1	1	0,39%
Ecuador	1	1	0,39%
Finland	1	1	0,39%
Sweden	1	1	0,39%
United States	1	1	0,39%

3.2 What are the main institutions that published in WER?

As it has already been mentioned, 121 institutions have had at least one publication at WER. However, due to space limitation, in Table 5 only the institutions that have had at least 10 publications since the beginning of the WER (15 years ago) are presented.

In this same table, the home country, amount of published papers and publishing ration for each institution are shown, considering the total of published papers up to 2012.

The institutions with the most number of publications, due to logical reasons, are located in the countries pointed out in Table 4. Among the 9 institutions with the most number of publications, 4 of them are located in Brazil, 2 in Argentina, 2 in Spain and 1 in Canada.

After the identification of the most active countries and institution, a verification of the most discussed topics and their relation with the research groups was performed.

Table 5. Number of papers per institution

Institution	Country	Number of papers	Percentage (of 258 papers)
Universidade Federal de Pernambuco	Brazil	44	17%
Pontifícia Universidade Católica do Rio de Janeiro	Brazil	35	14%
Universidad Politécnica de Valencia	Spain	24	9%
Universidad Nacional del Centro de la Provincia de Buenos Aires	Argentina	16	6%
Universidad Nacional de La Plata	Argentina	14	5%
Universidade Metodista de Piracicaba	Brazil	12	5%
Universidade Estadual do Rio de Janeiro	Brazil	11	4%
York University	Canada	11	4%
Universitat Politécnica de Catalunya	Spain	10	4%

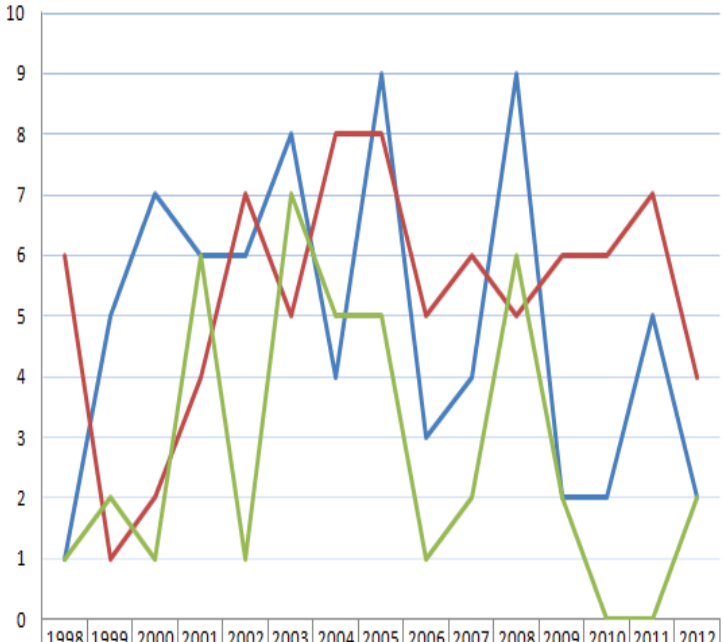
3.3 What are the main topics discussed in WER?

In Table 6 it is shown for each one of the topics defined in Table 3, the amount of papers that were related to the topic and the corresponding percentage in an array of 258 papers.

Table 6. The most discussed topics in WER

Topic	Number of papers	Percentage (of 258 papers)
Requirements modeling	80	31%
Requirements elicitation	73	28%
Process/Method	41	16%
Requirements management	35	14%
Requirements specification	32	12%
Tools	32	12%
Quality requirements	26	10%
Reuse	25	10%
Non-functional requirements	25	10%
Measurement/Metrics	16	6%
Agent-oriented paradigm	12	5%
Requirements traceability	11	4%
Cognitive approaches, educational and knowledge management	10	4%
Requirements analysis and Requirements negotiating	6	2%
Requirements validation	6	2%

As shown in Table 6, the 3 most related topics were: Requirements modeling, Requirements elicitation and Process/Method. In order to have a better idea of the intensity that these topics were discussed throughout the editions, the illustrative graphic in Fig. 1 was used.



	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012
Requirements elicitation	1	5	7	6	6	8	4	9	3	4	9	2	2	5	2
Requirements modeling	6	1	2	4	7	5	8	8	5	6	5	6	6	7	4
Process/Method	1	2	1	6	1	7	5	5	1	2	6	2	0	0	2

Fig. 1. Number of papers related to the topic per year

Through this graphic it is possible to observe that the topic Requirements elicitation was highlighted up to 2008, with some peaks. As far as the topic Requirements modeling, since 2009, it started to be the more discussed than the topic Requirements elicitation. It´s important to observe that in general is difficult to separate the both terms, they are much related. Usually the elicitation activity is supported by the modeling activity and vice-versa. However the modeling activity in the last years may has been seen as the main activity in this relationship.

Process/Method is a topic that has had many peaks and starting on 2009, it presented a significant reduction in the number of published papers. This oscillation may be related to the period of the creation and dissemination of quality maturity models.

As it has been previously mentioned, the topic relation was based in Table 3, but the students who made the classification could also identify other topics that they judged relevant in the article. These other topics are presented in Table 7.

Table 7. Other relevant topics discussed in WER

Topic	Number of papers	Percentage (of 258 papers)
i*	26	10%
Models transformation	22	9%
Oriented goals (GORE)	18	7%
Scenarios	15	6%
Business modeling	15	6%
LEL/LAL	12	5%
Meta model, ontology and taxonomy	11	4%
Natural language	10	4%
Tropos	9	3%
Distributed development	8	3%
Model driven development	8	3%
Patterns	8	3%
Organizational modeling	8	3%
Inspection	7	3%
Product lines	6	2%
Variability	6	2%
Verification	6	2%
NFR Framework	5	2%
Aspects	5	2%
Software transparency	4	2%

These topics can also be understood as subtopics, since each one of them is related to at least one topic from Table 3. Out of these subtopics, the most related ones were: i*, Models transformation and Oriented goals, also known as GORE (Goal Oriented Requirement Engineering). The i* is an objective oriented approach used to describe not only social and intentional needs in the organizational environment, but also functional and non functional Requirements [9]. These three topics appear in conjunction in many situations, since there are strongly related.

With the mapping of the main topics and institutions, the relation of these topics with the institutions and vice versa has been also performed.

3.4 Which institutions have been discussed the main topics?

According to the 3 highlighted topics in Table 6 and the most related subtopic in Table 7, it was possible to identify which institutions had the most publications related to these topics. In Table 8 it is shown the 3 institutions with the most papers related to the 4 topics selected for evaluation.

Table 8. The main topics and related institutions

Topic	Institution	Number of papers
Requirements elicitation	Pontifícia Universidade Católica do Rio de Janeiro	13
	Universidade Federal de Pernambuco	11
	Universidad Nacional de La Plata	8
Requirements modeling	Universidade Federal de Pernambuco	14
	Universidad Politécnica de Valencia	11
	Pontifícia Universidade Católica do Rio de Janeiro	11
Process/Method	Universidade Federal de Pernambuco	10
	Pontifícia Universidade Católica do Rio de Janeiro	6
	York University	4
	Universidade Estadual do Rio de Janeiro	4
i*	Universidade Federal de Pernambuco	12
	Universidade Estadual do Rio de Janeiro	4
	Universitat Politécnica de Catalunya	4

The Requirements elicitation topic was the most discussed one by the Pontifícia Universidade Católica do Rio de Janeiro, whereas the Requirements modeling, Process/Method and i* was the most discussed by Universidade Federal de Pernambuco (UFPE). The UFPE appears as one of the 3 most active institutions as far as the 4 topics analyzed are concerned.

The topics that main institutions have been discussing have also been identified. The results are presented as follows.

3.5 Which topics have been discussed for the main institutions?

Based on the 5 most active institutions highlighted in Table 5, the most discussed topics by them have been analyzed. In Table 9, the institutions, the 3 most discussed topics and the amount of papers related to this topic are presented.

Both the Universidade Federal de Pernambuco and the Universidad Politécnica de Valencia have been discussing with more emphasis the Requirements modeling topic, whereas the Pontifícia Universidade Católica do Rio de Janeiro and the Universidad Nacional de La Plata have been discussing the Requirements elicitation topic. The Universidad Nacional del Centro de la Provincia de Buenos Aires has been emphasizing the Scenario subtopic discussion.

With the mapping of the most discussed topics at WER, it is observed that some topics were more discussed in the past, whereas other topics gained more importance in the last few years.

Based on this observation, some topics were extracted where these variations are more meaningful. These variations may indicate trends in Requirements Engineering and the results are shown below.

Table 9. The main institutions and the related topics

Institution	Topic	Number of papers
Universidade Federal de Pernambuco	Requirements modeling	14
	i*	12
	Requirements elicitation	11
Pontifícia Universidade Católica do Rio de Janeiro	Requirements elicitation	13
	Requirements modeling	11
	Process/Method	6
Universidad Politécnica de Valencia	Requirements modeling	11
	Models transformation	9
	Business modeling	6
	Requirements specification	6
Universidad Nacional del Centro de la Provincia de Buenos Aires	Scenarios	6
	Requirements elicitation	5
	Requirements modeling	5
Universidad Nacional de La Plata	Requirements elicitation	8
	LEL/LAL	7
	Measurement/Metrics	5

3.6 Which are the trends in Requirements Engineering?

According to the variation on the amount of published papers related to the analyzed topics, a trend analysis in the Engineering Requirements area was performed. In order to facilitate the visualization of this analysis, the publications were gathered into triennium groups, coming to a total of 5 groups, according to what is shown in Table 8. Between the second and fourth triennium there has been an increase in the amount of publications, when compared to the first and fifth triennium. It is important to observe this detail so that a correct analysis in the variation of publications can be performed.

Table 10. Numbers of paper published per triennium

Year	Number of papers
1998-2000	37
2001-2003	63
2004-2006	63
2007-2009	55
2010-2012	40

Among the evaluated topics, it has been noted some significant variations. These topics are represented in Table 11. The amount of publications for each triennium and its corresponding percentage of the total amount of published papers are shown on it. This percentage was used to balance the difference in the number of publications that has happed among the trienniums.

Table 11. Number of publications per topic/year

Topic/year	1998/2000	2001/2003	2004/2006	2007/2009	2011/2012
Requirements management	5 (14%)	6 (10%)	12 (19%)	8 (15%)	4 (10%)
i*	2 (5%)	4 (6%)	5 (8%)	7 (13%)	8 (20%)
Measurement/Metrics		4 (6%)	9 (14%)	2 (4%)	1 (3%)
Business modeling		4 (6%)	2 (3%)	4 (7%)	5 (13%)
Oriented-agent paradigm		1 (2%)	8 (13%)	2 (4%)	1 (3%)
Process/Method	4 (11%)	14 (22%)	11 (17%)	10 (18%)	2 (5%)
Quality requirements	1 (3%)	6 (10%)	9 (14%)	5 (9%)	5 (13%)
Software transparency				1 (2%)	3 (8%)

Based on these percentages the graph was created (Fig.2) where it is possible to observe these variations in a clearer manner.

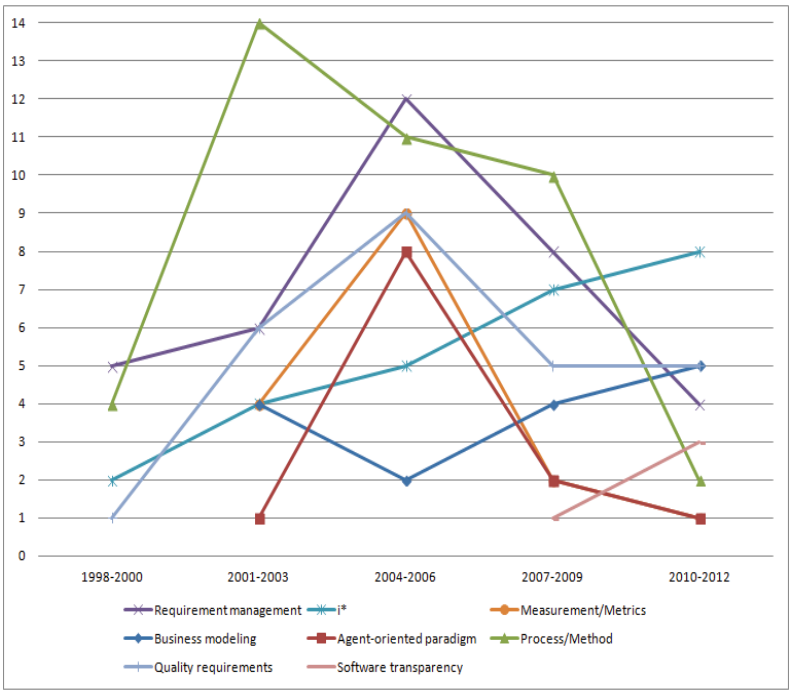


Fig. 2. Publications' variation of some topics

Through Fig. 2 it is possible to observe an increase in the past few years of the following topics: i*, Business modeling and Software transparency. On the other hand, it has been observed a decrease in the number of publication of the following topics: Process/Method, Requirements Management, Measurement/Metrics and Agent-

oriented paradigm. The Quality software topic had its pinnacle in the 2004-2006 trienniums, but it still bears certain stability.

4 Conclusion

When studying a research area, it is important to identify the answers to some questions, such as: the most active groups, the most discussed topics, the identified trends and so forth. This information may bring some benefits as the identification of new information sources, establishment of new partnerships and the orientation of researches toward more relevant topics at a given time

The revision of 258 papers published at WER had the goal of answering these questions as far as this event is concerned. Some of the main obtained results were the identification of Brazil, Spain and Argentina as the home countries of the main institutions that have published at WER so far. The Universidade Federal de Pernambuco located in Brazil, is a reference as the most active in this event. The Requirements elicitation and Requirements modeling topics are the most referenced in the published papers and the Requirements modeling is the most cited in the past few years. It has been observed an increase in the reference to the following topics: i*, Business modeling and Software transparency.

This is a preliminary study, since the issues exposed here may be broaden and explored in a deeper fashion in future works. All the data collected by this paper have been uploaded into an access database and is available for free use[1].

Other analysis may be performed in the database mentioned above, allowing distinct points of view, other than the one presented in this work. A new paper classification may be performed in order to refine the obtained results and it may also amplify the events to be analyzed. A similar but more refined review may be performed, including the main events or journals in the Requirements Engineering area.

References

1. Fiorini, S. T., Leite, J. C. S. P., Lucena, C. J. P. Organizando Processs de Requisitos, In Workshop on Requirements Engineering, 1998.
2. Thayer, R., Dorfman, M. System and Software Requirements Engineering. IEEE Computer Society Press Tutorial, 718p, 2000.
3. Pressman, R., S. Engenharia de Software – Uma Abordagem Profissional. McGraw-Hill, 2011.
4. Dominguez, J., The Curious Case of the CHAOS Report 2009.

[1] https://docs.google.com/open?id=0Bx4BYAKqdH1xc05CalJ0Mmlualk

5. Davis, A., Dieste, O., Hickey, A.; Juristo, N., Moreno, A. M. Effectiveness of Requirements Elicitation Techniques: Empirical Results Derived from a Systematic Review. In Proceedings of the 14th IEEE International Requirements Engineering Conference. IEEE Computer Society, 2006
6. Dieste, O., Juristo, N. Systematic Review and Aggregation of Empirical Studies on Elicitation Techniques, IEEE Transactions on Software Engineering, IEEE computer Society Digital Library. IEEE Computer Society, 2010
7. Condori-Fernández, N., Daneva, M., Sikkel, K., Wieringa, R., Dieste, O., Pastor, O. A systematic mapping study on empirical evaluation of software requirements specifications techniques. Third International Symposium on Empirical Software Engineering and Measurement ESEM 2009, IEEE Computer Society, Florida, USA, pp 502-505, 2009.
8. Cezario, R., Barreto, F., Benitti, V. Padrões de Escrita de Requisitos : um mapeamento sistemático da literatura. In Workshop on Requirements Engineering, 2011.
9. Lucena, M., Silva, C., Santos, E., Alencar, F., Castro, J. Modularizando Modelos i *: uma Abordagem baseada em Transformação de Modelos. In Workshop on Requirements Engineering, 2009.

Índice de Artículos

Índice de Autores