Proceedings of the Eighth International i*Workshop - iStar 2015

in conjunction with the 23rd International Requirements Engineering Conference (RE 2015)

Ottawa, Canada, August 24-25, 2015

Edited by

Jaelson Castro*

Gilberto Cysneiros Filho**

Sotirios Liaskos***

*Universidade Federal de Pernambuco, Recife, Brazil

**Universidade Federal Rural de Pernambuco, Recife, Brazil

***York University, Toronto, Canada

a CEUR Workshop Proceedings, ISSN 1613-0073

http://ceur-ws.org/Vol-1402

Table of Contents

- Formalization of i* Mapping Rules for Class Diagram. Josenildo Melo, Aêda Sousa, Celso Sá Filho, Fernanda Alencar, p. 97-102
- US2StarTool: Generating i* Models From User Stories. Renato Mesquita, Aline Jaqueira, Marcia Lucena, Celso Sá Filho, Fernanda Alencar, p. 103-108
- Specifying Guidelines to Transform i* Model into User Stories: an Overview. Celso Sá Filho, Aêda Souza, Josenildo Melo, Marcia Lucena, Fernanda Alencar, p 109-115

Organization

Organizing Committee

- Jaelson Castro (Universidade Federal de Pernambuco, Brazil)
- Sotirios Liaskos (York University, Canada)

Steering Committee

- Xavier Franch (Universitat Politecnica de Catalunya, Spain)
- John Mylopoulos (University of Trento, Italy)
- Eric Yu (University of Toronto, Canada)

Publicity Chair

- Carla Silva (Universidade Federal de Pernambuco, Brazil)

Proceedings Chair

- Gilberto Cysneiros (Universidade Federal Rural de Pernambuco, Brazil)

PC members

- Fernanda Alencar
- Raian Ali
- Daniel Amyot
- Juan Pablo Carvallo
- Dolors Costal
- Luiz Marcio Cysneiros
- Fabiano Dalpiaz
- Neil Ernst
- Hugo Estrada
- Sepideh Ghanavati
- Paolo Giorgini
- Daniel Gross
- Renata S.S. Guizzardi
- Jennifer Horkoff
- Alexei Lapouchnian
- Maria Lencastre
- Julio Cesar Sampaio do Prado Leite
- Lin Liu
- James Lockerbie
- Lidia López
- Alejandro Mate
- Mirko Morandini
- Haralambos Mouratidis
- Gunter Mussbacher
- Elda Paja

- Oscar Pastor
- Michalis Pavlidis
- Anna Perini
- Michael Petit
- Alicia Martinez Rebollar
- André Rifaut
- Carla Silva
- Vitor E. Silva Souza
- Angelo Susi
- Yves Wautelet
- Jelena Zdravkovic

Preface

The iStar workshop series is dedicated to the discussion of concepts, methods, techniques, tools, and applications associated with i* and related approaches. Following successful workshops in Trento, Italy (2001), London, England (2005), Recife, Brazil (2008), Hammamet, Tunisia (2010), Trento, Italy (2011), Valencia, Spain (2013), and Thessaloniki, Greece (2014), this year (2015) the workshop was held in Ottawa, Canada. As with previous editions, the workshopb's objective was to provide a unique opportunity for exchanging ideas and recent progress, comparing notes, and forging new collaborations. This year, the workshop was in conjunction with the 23rd International Requirements Engineering Conference (RE'15), benefiting from the common themes and interests shared by the two events.

As with past editions, we have tried to keep the format informal so as to maximize interaction. Aiming at an inclusive and discussion-oriented workshop, the main criterion for paper acceptance in iStar'15 was relevance and potential for raising discussion. A 36-member program committee, consisting of scholars and practitioners with expertise and interest in the field, were involved in reviewing a total of 26 complete paper submissions. Each of the papers was reviewed by three arms-length program committee members. Of the submitted papers, 19 were accepted for presentation in the workshop and revised versions of the papers are included in the proceedings that follow. In the event, 13 out of the 19 accepted papers were given 30 minutes for presentation and discussion. The remaining 6 were deemed by the reviewers to be reporting work at its earlier stage and were given a short presentation slot of 15 minutes.

In addition to the paper presentation sections, the workshop included a keynote presentation by Prof. Julio Cesar Sampaio do Prado Leite entitled "Elicitation Awareness in Conceptual Modeling: The Role of Transparency", as well as a keynote presentation by Prof. Dr. Matthias Jarke titled "Strategy Modeling vs. Data-Driven Engineering: Can They Be Reconciled?". The event closed with a panel session on the i* standardization effort, featuring Daniel Amyot, Jennifer Horkoff and Eric Yu.

We would like to thank the authors and participants for their contributions and we hope they benefited from their participation. We would also like to show our appreciation the reviewers for their good work on the papers as well as the International i* Workshop steering committee for its advice and support throughout. Last but not least, we want to thank the organizers of the RE'15 conference for their help support during various stages of organizing this workshop.

Jaelson Castro, Universidade Federal de Pernambuco, Brazil
Sotirios Liaskos, York University, Canada

Elicitation Awareness in Conceptual Modeling: The Role of Transparency

Julio Cesar Sampaio do Prado Leite

Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)
julio@inf.puc-rio.br

This talk will review how transparency brings the possibility of process awareness in requirements building. Conceptual modeling has been the prominent facet of requirements engineering with good reasons.

The task of specifying, at a high level of abstraction, the myriad of knowledge needed to anchor software development is one of the great challenges of computer science. As such, different artificial languages have been proposed to serve as basis for this task.

One of the consequences of the central role of modeling, has been that, often, elicitation processes are driven by the modeling language of choice. As such, elicitation is partially performed, increasing the possibility of faulty models.

Attaching the non-functional requirements transparency to the requirements building process increases the overall awareness in the universe of discourse, making explicit the information sources, their rationale, as well as the requirements engineers rationale.

In particular, we will focus on the challenges of how to introduce transparency as to help towards elicitation awareness in conceptual modeling. We will use i* as an example of modeling language.

Strategy Modeling vs. Data-Driven Engineering: Can They Be Reconciled?

Matthias Jarke

Informatik 5
RWTH Aachen
jarke@informatik.rwth-aachen.de

The original idea of i* was to enable light-weight graphical modeling of multi-stakeholder strategies in business and information systems engineering. Modeling the strategic business environment was considered an elegant way to bridge the well-known gap between business process management and information systems, e.g. in the Tropos methodology. With the digitalization of more and more aspects of life, the resulting flood of "big data" has enticed entirely different bottom-up development strategies emphasizing rapid time to market, continuous customer-driven system evolution, end-user and open source development. After reflecting a bit on these two very different trends, the talk will discuss if and how strategy-driven and data-driven systems engineering can profit from each other. Some recent experiences from supporting professional web communities, mostly in the context of engineering knowledge management and continuous education, will illustrate the ideas.

Goal Modeling Education with GRL: Experience Report

Daniel Amyot

School of Electrical Engineering and Computer Science
University of Ottawa, Ottawa, Canada
damyot@uottawa.ca

Abstract. Goal modeling and analysis with the Goal-oriented Requirement Language (GRL) is taught in software engineering and computer science at the University of Ottawa since 2003. This paper presents the general education approach taken in an undergraduate requirements engineering course and in a graduate software engineering course. Some of the particularities of these courses involve the use of a general GRL modeling pattern, the combined use with Use Case Maps (for operationalization and for business processes), the coverage of qualitative and quantitative analysis approaches, the use of indicators, and automated evaluations of strategies supported by the jUCMNav tool. This paper also reflects on some successes and difficulties observed in the past decade while teaching these concepts.

Keywords: Education · Experience · Goal-oriented Requirement Language · jUCMNav

1 Introduction

Goal modeling is taught at the University of Ottawa since 2003. The Goal-oriented Requirement Language (GRL), part of the User Requirements Notation (URN) [1,7], is the language being taught in the following courses, with a total audience of well over 1,000 students:

- *Introduction to Software Engineering* (2003-2004), undergraduate, computer science program, 3rd-year, without tool support or labs.
- *Software Requirements Analysis* (2005-2014), undergraduate, 3rd-year, software engineering program, with tool support and labs.
- *Software Engineering* (10 times between 2004 and 2015), graduate, masters and Ph.D., computer science program, with tool support but no labs.

In the above semester-long courses, goal modeling and analysis with GRL is the topic of a 3-hour lecture. In addition, one course has a 3-hour laboratory where students can learn jUCMNav, an Eclipse plug-in for GRL modeling and analysis, with the help of a teaching assistant [6]. The same laboratory material (tutorial on the construction of a

goal model and strategies, with an additional exercise) is made available to the students of the graduate course, but for self-study only. All courses get an assignment with a GRL modeling and analysis problem (textual description) that requires the use of jUCMNav. Another (shorter) problem is always present in a partial/final exam.

This paper reports on my experience teaching GRL, mainly for the past decade (the last two courses) because of the availability of tools. The focus here is on the styles of models being taught (section 2) and on the types of analyses they enable (section 3), together with the informal feedback I perceived from students during the courses.

2 Teaching and Learning GRL Modeling

GRL is a rich goal-oriented modeling language that can be used for different purposes, from social modeling to decision making and rationale documentation. The courses introduce GRL with the later part. After a bird's eye introduction to URN, students are taught the need for documenting design rationales in software. *Rationales* are often described with comments in the code and in repository commits in practice. They are also often represented in a tabular way, where different options (rows) are compared against different criteria (columns), using a coarse-grained qualitative scale or a quantitative scale with some aggregation function (e.g., weighted sum). Students often see such tables in magazines and web sites. They then learn that such tabular representation only captures a partial view of reality: dependencies between criteria are not shown, and "who is concerned with what criteria" is not described either. With these limitations in mind, the concept of goal and its use in software engineering are introduced, followed by the syntax of GRL based on a security example.

This quickly leads to the introduction of a *GRL pattern* (Fig. 1) that describes general decision making for systems with alternatives.

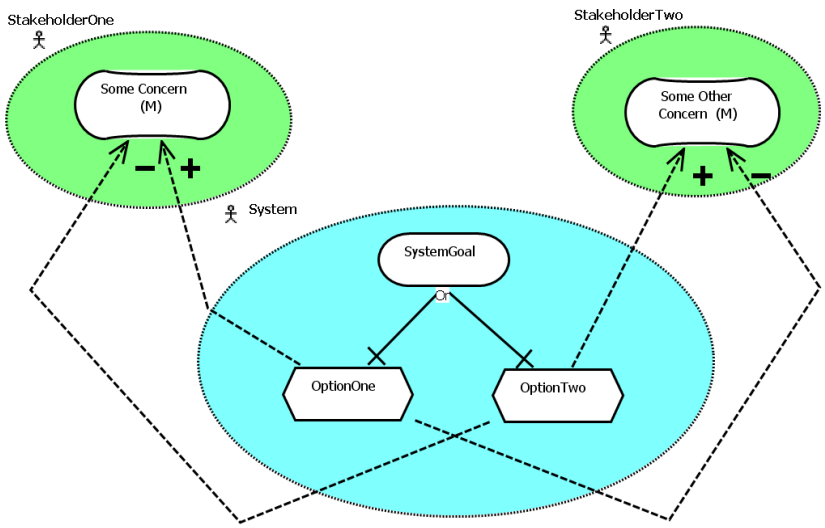


Fig. 1. Common GRL pattern for decision making in context

As shown in this figure, system functionalities are captured with *goals* (that can be AND-decomposed). The various means of achieving these goals are captured as *tasks* linked with OR-decompositions. *Actors* capture system stakeholders and *softgoals* represent their concerns, often non-functional in nature. Some of the alternative tasks will also have some positive or negative impacts on some of the stakeholder concerns.

With such a pattern, the students learn that global decisions (one for each OR-decomposition, in order to satisfy system goals) become non trivial and that it is difficult to satisfy all actors (leading to the notion of *trade-off*). They also learn that once agreed on, a global decision (defined as a *strategy* in GRL) combined with the model itself document the "why" aspect of the system, which is a view absent from UML.

GRL supports quantitative scales for contributions ([-100..100]), for satisfaction values, and for the importance of intentional elements to their containing actor. There are also qualitative scales for contributions {Break, ..., Make}, satisfactions {Denied, …, Satisfied}, and importance {None, …, High}. Students generally understand that a qualitative scale is used in the early modeling steps, when little information is available. As the understanding of the problem and of the potential impact of solutions improves, the modeler can move to a more fine-grained quantitative scale.

However, in a quantitative context, it becomes difficult to find proper values (e.g., should the weight of a contribution be 30 or 40?). Although jUCMNav and GRL contain some features that can help cope with such decisions, e.g., value ranges and contribution overrides [3], there is neither time to cover these advanced features nor their other usages (e.g., for sensitivity analysis). Yet, some time is devoted to the coverage of GRL *indicators* as a means to better root part of the goal model in reality, and also to connect GRL to business process modeling (where managers use indicators to monitor systems and satisfaction). In GRL, an indicator converts an observable value (in a real unit like kilometers or Euros) to a GRL satisfaction value through a comparison with target, threshold, and worst-case values. Students learn to convert uncertain contribution weights into more meaningful indicators with a contribution of weight 100. This helps remove some of the uncertainty and make models more falsifiable [8].

URN combines GRL with *Use Case Maps* (UCM), a notation for causal scenarios superimposed onto a component structure [7]. In both courses, a second 3-hour lecture is devoted to UCM (with jUCMNav support). Students learn to use UCM to operationalize some of the tasks and goals found in GRL models. They learn that both views are needed for requirements engineering activities and for business process modeling as the "why" and "how much" aspects are uniquely covered by GRL and the "when" aspects are uniquely covered by UCM (both cover "what", "who" and "where" aspects). They also learn that both views can be developed in parallel and iteratively as some stakeholders (e.g., managers) tend to discuss more in terms of objectives (at the GRL level) whereas others involved in operations will describe their knowledge and needs more in terms of UCM-like concepts. Finally, they also realize that creating traceability links between elements of the GRL and UCM views can help answer *consistency* and *completeness* questions (e.g., why keep a scenario without a goal/purpose? Why is this functional goal not refined by any scenario?). All of these concepts are illustrated with intuitive examples from the telecommunication domain, discussed briefly in [1].

The complexity of goal models can quickly become an issue [5], and this is also the case in GRL (as it is in *i**) because the language lacks modularity constructs. In jUCMNav, complexity is managed through having different diagrams (views) as part of one model, where the elements and links can be referenced in multiple views. jUCMNav supports many ways of navigating between references of an intentional element or actor. Students learn to decompose complex models with many diagrams.

Unlike *i**, GRL does not distinguish explicitly between *strategic dependency* (SD) diagrams and *strategic rationale* (SR) diagrams. These views can be distinct in GRL, but they are often intertwined. While several other courses often start with SD diagrams to introduce goal modeling [3], our courses focus first on the SR view, and covers GRL dependency links later, in a brief introduction to social modeling. One lesson learned from experience is that computer science and software engineering students see the value of SR views right away, but they do not see the value of SD views in the limited time available to cover goal modeling.

Although the students can distinguish between goals, softgoals and tasks fairly well, means-end links are no longer taught (as OR-decomposition is used instead). Correlation links are mentioned but not taught (contributions are used instead), and resources and beliefs are barely mentioned. In the early years, all of these concepts were covered, but the students were puzzled by the need for so many types of intentional elements and links. As students were using them mostly incorrectly, their coverage was simply minimized, without a real negative impact on the expressiveness of resulting models. This subset shares many commonalities with what is being taught by Dalpiaz in his first-year course for information system students [6].

On a couple of years, both at the undergraduate and graduate levels, my colleagues and I attempted to discuss other goal modeling languages (especially *i** and KAOS) and compare them with GRL, but this did not raise any real interest. I suspect this is because small differences between little-known languages do not become attractive until one goal language is actually mastered. Now we only mention their existence.

3 Teaching and Learning GRL Analysis

The teaching of analysis techniques is done iteratively and is interleaved with modeling. Once the GRL syntax is introduced, jUCMNav is used to create a model on the fly based on suggestions of the students on a domain they know (e.g., a university registration system). Once a few actors, their intentions, and some links (especially OR-decompositions) are available, the notion of "what-if" analysis is introduced. In GRL, what-if situations are captured with *strategies*, which are initial satisfaction values assigned to some of the intentional elements. A strategy is evaluated via a propagation algorithm, and several qualitative and quantitative ones are supported in jUCMNav [2]. The semantics of the various GRL links is revisited, this time in a more formal way based on the qualitative and quantitative propagation of satisfaction values (including factor evaluations). One lesson learned here is that a better understanding of how these algorithms work leads to a better and more consistent selection of GRL relationships (e.g., decomposition versus contributions) by students.

A related lesson learned is that students in these courses enjoy *automated* analysis and have little interest in manual or interactive propagation, maybe because of the immediate feedback and low effort coming with automation. With jUCMNav, one can easily describe many strategies, and the analyst can go from one to the next with instantaneous evaluation feedback. This is not something they could do with manual or interactive propagation. Conflicts can still be detected (e.g., via rules in the Object Constraint Language, automatically checked by jUCMNav) and then resolved in the strategy by explicitly setting the (resolved) satisfaction value of the intentional element under conflict. Students also learn to compare strategy results in different ways:

- By alternating between the graphical evaluations of the strategies of interest;
- By using a *strategy diff* feature in jUCMNav, which shows the deltas in the evaluation of a strategy on the model compared to that of a base strategy [3];
- By exporting the resulting evaluations of strategies as an Excel/CSV file or by generating a tabular report in HTML or Word using jUCMNav.

Special attention is dedicated to *trade-off analysis* with the help of pre-existing examples. Students learn how to create a reasonable set of potential strategies and select the "best one" based on which actor(s) we want to prioritize in terms of satisfaction. One challenge here is to come up with a set of strategies that is reasonable while not being exhaustive (as sometimes the number of potential combinations explodes rapidly). This is a problem akin to test creation and selection in software testing.

In a different lecture of the graduate course, students are introduced to *aspect-oriented modeling* (AOM), and GRL and UCM are revisited in that context. Students are taught the benefits of aspects for handling cross-cutting concerns, with examples exploiting an aspect-oriented extension of URN [9]. Aspect-oriented GRL is briefly introduced, but not tested in assignments or exams (only aspect-oriented UCM is used in assignments and exams, because of the existence of partial tool support in jUCMNav). The value of aspect-oriented modeling and analysis is also clearer in a UCM context than in a GRL context, because cross-cutting concerns are less frequent at the goal level than in lower-level operational details.

The graduate students are also given another lecture on the *Object Constraint Language* (OCL), and again this is an opportunity to revisit GRL. jUCMNav has over one hundred predefined static semantic rules described with OCL, and it offers a user interface to select which ones to apply to a model, and even to create new ones [4]. There are, for example, rules use to restrict the use of the GRL language to a modeling style compatible with *i**. OCL is also used to compute various metrics on GRL models. Students are briefly taught about the URN metamodel, and how to create new rules or metrics using OCL and jUCMNav. This is further tested in an assignment.

4 Conclusions and Future Work

This paper provided a brief overview of the GRL education material taught in courses at the University of Ottawa, together with my perception of the student feedback on their learning experience related to modeling and analysis of goal models with GRL.

Students of the undergraduate course have to do a semester-long project in teams, leading to the creation and validation of a software requirements specification for real

stakeholders. One encouraging result is that some teams voluntarily choose to use GRL to model stakeholders' objectives together with the intended functionality of the system, and with alternatives defined and selected through "what-if" analysis of strategies. Similarly in the graduate course, where teams of students need to compare two software engineering tools in a given business context, several teams use GRL to capture stakeholder objectives and the impact of choosing one tool over the other one.

There are yet many opportunities to improve the learning experience of students in these courses. For one, there is a major lack of online video material (e.g., on YouTube) teaching goal modeling with GRL (or other languages), including tool support. We have a series of YouTube videos that were produced in 2010 to explain UCM modeling with jUCMNav. Although they totalize only about half an hour, these videos are much appreciated by the students, who wonder why there is nothing equivalent on the GRL side. Students also ask for more exercises that are smaller than those found in the assignment, with solutions, to better measure their understanding.

Future work is also needed on understanding the best subset of GRL to first introduce, and whether this would be different for undergraduate and graduate students.

Acknowledgments. The undergraduate courses discussed here have been taught in collaboration with at least four other professors over the years (G. v. Bochmann, O. Kabranov, S. Somé, and G. Mussbacher), partly because of the existence of multiple French/English sections and because of sabbatical years. I would like to thank them for a fruitful collaboration over the years. I also thank our numerous teaching assistants who have contributed to the education of GRL and jUCMNav.

References

1. Amyot, D., Mussbacher, G.: User Requirements Notation: The First Ten Years, The Next Ten Years. Journal of Software (JSW), Vol. 6, No. 5, 747–768 (2011)
2. Amyot, D., Ghanavati, S., Horkoff, J., Mussbacher, G., Peyton, L., Yu, E.: Evaluating Goal Models within the Goal-oriented Requirement Language. International Journal of Intelligent Systems (IJIS), Vol. 25, Issue 8, 841–877 (2010)
3. Amyot, D., et al.: Towards Advanced Goal Model Analysis with jUCMNav. ER Workshops 2012, LNCS 7518, Springer, 201–210. http://softwareengineering.ca/jucmnav (2012)
4. Amyot, D., Yan, J.B.: Flexible Verification of User-Defined Semantic Constraints in Modelling Tools. CASCON 2008. ACM Press, 81–95 (2008)
5. Babar, Z., Nalchigar, S., Lessard, L., Horkoff, J., Yu, E.: Instructional Experiences with Modeling and Analysis using the *i** Framework. 1st Int. iStar Teaching Workshop. CEUR-WS, Vol. 1370, 31–36 (2015)
6. Dalpiaz, F.: Teaching Goal Modeling in Undergraduate Education. 1st Int. iStar Teaching Workshop. CEUR-WS, Vol. 1370, 1–6 (2015)
7. ITU-T: Recommendation Z.151 (10/12): User Requirements Notation (URN) – Language definition. Geneva, Switzerland (2012)
8. Letier, E., Stefan, D., Barr, E.T.: Uncertainty, risk, and information value in software requirements and architecture. ICSE 2014. IEEE CS, 883–894 (2014)
9. Mussbacher, G., Amyot, D. Araújo, J., Moreira, A.: Requirements Modeling with the Aspect-oriented User Requirements Notation (AoURN): A Case Study. Transactions on Aspect-Oriented Software Development VII, LNCS 6210, Springer, 23–68 (2010)

Exploiting Online Discussions in Collaborative Distributed Requirements Engineering

Itzel Morales-Ramirez, Matthieu Vergne, Mirko Morandini, Anna Perini, and Angelo Susi

Fondazione Bruno Kessler
Via Sommarive 18, 38123 Trento, Italy

Abstract. Large, distributed software development projects, like Open Source Software (OSS), adopt different collaborative working tools, including online forums and mailing list discussions that are valuable source of knowledge for requirements engineering tasks in software evolution, such as model revision and evolution. In our research, we aim at providing tool support for retrieving information from these online resources, and for analyzing it. The solution we propose combines natural language processing techniques, machine learning, statistical and search based techniques to address two key problems, namely the so called expert finding problem, and the problem of identifying requests for changing requirements or soliciting new requirements, by exploiting online discussions.
In this paper, we describe the solution approach set up so far with the help of an OSS scenario, discuss some preliminary evaluation, and highlight future work.

1 Introduction

Large and distributed software development projects, such as Open Source Software (OSS), challenge traditional software development paradigms and call for tools to support distributed requirements elicitation, modeling, requirements negotiation, and the management of distributed teams [10]. OSS projects usually adopt online community platforms to support cooperative work. Platforms such as online forums and mailing lists where different types of stakeholders (e.g., users, developers, and analysts) share their knowledge by engaging in discussions, mainly using free or semi-structured Natural Language (NL) text. The level of expertise, on specific topics, of a particular stakeholder may be revealed by an analysis of these discussions. This information is key for an effective and successful software evolution process. While these discussions grow, spreading over a variety of interconnected domain concepts, and involving more and more stakeholders, manual analysis becomes rapidly an effort demanding and error prone task.

Automated support for extracting relevant information from these online discussions, and for identifying and ranking those members who can contribute key knowledge about a given topic, is crucial to improve collaborative Requirements Engineering (RE) and to provide better support to project management [4, 1]. In our research we are combining Natural Language Processing (NLP) techniques, Machine Learning (ML), statistical and search based techniques to provide a tool-supported method for the analysis of large message archives, with the ultimate purpose of supporting a requirements-driven evolution process as sketched in Fig. 1. The input to the process is user feedback expressed in online discussion as depicted in Fig. 2 (left side), which represents an excerpt

of a discussion posted in the OpenOffice Bugzilla issue tracking system[1]. These online discussions are considered a source of requirements knowledge (e.g., user's goals and preferences, domain assumptions), which can be extracted using analysis techniques so that experts can assess its impact on an existing requirements model. Key elements of the online discussion are highlighted in Fig. 2 (left side), namely topic, terms, participant and feedback classification. The user *dhpeterson* posted her message as *ENHANCEMENT* and it is accepted by the engineer who was managing the issue tracking system, thus providing input to release planning tasks, including requirements model revision and extension. For instance, the analysis of this user feedback could lead to a revision of the goal model representing the requirements of the OpenOffice Writer component, depicted in Fig. 2 (right side).

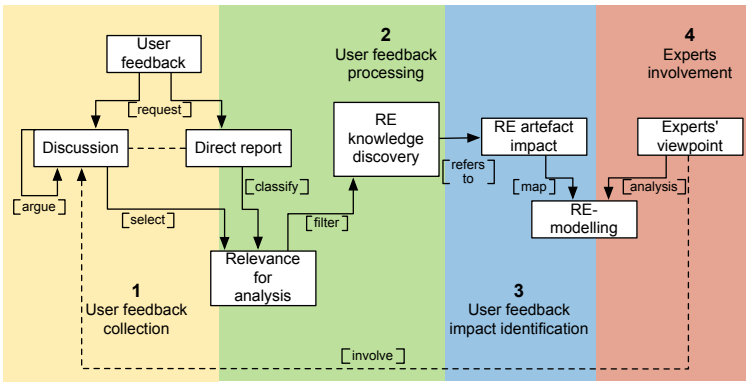


Fig. 1. Overview of our approach to requirements-driven software evolution.

The proposed process for exploiting online feedback and discussions addresses two basic problems, namely how to elicit requirements knowledge from big online discussions (we refer to this problem as *requirements knowledge discovery*), and how to elicit indicators of the level of expertise of the participants to a given discussion (we refer to it as *expert finding* problem). An initial vision of our approach for the case of Tropos requirements models have been presented in a previous iStar Workshop (iStar 2013 [13]). In this paper we give some details on the techniques we exploit (Section 2). We recall the main results achieved so far (Section 2), and ongoing and future work (Section 3).

2 Approach: RE knowledge discovery and Expert Finding

In this section we propose a formulation for the *Requirements knowledge discovery* and the *Expert Finding* problems, introduce the types of analysis and relative implementation techniques that we have defined so far. We will refer to the example taken from online discussions in OSS projects depicted in Fig. 2.

The Requirements Knowledge Discovery Problem. Online discussions can be considered almost synchronous written conversations [12] that can be described in terms of

[1] `https://bz.apache.org/ooo/show_bug.cgi?id=76801`

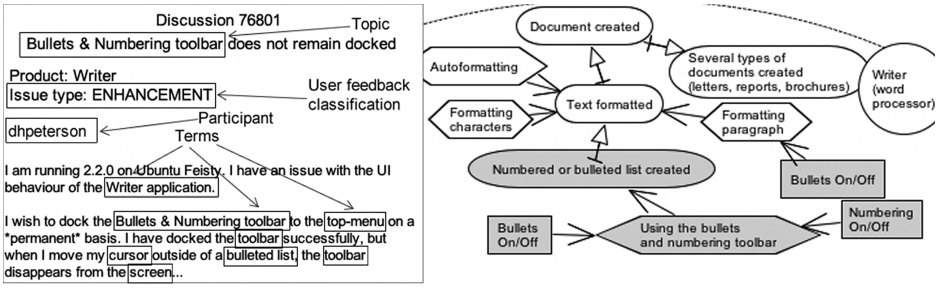


Fig. 2. Left: an excerpt of a discussion posted in the OpenOffice Bugzilla issue tracking system, with its key elements. Right: an excerpt of a requirements model of the OpenOffice-Writer component, in Tropos notation.

speech-acts, according to the Speech Act Theory (SAT) [2]. SAT rests on the following idea –by saying something, we react by doing something–, details of the used theory can be found in [7]. In other words, the speaker's intention can be one of persuading, inspiring or getting a hearer to do something and such an intention is manifested in the *speech-acts* she/he uses. Concretely, speech acts are classified according to specific performative verbs, such as *suggest*, *recommend*, *confirm*, and *advise*, etc., which reveal the speakers' intentions. On these premises, discovering requirements knowledge in online discussions amounts to recognizing those fragments of conversation that contain specific *speech-acts* combinations or patterns, which are found to be commonly used for expressing feature requests, bugs or clarification requests.

The Expert Finding Problem. The problem of expert finding in online discussions can be conceived as a problem of Information Extraction (IE). Formally, given a set P of *Participants* (who are users and developers participating to the mailing list, such as "dhpeterson") engaged in a set of discussions D (e.g., "Discussion 76801") about a set of topics T (e.g., "Bullets and Numbering toolbar"), the problem of expert finding can be stated as the problem of ranking the *Participants* according to their expertise on a topic $t \in T$, based on the emphasis made in a sentence expressing their intentions I (e.g., classifying a message as FEATURE or ENHANCEMENT) and their use of terms related to the topic t (e.g., "bullet list").

We apply two types of analysis to online discussion archives: (a) the analysis of the structure of the archives – i.e., the authors of the messages and the terms found in the messages with their frequency; we refer to this as to the **content** of online discussions; and (b) the analysis of the **intentions** of the participants to an online discussion – i.e., identification of *speech-acts* combinations and patterns.

Our approach to address the Requirements Knowledge Discovery problem rests on the second type of analysis. Specifically, we take online discussions as txt files that we pre-process and clean using NLP techniques. We apply a sentence splitter, a tokenizer, and use the Hepple Tagger (POS tagger). Once each word has been tagged, we use a lemmatizer, gazetteers and lexico-syntactic rules, called JAPE, to finally annotate the intentions [7]. After the annotation is performed we recognize some patterns or combinations of intentions that can lead us to identify a possible feature or bug. For example, feature request indicators correspond to combination of *speech-acts* of type *Requirements* and one among the followings *Positive opinion*|*Questions*|*Suppositives*|*Suggestive* or as a combination of the *speech-act* type *Positive* with an URL link.

To address the Expert Finding problem we defined two approaches. The first exploits the analysis of the content of online discussions, as above defined, the second combines intention and content analysis. Specifically, we first retrieve stakeholders, terms and topics by applying NLP techniques for IE to the considered online discussions. The retrieved elements are represented as nodes of a weighted graph, and by counting their co-occurrences (e.g., how many times a given stakeholder use a given term) we define weights for their connecting arcs. By building a Markov Network (MN), we can compute probabilities for each node of the graph based on the specific topic we are looking experts for. On the basis of these probabilities, stakeholders are ranked along their expertise to a given topic [14]. In the intent- and content-based approach the weights of the arcs in the graph take into account the stakeholder's intentions revealed in her/his messages when talking about a given topic [9].

Results so far. Since the initial vision of our research, which has been presented in [13, 9], we refined and experimentally evaluated the proposed analysis techniques. The online discussion archives we consider are those taken from OSS projects. The overall dataset that we have crawled consists of 713 discussions from the year 2012, containing 2728 messages and 215 participants. For evaluating the analysis techniques we devised an experiment to address the requirements knowledge discovery. We have sampled 20 discussions with 310 messages (1685 sentences) and designed an empirical evaluation aiming at measuring the time effort required for manually annotating sentences w.r.t. their intentions [7]. A first execution of the study has been performed with twenty subjects working in a distributed way using a crowdsourcing-like platform. We observed that the time for annotating has the following distribution: a mean of 35 seconds, a median of 18 and a standard deviation of 56 seconds. On the basis of the participant profiles we derived some insights based on a post-questionnaire filled in by the participants to the study and we applied ANOVA test to identify influencing factors. A technical report discussing the results is available online[2].

Concerning the approach to the expert finding problem, MN techniques revealed scalability problems when considering large OSS discussions, and the attempt to mitigate them using approximate computation is not satisfactory since it gives unstable solution rankings. In order to fix this stability issue, we are now investigating the applicability of search-based techniques. For the implementation, we use NSGAII [5] which allows us to properly deal with the scalability problem, although the functions we tried to optimize so far did not allow us to get meaningful results yet.

Related work. Regarding relevant related work of the requirements knowledge problem, we shall mention the automated identification of intentions presented by Twitchell et al. [12]. This investigation proposes a tool that is based on SAT, dialogue acts and fuzzy logic to analyze transcripts of telephone conversations. The goal of this research is to derive participant profiles based on a map of the intentions expressed in the conversation. The classification of emails using speech acts is investigated by Carvalho et al. [3]. They are interested in classifying emails regarding office-related interactions as negotiation and delegation of tasks. Among the related works that are of particular interest in the expert finding problem, we can mention Serdyukov and Hiemstra [11], in which the content of documents is analyzed to identify the contributions of their different authors

[2] `http://selab.fbk.eu/imramirez/TR_CAiSEDec2014.pdf`

(i.e., used terms) and the probability that a given author relates to a queried topic is computed. Focusing on the social dimensions, the work of Zhang et al. [15] consider the question/answers in an online forum to identify the knowledge seekers (i.e., non-experts) and the knowledge providers (i.e., experts). They compare several algorithms to rank people, starting from the simple counting of answers, then combining it with the number of questions, which should be negatively correlated to the level of expertise, before to propagate the computed values over the community (PageRank-like) to simulate social recognition.

3 Ongoing and Future work

In parallel with the above mentioned improvements of the analysis techniques, we are working at the consolidation of our problem formulation through the elaboration of a conceptual framework that rests on novel ontologies. Specifically, for the purpose of giving an ontological foundation to our intent-based analysis approach, we are extending a communication ontology that relies on the SAT and that accounts for concepts regarding the software development [8].

Analogously, we are working on a more rigorous conceptualization of expert finding systems to help find new indicators to be considered. Indeed, it seems that the indicators used so far (terms and topics, additionally roles) relate to the accessible knowledge and the social recognition of the stakeholders, which seem not the best indicators for inferring expertise [6]. We believe that this conceptualization could be helpful to the research and engineering community, also for supporting a more robust and faster design of expert finding systems.

As for future work, we believe that our intent-based analysis of online discussions could be used in combination with already existing techniques as topic modeling or sentiment analysis to build an enriched classification model for categorizing messages as bug reports or feature request. For example by doing a basic text analysis[3] of the first message posted in the discussion 76801, we can get information on terms and frequencies, as those summarized in Table 1. But the intent-based analysis can provide an emphasis on the importance of the found terms, if the terms are contained in a sentence expressing a certain type of intention. Further work is needed to combine information on frequency and similarity between phrases to rank the relevance of a section of a goal model to be changed.

Table 1. Top 3 phrases (containing nouns) with their frequency and similarity % (Levenshtein distance) w.r.t. the goal *Numbered or bulleted list created* and the task *Using the bullets and numbering toolbar* depicted in Fig. 2.

4 words	F	G-sim%	T-sim%	3 words	F	G-sim%	T-sim%	2 words	F	G-sim%	T-sim%
observe that the toolbar	4	21	35	the bulleted list	6	45	35	the toolbar	10	15	28
of the bulleted list	2	45	38	that the toolbar	4	15	33	bulleted list	9	39	25
bullets & numbering toolbar	2	24	66	a bulleted list	3	42	28	the bulleted	6	30	28

[3] `http://www.online-utility.org/text/analyzer.jsp`

Concerning the evaluation of our intent-based analysis approach, we are also designing a study for evaluating manual annotations versus automatic annotation of our tool, measuring the accuracy in terms of precision and recall.

References

1. T. A. Alspaugh and W. Scacchi. Ongoing software development without classical requirements. In *21st IEEE RE 2013, Rio de Janeiro-RJ, Brazil, July 15-19*, pages 165–174. IEEE Computer Society, 2013.
2. K. Bach and R. M. Harnish. *Linguistic Communication and Speech Acts*. MIT Press, Cambridge, MA, 1979.
3. V. R. Carvalho and W. W. Cohen. On the collective classification of email speech acts. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 345–352. ACM, 2005.
4. C. Castro-Herrera and J. Cleland-Huang. Utilizing recommender systems to support software requirements elicitation. In *RSSE*, RSSE'10, pages 6–10. ACM, 2010.
5. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. *A Fast and Elitist Multi-Objective Genetic Algorithm: NSGA-II*. 2000.
6. K. A. Ericsson. The Influence of Experience and Deliberate Practice on the Development of Superior Expert Performance. In K. A. Ericsson, N. Charness, P. J. Feltovich, and R. R. Hoffman, editors, *The Cambridge handbook of expertise and expert performance*, pages 683–703. Cambridge University Press, New York, NY, US, 2006.
7. I. Morales-Ramirez, A. Perini, and M. Ceccato. Towards supporting the analysis of online discussions in OSS communities: A speech-act based approach. In S. Nurcan and E. Pimenidis, editors, *Information Systems Engineering in Complex Environments - CAiSE Forum 2014, Thessaloniki, Greece, June 16-20, 2014, Selected Extended Papers*, volume 204 of *Lecture Notes in Business Information Processing*, pages 215–232. Springer, 2014.
8. I. Morales-Ramirez, A. Perini, and R. S. S. Guizzardi. Providing foundation for user feedback concepts by extending a communication ontology. In *33rd International Conference, ER 2014, Atlanta, GA, USA*, volume 8824 of *LNCS*, pages 305–312. Springer, 2014.
9. I. Morales-Ramirez, M. Vergne, M. Morandini, A. Siena, A. Perini, and A. Susi. Who is the expert? combining intention and knowledge of online discussants in collaborative RE tasks. In *ICSE '14, May 31 - June 07*, pages 452–455. ACM, 2014.
10. B. Nuseibeh and S. Easterbrook. Requirements engineering: a roadmap. In *Proceedings of the Conference on The Future of SE*, ICSE '00, pages 35–46, New York, NY, USA, 2000.
11. P. Serdyukov and D. Hiemstra. Modeling Documents As Mixtures of Persons for Expert Finding. In *Proceedings of the IR Research, 30th European Conference on Advances in Information Retrieval*, ECIR'08, pages 309–320, Berlin, Heidelberg, 2008. Springer-Verlag.
12. D. P. Twitchell and J. Nunamaker, J.F. Speech act profiling: a probabilistic method for analyzing persistent conversations and their participants. In *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on*, pages 10 pp.–, 2004.
13. M. Vergne, I. Morales-Ramirez, M. Morandini, A. Susi, and A. Perini. Analysing user feedback and finding experts: Can goal-orientation help? In J. Castro, J. Horkoff, N. A. M. Maiden, and E. S. K. Yu, editors, *Proceedings of the 6th International* i* *Workshop 2013, Valencia, Spain, June 17-18, 2013*, volume 978 of *CEUR Workshop Proceedings*, pages 49–54. CEUR-WS.org, 2013.
14. M. Vergne and A. Susi. Expert Finding Using Markov Networks in Open Source Communities. In *CAiSE*, number 8484 in LNCS, pages 196–210. Springer, 2014.
15. J. Zhang, M. S. Ackerman, and L. Adamic. Expertise networks in online communities: structure and algorithms. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 221–230, New York, NY, USA, 2007. ACM.

Implementing GPI, a language for Organizational Alignment

Henrique Prado Sousa, Julio Cesar Sampaio do Prado Leite

Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brasil
{hsousa, julio}@inf.puc-rio.br

Abstract. Organizational alignment aims to align strategic, tactical and operational decision levels in a given organization. The traditional operational view, focused on processes, produces gaps between strategic and operational layers. As such, many tools and techniques were developed giving more focus on one of the layers failing to promote their integration. In our current research, we are trying to identify ways to better represent organizational alignment, for an improved organizational analysis. Following in this direction, we integrated a process and a goal modeling notations (BPMN and i*) and inserted metric oriented elements, generating the GPI (Goal, Process and Indicators) language. Then a tool was created by reusing the Oryx architecture. This paper briefly present the GPI language, details its implementation in the Oryx editor and reports on ongoing research on alignment based on this infrastructure.

Keywords: GPI tool, Organizational alignment, i*, BPMN, Business process modeling, Goal modeling.

1 Introduction

Organizational alignment is a business non-functional requirement that aims to increase adherence between the operational, tactical and strategic decision levels of an organization. It means that what is defined at the strategic layer, must be fulfilled by the others layers. In order to reach that goal, business components must be in accordance towards the strategic decisions of an organization. Business alignment targets efficacy, compliance with strategic goals, whereas efficiency is usually more of an operational concern as to maximize productivity. However, filling the gap among these layers is not a trivial task, since each one has a different focus and concern. Notwithstanding the difficulties, it is important that a mapping of these layers allows for an analysis in a way to identify problems and opportunities for improvement.

Modeling business layers requires a suitable language to describe the particulars of each level. GPI, a modeling language, is an ongoing research study on a way of linking these layers offering more analysis capability. GPI is being designed to allow a smooth transition among goals, at different abstraction levels, and processes as operational implementation of these goals. As to enforce the trace among the layers and a way of evaluating effectiveness, indicators are used as a standard measure (indicators

are not detailed in this paper). A design decision took from the start was to reuse resources of languages that were already available and to adapt them to be integrated in a manner to help alignment analysis. The reuse of elements does not mean that all of their semantic will be present in GPI, once the integration between process and goal layers has its own rules (which are still under study). We have chosen BPMN and i* as the main languages to be integrated, and performed the integration by the inclusion of abstraction levels for both the goal concept as well as for the process concept. With this, a new architecture of organizational modeling was proposed. Following we will focus on the GPI language and its implementation using the meta editor Oryx.

2 GPI Language

GPI language is being designed to fill the gap not dealt by other business oriented languages, for example: lack of business elements, only process or goal models, lack of enough connection semantics, insufficient traceability, lack of alignment analysis resources and methods, and nonstandard graphical design used in notations.

To design a language that offers elements to deal with these difficulties, we defined a new language and architecture with more levels of traceability, different layers of abstraction and a goal modeling method. To do this, we are following a four step process: 1. Choose goal and process notation (done); 2. Identify similarities between notations (done); 3 – Establish integration toward alignment contribution (partially done); 4 – Evolve the language, establishing new modeling methods and operationalizations towards organizational alignment (under study).

In the first step - choose goal and process notation - instead of creating other elements to represent the same definitions existent in many consolidated notations, we decided to choose different languages for the layers, choosing among the most known and/or used languages available (based on our experience). In the goal layer we analyzed the languages: GRL [7], i* [17], NFR framework [4], Goal models of ARIS framework [12] and the UML extension proposal [6]; in the process layer: UCM [1], EPC [12], BPMN [10] and the UML extension proposal [6]. After some study we conclude that the best languages to use in this work were i* and BPMN (more details see [14]). Some justification, briefly: BPMN is an international standard and is also supported by many free modeling tools, reaching a large number of users; i* has more semantic connections between its elements (and is one of the goal languages most studied nowadays).

In the second step - identify similarities between notations - we mapped similar elements between notations. For example, an actor is similar to a pool, representing a role and a place where main business elements under his responsibility are delimited in the model; a resource is similar to an artifact, and a task is similar to an activity, representing actions executed by actors/agents. Other elements, for example, decomposition and means-end relations from i* need a case by case mapping. Some elements marked as similar between i* and BPMN may have only a partial semantic similarity, being different when considering specific states. In these cases we made the link as convenient to GPI goals. The full study of similarities can be seen in [14].

In the third step - establishing integration towards alignment contribution – we defined the integration oriented by 5W1H [9] in order to provide elements and connections enough to maintain traceability among business elements and layers. We try to answer questions based on What, Why, Where, When, Who and How, mainly when the answers are present in other layers. For example, making the following question considering a specific goal, in a goal layer: What are the partial products developed in order to reach this goal? Who are the key responsible roles of this goal? When this goal is reached? Or from operational layer: Why this role does these steps? Changing a specific activity, what goals will be affected? Then the GPI language must enable a level of traceability and detail enough to link high level elements to all low level operational elements and also to enable the model to answer 5W1H questions.

The results we achieved led to explicit elements that increased business model detail, thus leading to a new layer (aka the tactical level). It helps, for example, to the understanding of the connection between individual efforts and the overall business goals. Considering that every element only exists because of some business "goal", it is important to connect them to, for example, justify their existence, or still, correctly identify responsibilities and process execution fails.

This new layer was inserted between the level of traditional business goals and processes. This layer (for now, we call "intermediate layer") is an intermediate goal layer that is specific to represent agent/role goals, which are the closest of the operational level (Fig. 1). Indirectly, these goals (for now, we call "local goals") are the decomposition of high level goals, linked to their respective part of operational layer. The insertion of this layer helps to reduce the gap between traditional goal and process layers in the cases that their integration is made only by linking a goal direct to a process no matter what his size and complexity are. In this case, much information about how process achieves their goals is lost. It also helps to justify, from the point of view of actors, the execution of their activities.

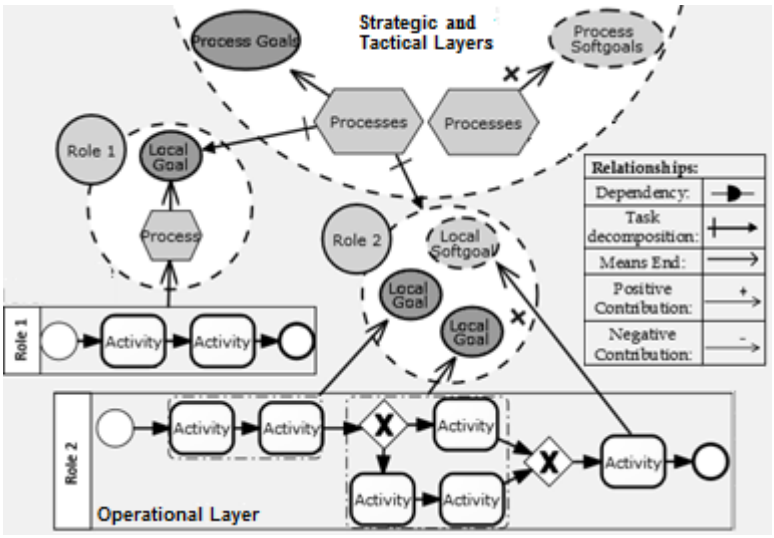


Fig. 1. Relationship of "local goals" and traceability through layers

The intermediate layer is inspired on i* Strategic Diagram ideas, considering actors of processes as agents and representing its strategic reasoning. The same idea is presented in the strategic layer, but considering the business as the actor of its high level goals and macro processes. The strategies in this layer are the high level business reasoning applied in the value added chain.

The main diagram of GPI enables the modeling of these three layer together (for now, we are calling "Integrated Diagram"), in order to create a general view that can contribute to alignment analyzing (Fig. 1).

In the fourth step - evolve the language, establishing modeling methods and operationalizations toward organizational alignment – is our current study that focuses specifically on the alignment issue. We expect that during the evaluation of the GPI we can identify elements that will help in the evolution of language, on the operationalizations that contributes to alignment and its analysis, and on the definition of a method to better take advantage of the intermediate layer.

3 GPI tool construction

The GPI language was implemented by reusing the Oryx [11] open source tool. Oryx is an academic framework with a special architecture that permits the inclusion of new notations by defining them using the Oryx language.

In this language, a notation is defined by a "Stencil set" that is composed essentially by properties of elements (i.e. id, title and description) and some pre-defined rules which establish the interaction between the objects (i.e. connection, cardinality and containment rules). Each stencil represents an object, but the stencil set is a singular file written in JSON (JavaScript Object Notation - json.org) language. Oryx also allows the definitions of "extensions" that enable the insertion or exclusion of selected stencils or a complete Stencil set (**Erro! Fonte de referência não encontrada.**).

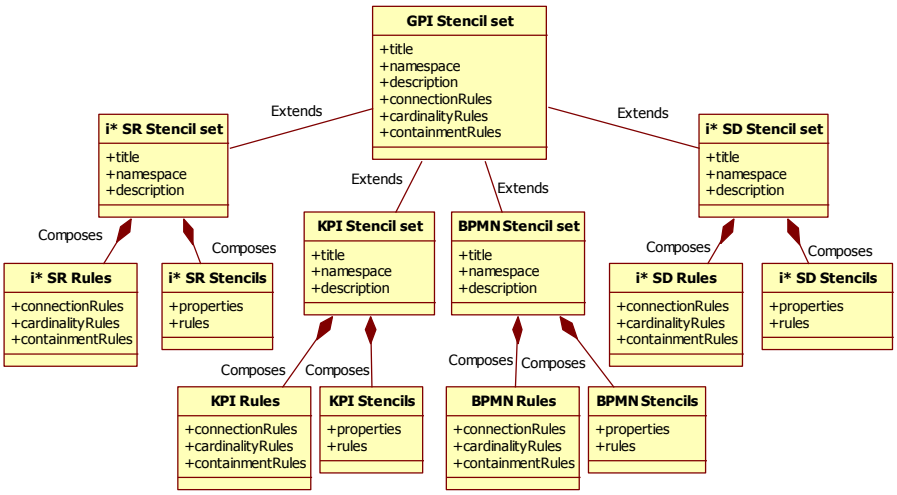


Fig. 2. Stencil set GPI and extension schema [adapted from 16]

It permits to define some "views", for example, to show basic or advanced elements. In our case, the extensions were used to allow the quick change between the languages. To organize the tool this way, a stencil set was defined to each kind of diagram involved: i*SR, i*SD, BPMN (Oryx native BPMN 2.0 stencil set was reused) and KPI. The stencil set GPI is configured to accept all others elements, then it is extended by others stencils set according the perspective selected by user. It means that "GPI tool" supports not only the Integrated Diagram, but also i* Strategic Diagram, i* Strategic Reasoning and BPMN languages. In the case of Integrated Diagram, the GPI tool enables i*SR, BPMN and KPI stencil sets, in addition to its own stencil set that contains the integration rules divided between connection, cardinality and containment rules.

The potential of Oryx tool was not completely investigated. The Oryx Core may be customized in order to insert new functionalities to the tool, like automated model analysis. This line of customization will be studied as part of our ongoing research.

4 Discussion

The GPI language introduces a new proposal of business modeling that helps to investigate alignment. Some elements, not before emphasized, are now demonstrated as an important component to interconnect goal and operational layer.

This new approach may demand a specific method of information elicitation, which we will pursue. Modeling business and goals to perform organizational alignment analysis must be improved, to reduce cost due to rework if goals are not being pursued by operational level, or pursued in an inefficient way. Then, "alignment analysis" needs to be improved by adequate methods and techniques.

Some issues also must be investigated. The complexity of GPI business models grows because more visual information is necessary. The same happens with modeling business using GPI because of the existence of a new layer. But it can be mitigated by analyzing alignment from the viewpoint of one macro goal, what will modularize the models and reduce the number of business elements. It is also possible to factor business process modeling using different methods, for example, starting it from the intermediate layer followed by goal and process layers, like the middle-out approach; or first modeling goal and process layers to after linking them by the intermediate layer, like the meet-in-the-middle approach.

The Integrated Diagram helps on making a model more transparent, mainly considering the explicit local goals and the link with business process and goals, establishing a high level of traceability and clearly contribution of each actor. It addresses the problem of making employees knowing their role in the job. With the use of "local goals", we expect to identify deviations of understanding, tacit knowledge, and lack of traceability links to activities or goals demonstrating misalignment. To complement the language, it was included in the operational layer the Business Rule and KPIs. GPI also has a proposal to use KPI in a manner to enable alignment analysis.

Other works as [1], [3], aims to help business alignment by introducing specific elements. In [1] is proposed a traceability link to interconnect goal and process (as sce-

narios); in [2], it is used a goal taxonomy in order to harmonize the goal domain to subsequently align process models. Other works [4], [8], [13] evaluate alignment by using KPIs. GPI proposal is a specific business and goal modeling language that brings up a new approach in order to alignment analysis by modeling concepts that improve traceability and detailing among layers.

References

1. Amyot, D., Mussbacher, G., "URN: Towards a New Standard for the Visual Description of Requirements". In Proceedings of SAM'2002. pp.21~37.
2. Cardoso, E.C.S., Junior, P.S.S., Almeida J.P.A.; Guizzardi, R.S.S., "Semantic Integration of Goal and Business Process Modeling", International Conference on Research and Practical Issues of Enterprise Information Systems, Natal/RN, Brazil, 2010.
3. Cardoso, E.C.S.; Guizzardi, R.S.S.; Almeida, J.P.A.; "Aligning Goal Analysis and Business Process Modelling: A Case Study in Health Care", International Journal of Business Process Integration and Management 2011, 2011.
4. Chung, L., Nixon, B., Yu, E., Mylopoulos, J.; "Non-Functional Requirements in Software Engineering" – Kluwer Academic Publishers – Massachusetts, USA, 2000.
5. del-Río-Ortega, A., Cabanillas, M. R. C.,, Cortés, A. R., "On the definition and design-time analysis of process performance indicators", Inf. Syst. 38(4): 470-490, (2013)
6. Eriksson, H-E.; Penker, M.; "Business Modeling with UML – Business Patterns at Work", John Wiley & Sons, (459 pages), ISBN: 0471295515, 2000.
7. GRL; available at "http://www.cs.toronto.edu/km/GRL", accessed in 2/2/2015.
8. Kaplan, R., Norton, D., "The balanced scorecard-measures that drive performance", Harvard Business Review 70, (1992).
9. JCS do Prado Leite, Y Yu, L Liu, SK Eric, J Mylopoulos, Quality-based software reuse, Advanced Information Systems Engineering, 2005, 535-550
10. OMG, "Business Process Model and Notation (BPMN)", Version 2.0.
11. Oryx; available at "http://Oryx-project.org/research", accessed in "15/01/2015".
12. Scheer, A.W.; "ARIS – Business Process Modeling", Berlin; Heidelberg; New York; Barcelona; Hong Kong; London; Milan; Paris; Singapore; Tokyo: Springer, 2000, 218 p. 3. ed., Bibliography: ISBN, 3-540-65835-1.
13. Shamsaei, A., Pourshahid, A., Amyot, D., "Business Process Compliance Tracking Using Key Performance Indicators", International Workshop on Business Process Design, (2010)
14. Sousa, H.P. "Integrating Intentional Modeling to Business Modeling", Master's Dissertation, Departamento de Informática, PUC-Rio, 02/2012.
15. Sousa, H. P., Leite, J.C.S.P.; "Applying Reverse Engineering and Software Reengineering in the Development of Plugins to the Oryx Tool". Monographs, PUC-Rio. 2012.
16. Sousa, H.P., Leite , J.C.S.P.; "Modeling Organizational Alignment", Conceptual Modeling, Lecture Notes in Computer Science, Yu, Eric and Dobbie, Gillian and Jarke, Matthias and Purao, Sandeep, Springer, 8824, 2014, 407-414, ISBN: 978-3-319-12205-2.
17. Yu, E.; "Modelling Strategic Relationships for Process Reengineering". Phd Thesis, Graduate Department of Computer Science, University of Toronto, Canada, 1995, pp.124.

Using Roles for OSS Adoption Strategy Models

Dolors Costal, Lidia López, Xavier Franch

GESSI Research Group, Universitat Politècnica de Catalunya (UPC), Barcelona, Spain
{dolors, llopez, franch}@essi.upc.edu

Abstract. Increasing adoption of Open Source Software (OSS) in information system engineering has led to the emergence of different OSS adoption strategies that affect and shape organizations' business models. OSS adoption strategies can be operationalized by *i** models describing the consequences of choosing each strategy. When an organization decides to adopt an OSS component, it becomes a part of the OSS ecosystem around this component. Therefore, OSS adoption strategy models need to be structured in the way to explicitly describe the role of the adopter organization within the OSS ecosystem, which may be quite different depending on the level of compromise that the organization prefers. Making visible the roles played by the different agents involved in the OSS ecosystem, the involvement of the organization in the OSS community arises naturally. This paper includes a set of roles that emerge in an OSS ecosystem and their responsibilities, and describes the issues behind the fact of using the *i* role* and *plays* constructs.

Keywords: Open-Source Software; OSS; OSS adoption; OSS ecosystem; i-star; organizational role.

1 Introduction

In the context of organizations adopting OSS components in their business strategies and business models, [1] defines a portfolio of OSS adoption strategies that embrace well-established goals, activities and resources that characterize the main aim of each strategy, as part of the FP7 European project RISCOSS (www.riscoss.eu) [2]. These OSS strategies are modeled using the *i** framework [3], which allows the connection of low-level goals referred to the adoption strategies and the high-level business goals connected to the business model.

The OSS adoption strategies models contain two actors: (1) the organization that adopts the OSS component (OSS Adopter) and (2) the OSS community that produces it. The adoption strategies are defined in order to describe the role of the OSS adopter in the OSS Community, i.e. in the OSS ecosystem around the OSS component. The rationale behind the elements inside the organization actor is identifying the goals and activities that the company needs to achieve and perform connected to the role of the organization in the OSS ecosystem. Most of the goals and tasks inside the organization actor are included because the organization becomes part of the community. This fact indicates that they should be in the OSS community actor in some way instead of in the adopter organization. The OSS Community actor should be refined making the different roles explicit.

Making these roles explicit, we would take advantage of the *i** modelling language highlighting the motivation for having some elements in the model. If an organization (*OSS Adopter*) wants to contribute to the community (goal *OSS community contributed*), it can do it for example producing code (*Developer*). Therefore, the task *maintain code* must be included in the OSS adopter rationale (Fig. 1, left side). Using roles would explicitly add the rationale for the presence of some activities, for example, the OSS Adopter needs to *maintain code* only because the organization has decided to contribute to the OSS community playing the role of *Developer*.

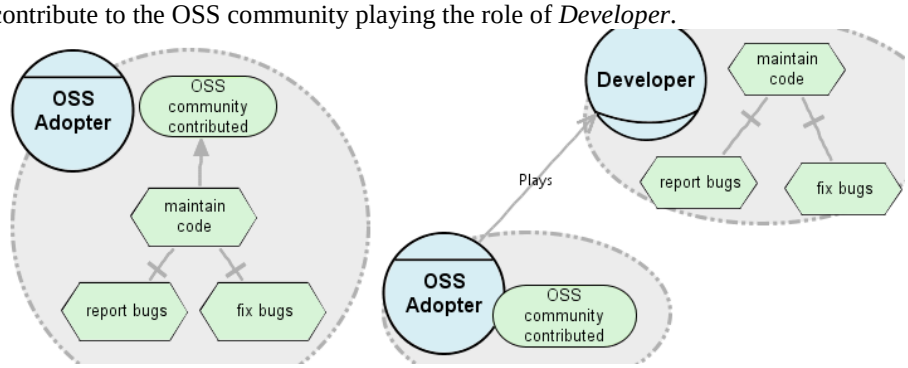


Fig. 1 OSS adoption strategy models using (right) and not using (left) *i** roles[1]

The OSS adoption strategies models catalogue presented in [1] contain all the activities modelled inside the rationale of the *OSS Adopter* actor according to its adoption strategy. That paper also uses the notion of coverage to assess which is the OSS strategy that better fits the organizational goals. The use of roles has been discussed regarding understandability versus modularity; the proposal presented in [1] prioritizes understandability (model on the left). In the work presented in this paper, we explore the use of roles in the definition of OSS adoption models, prioritizing modularity. Using roles also helps to understand the responsibilities assigned to each role, but it increases the complexity and makes the model less legible. Fig. 1 shows both alternatives. The use of roles for representing the responsibilities associated to a user group (OSS Adopters in our case) is also present in previous works like [4], where roles and the *plays* link are used to encapsulate and analyse the role's (e.g., Developer) responsibilities that are present in the agent playing the role (e.g., OSS Adopter).

The remainder of the paper is organized as follows. Section 2 introduces the OSS roles that characterize an OSS Community. Section 3 develops the main contributions of the paper including the proposal of using *i** roles in order to structure the OSS adoption strategy models. Last, Section 4 provides the open issues to be considered during the research in order to complete our proposal.

[1] The names of the intentional elements inside the actors have been modified respect to the names in the OSS adoption strategy models from [1] to improve the legibility of the models.

2 OSS Roles and Activities Ontology

In the context of the European Project RISCOSS, we have developed an ontology for supporting risk analysis in OSS ecosystems [5]. Part of this ontology contains terms for describing OSS projects. The activities and resources relevant for the interaction between an OSS adopter and an OSS community has been used for creating a set of models describing different strategies that the organizations can follow when they need to adopt an OSS component [1]. In [1] we have presented in detail the terms of the RISCOSS ontology related to the activities and resources used for developing the OSS adoption strategy models. The work presented in this paper is related to the terms defining the different roles in the OSS projects.

Table 1 contains an excerpt of the OSS roles defined in the RISCOSS ontology, for each role there is the agent than can play it (property *played by*), the decision that it can take (property *take*) and the activities that it can perform (property *perform*). The resources involved in the activities are included into brackets. Indentation in the OSS Roles column indicates the existence of a category/subcategory relationship between roles.

Table 1. OSS Roles

OSS Roles	***Properties***
Governance Role	
CommunityManager	played by Visionary
Communicator	played by Communicator
Contributor	
Administrator	take MediumTermDecision, perform MaintainSite
Developer	perform MaintainCode, FixBug, DiscussBug (BugMessage), DiscussChange (ChangeMessage)
Committer	take ShortTermDecision, perform Supportnewbies, DiscussSolution (SolutionMessage)
Disseminator	played by Distributor, perform PackageFlossForNewUsers
Documenter	perform MaintainDocumentation
Project Manager	take LongTermDecision, played by Visionary, perform Nominate
Tester	perform DiscussBug(BugMessage)
User	perform learn using OSS, ask questions

3 Using Roles to Model OSS Adoption Strategies

When an organization decides to adopt an OSS component, it may have different levels of involvement in the OSS community depending on its adoption strategy [1]. It may have a minimum involvement of only using the OSS component (OSS Acquisition and OSS Release); it may be actively involved in the OSS community by performing some activities such as reporting bugs, developing patches, etc. (OSS Integration and OSS Fork), with the purpose of gaining influence on the OSS component

evolution; or, even, it may be deeply involved in the OSS community, trying to lead it (OSS Initiative and OSS Takeover). These different levels of involvement can be modelled in a natural way using the *i* role* and *plays* constructs. More concretely:

1) As a first step, we use *i** roles to model the set of OSS roles described in Section 2 (see Table 1).
2) Then, as a second step, we use the *i** plays construct to relate an *i** agent representing an OSS adopter organization with the adequate *i** roles obtained from the first step. To choose the adequate roles for an OSS adopter organization, we take into account the activities that it performs in the OSS community according to its adoption strategy. The result of this second step is the OSS adoption strategy model of the organization.

Next two subsections illustrate these two steps.

3.1 OSS Roles as *i** Roles

*i** roles can be used to model the set of OSS roles described in Section 2 (see Table 1) and the relationships that exist between them. Fig. 2 represents these roles and their specialization (is-a) and aggregation relationships (is-part-of). To complete the roles definition, we add the activities that each OSS role performs represented as *i** tasks in the corresponding *i** role SR diagram (not shown in Fig. 2 for space reasons). For example, the role *User* has as tasks: *learn using OSS, ask questions*.

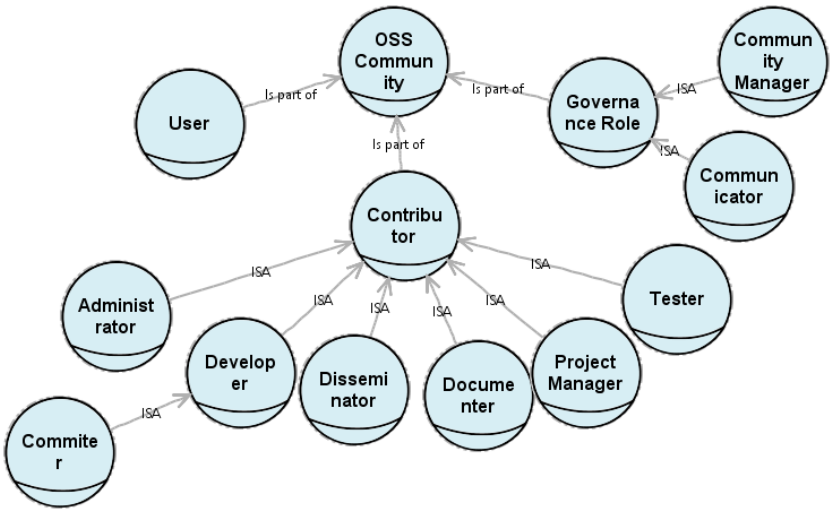


Fig. 2 OSS roles as *i** roles

The complete OSS ecosystem model, including dependencies between roles, has been obtained by using an adaptation of the RiSD methodology presented in [6] for systematic construction of i* SD models for ecosystems.

3.2 Use of *i** Roles and *i** Plays for OSS Adoption Strategy Modelling

Once we have the *i** roles and their SR diagrams, the *i** plays construct can be used to relate an agent representing the OSS adopter organization to them. For instance, if we have an OSS adopter whose adoption strategy is OSS Acquisition, which consists on having a minimum involvement in the OSS community meaning that it basically wants to use the component, this adoption strategy can be simply modelled by means of defining a plays relationship between the agent that represents the OSS adopter and the *User i** role (no more role are played by the adopter in this strategy). Fig. 3 illustrates this case.

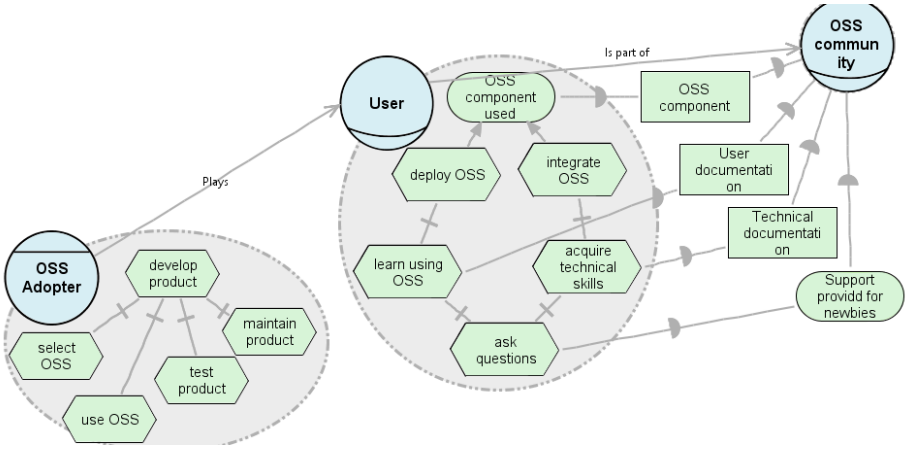


Fig. 3 Using *i* plays* construct to relate the OSS adopter and User

4 Open Issues

There are some open issues related to the presented proposal of using roles to model OSS adoption strategies.

First, the implications of using the *i** plays construct arise some questions over its related agents and roles that need clarification. For example: Does the agent that plays a role inherit all the dependencies defined on the role? Are the agent's goals exactly the same as those of the role? May the agent have additional goals? May the agent get rid of some role's goals?

Second, we have used the is-part-of relationship to relate, for example, the *User* role and the *OSS community* role (see Fig. 2 and Fig. 3). Therefore, when applying the adoption strategy model to a specific OSS community, e.g. the *Eclipse community*, we would need to replace *OSS community* by *Eclipse community* in the model and, thus, represent that the *User* role is part of the *Eclipse community*. However, this is not straightforward, since the is-part-of relationship is required to relate only actors belonging to the same type [7], and we have that *User* is a role while the *Eclipse community* is an actor that seems more realistic to model as an agent than as a role.

Finally, not all the goals of an OSS adopter, according to the [1] catalogue of OSS adoption strategy models, can be located as goals of the OSS roles (roles depicted in Fig. 2) because not all the adopters that will apply a specific role will share them. For instance, considering the OSS Acquisition adoption strategy defined in [1]:

1) The acquisition adoption strategy includes the following goals for the OSS adopter organization: *Take benefit from OSS Community*, *OSS involvement minimised* and *Do not care about OSS evolution for maintenance*.
2) The role played by an adopter applying the acquisition strategy would be that of *User* (it will apply only this role).
3) The *User* role is played also by adopters applying other adoption strategies such as, for example, the OSS Integration adoption strategy [1].
4) The goals listed in (1) cannot be placed in the *User* boundary since not all the adopters playing the *User* role will share those goals. In particular, adopters applying the integration strategy do not share the goals *OSS involvement minimised* and *Do not care about OSS evolution for maintenance.*

Acknowledgements

This work is a result of the RISCOSS project, funded by the EC 7th Framework Programme FP7/2007-2013, agreement number 318249.

References

[1] López, L.; Costal, D.; Ayala, C.; Franch, X.; Annosi, M.C.; Glott, R.; Haaland, K.: Adoption of OSS components: A goal-oriented approach. In Data & Knowlegde Enginering. 2015. http://dx.doi.org/10.1016/j.datak.2015.06

[2] Franch, X., et al.: Managing Risk in Open Source Software Adoption. In International Conference on Software Engineering and Applications (ICSOFT 2013), pp. 258-264.

[3] Yu, E.: Modelling Strategic Relationships for Process Reengineering. PhD. thesis, Toronto, 1995.

[4] Sun, J.; Liu, F.; Zhang, H.; Liu, L.; Yu, E.: Understanding the Diversity of Services Based on Users' Identities. In International Conference on Advanced Information Systems Engineering (CAiSE 2011), pp. 612-626.

[5] C. Ayala, L. López, D1.3 Modeling Support (consolidated version), Technical Report, RISCOSS FP7 project, 2014.

[6] Franch, X., Grau, G., Mayol, E., Quer, C., Ayala, C., Cares, C., Navarrete, F., Haya, M., Botella, P.: Systematic Construction of i* Strategic Dependency Models for Socio-technical Systems. In International Journal of Software Engineering and Knowledge Engineering (IJSEKE), 2007, Vol. 17(1), pp.79-106

[7] *i** Guide, ISA Association, istarwiki.org

An Initial Approach to Reuse Non-Functional Requirements Knowledge

Rodrigo Veleda, Luiz Marcio Cysneiros
School of Information Technology – York University, Canada
rveleda@yorku.ca, cysneiro@yorku.ca

Abstract. Non-Functional Requirements (NFR) can be seen as qualities that software should deliver to cope with the stakeholders' demands. NFRs are fuzzy in nature and hence hard to identify. Despite the fact that both developers and users may value NFRs, they frequently do not identify the need for an NFR. Even when an NFR is identified as required, possible solutions to implement this NFR may be hard to figure out. Furthermore, interdependencies among NFRs may implicate that a solution for one NFR may, at the same time, bring synergy to one NFR while conflicting with another. One approach to deal with that is to use Softgoal Interdependency Graphs (SIG) to capture knowledge describing alternatives to implement NFRs. We have obtained empirical evidence that using catalogues can help eliciting NFRs despite the fact that catalogues do not scale too well. To address this question, we have investigated the use of ontologies and semantic web techniques to represent SIGs in a machine readable format. We have produced a tool (NDR) that starts to use these concepts. In its current form, the NDR tool only allows very basic queries done manually. The NDR tool is part of the NDR framework which will facilitate the reuse of NFR knowledge on Alternatives to incorporate NFRs into the design of target systems.

Keywords: Non-Functional Requirements, Reuse, Knowledge

1 Introduction.

Requirements engineers have to address both functional and non-functional requirements to develop software systems [1]. Functional requirements are responsible to represent what the system is capable of in terms of available features. On the other hand, non-functional requirements are known to represent quality attributes [2, 3]. These quality characteristics include privacy, performance, usability and other similar aspects related to the quality of a software system.

The first challenge for eliciting NFRs lies on the fact that they are fuzzy in nature and quite frequently are missed both by software engineers and stakeholders. Furthermore, choosing one solution to implement one NFR might bring synergies and perhaps most importantly conflicts to another NFR bringing the perception that one NFR can rarely be expected to be 100% satisfied. We use the term *satisfice* [4, 5] to represent the idea that NFR is satisfied within acceptable limits.

Some works have proposed the use of catalogues representing knowledge to *satisfice* NFRs as a way of helping not only to elicit NFRs but also to reason about the complexity involved in choosing alternatives to *satisfice* an NFR [4], [6]. In fact, empirical work has suggested that the use of catalogues can contribute to avoiding omissions and missed conflicts, despite the fact that SIGs do not scale too well [6].

These catalogues are implemented using Softgoal Interdependency Graphs (SIG) [4]. SIG catalogues promote a graphical representation of essential quality characteristics for

satisficing a given non-functional requirement. SIGs also demonstrate possible tradeoffs among non-functional requirements in the target system.

In this paper, we discuss an ongoing research by introducing the NDR Tool. The NDR Tool is currently under development, and it is part of the NDR Framework which aims to facilitate the reuse of non-functional requirements knowledge captured in SIGs. Our application strives the extraction of present knowledge from SIGs and represents it in a machine readable format using ontologies and semantic web techniques [7, 8] Furthermore, the tool proposes the storage of collected knowledge within an ontology repository that currently follows the proposed Non-functional requirement and Design Rationale (NDR) Ontology [7]. We are developing the NDR tool to store as many alternatives as possible to *satisfice* NFRs. It will also use RDF [9] queries for retrieving alternatives for one specific problem, allowing software engineers to select one alternative and import this alternative to its *i** models representing the target system. This work depicts the tool's currently available features and also denotes the potential challenges and future tasks for implementation.

The remainder of this paper is structured as follows: Section 2 illustrates the related work. Section 3 describes the objectives of our research and its scientific contribution. Finally, Section 4 presents the on-going and future work.

2 Related Work

Some works [10, 11] focus on experienced-based elicitation and recommendation for the use of non-functional requirements in software service. Others [12, 13, 14, 15], aim the use of ontologies to assist non-functional requirements elicitation. Nevertheless, none of these proposed works addresses the challenge of investigating the potential tradeoffs between multiple non-functional requirements. Nor they interact with *i** tools to facilitate the reuse of the knowledge.

Considering the use of SIGs aiming the reuse of knowledge, Sancho et al. [16] proposes an ontological database. Their work consists of two ontologies both written in OWL [17]: The NFR ontology and the SIG ontology. The NFR ontology explains the NFRs concept and relationship among them. The SIG ontology depicts SIG constructs and their relationships. We have identified two shortcomings within this approach. First, the SIG ontology does not define any class to describe the Correlation interdependency between Softgoals therefore limiting reasoning involving more than one NFR. Second, it does not enforce the use of proper kind of Softgoals as parent and offspring of each Refinement. The NDR ontology is based on this work.

Hazeem et. al [13] introduced the ElicitO. ElicitO is an ontology-based tool that supports non-functional requirements elicitation by providing a knowledge base that relates non-functional requirements and its associated metrics. Also, Najera [14] highlights an approach that uses OWL and RDF targeting the representation of *i** variants. Unfortunately, the reuse of non-functional requirements knowledge is not tackled in this work nor is cited as future work.

3 Research objectives and scientific contribution

Our long-term goal is to develop a framework that can help software engineers to elicit and model non-functional requirements empowered by the knowledge that has previously been elicited and validated. We believe that the use of a well-defined knowledge base could play an essential role to achieve this goal. Therefore, our environment will emerge

as the result of further developing our ontology and tools to store and retrieve knowledge on *satisficing* non-functional requirements.

Hence, our first goal is to develop further the NDR tool to efficiently store NFR knowledge while allowing querying at different levels of granularity for retrieving existing information. .

The next step will be to develop mechanisms to import and export knowledge from and to other Tools that support *i**. Techniques to import SIGs from *i** tools will be implemented as well as the ability for choosing alternatives from existing SIGs in the NDR tool to be imported into i* models expressed in tools such as jUCMNav [18].

At first, this environment will only be open to accepting queries from the academic community. However, in a near future we envision to accept contributions from other research groups working with NFR knowledge to enrich the existing knowledge base. At the same time, we will also allow members from the industry to query the knowledge base and submit comments with their perception. At a later stage, contributions to add to the knowledge base will be accepted from a broad audience.

4 Ongoing and Future work

4.1 NDR Ontology

The NDR Ontology proposed in [7] represents NFRs and design argumentative rationale knowledge in a machine-readable format. This representation follows the proposed standards of OWL. Therefore, each examined SIG catalog is converted into semantic graphs, establishing new sets of instances of NDR Ontology and expressing a machine-readable form. In operational terms, RDF is widely used to describe ontologies (mainly at semantically enriched Web sites). RDF encodes information as triplets (resources) that relate a property to other resources or plain literal data. Thus, RDF models are directed labeled graphs that allow representing meaningful contents. RDF Schema (RDFS) [19] allows describing properties and classes of RDF resources and supports a generalization hierarchy for properties and classes. As a short-term goal, we will further develop the ontology and the tool capability of producing the necessary SPARQL queries [20] from user-friendly dialogs.

In our proposed approach, we envision to provide the NDR Ontology available within the NDR Tool in a cloud environment. We believe that making our proposed approach available in a cloud will facilitate not only its use by different audiences but also the opportunity to receive contributions to enlarge the knowledge base that will be available.

Currently, we have implemented a proof of concept version of the NDR Ontology on our cloud ontology repository. In order to have a user-oriented layer displaying details of the target ontology graphically, we have integrated our platform with WebVOWL [21]. The WebVOWL is a web-application that implements the Visual Notation for OWL Ontologies (VOWL) [22]. A graphical visualization of the NDR Ontology version is currently implemented in

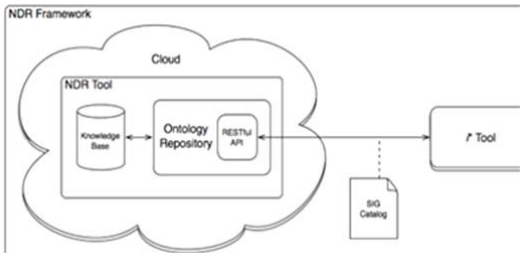


Fig. 1. The NDR Framework Architecture

our platform[1]. We are in the process of importing SIG catalogs into the target environment to also demonstrate the knowledge expansion graphically.

4.2 NDR Framework Conceptual Architecture

To maintain and assure a reliable integration between SIGs developed with *i** Tools and its conversion into ontology knowledge, we propose the NDR Framework. Figure 1 illustrates its conceptual architecture.

The NDR Tool is mainly composed of a knowledge base and an ontology repository. As mentioned in Section 4.1, currently, only the NDR Ontology is implemented within the platform. We aim to develop a generic ontology repository that can be instantiated in domain specific ontologies to provide extensibility of our platform. In other words, the NDR Tool will be able to handle several ontologies, preserving a valuable and vast knowledge base.

Aside from holding ontologies, the NDR Tool will also be in charge of handling the conversion of SIG catalogs into ontology instances. Besides the execution of parsers designed for each type of supported SIG and ontology, our platform will detect if the artefact that is being provided contains relevant knowledge based on definitions manually defined by repository administrators. An approach based on Open-Source concepts will be developed to handle this.

Access to the knowledge contained in the NDR Tool will be possible through the use of web services. We envision to implement RESTful web service endpoints that can be invoked externally by third-party *i** applications. Essential features such as artefact importation and knowledge retrieval will be implemented within these services, facilitating future integrations.

To illustrate an appropriate real-world example of the applicability of the NDR Framework, we portrait a SIG representing the non-functional requirement of Transparency as demonstrated in Figure 2 is uploaded into our framework. The NDR Tool will extract the knowledge in the provided SIG based on the settings manually defined by the repository administrator.

Then, the tool will convert the selected knowledge into a machine-readable format, following the NDR Ontology standards.

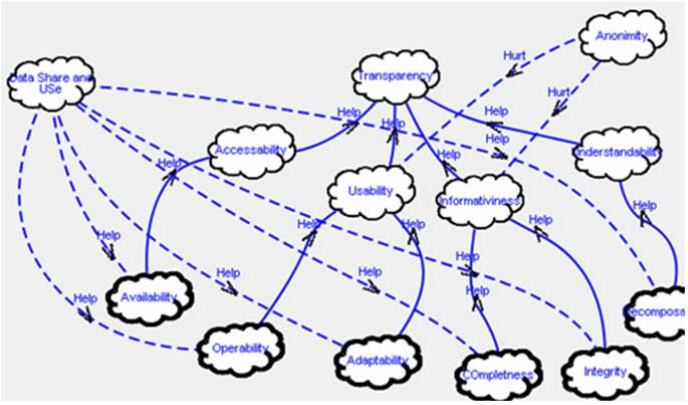


Fig. 2. Transparency SIG

As it is noticeable in Figure 2, the visual information represented in the SIG illustrates the scalability problem mentioned earlier in this paper. As more details such as correlation and decompositions are added into the SIG, its understandability

[1] http://132.206.206.138:8888/

and clarity becomes affected.

By having the knowledge ready in a machine-readable format, the NDR tool will manage to update the ontology with new individual instances and persist the modifications into a database. At this point, the extracted knowledge from the Transparency SIG is available for reuse.

The reuse of stored knowledge will be possible by accessing web service endpoints. Once an endpoint is reached, the NDR Tool will handle the request by applying the SPARQL queries over the stored knowledge. For instance, a software engineer wants to know which non-functional requirements are directly related to the *satisficing* of Transparency. After receiving this request, the NDR Tool will execute an SPARQL query similar to the following:

```
SELECT DISTINCT ?interlinkId ?softgoalParent ?softgoalSpring
?contributionKind WHERE {?interlinkId rdf:type ndr:Correlation.
?interlinkId ndr:correlationHead ?softgoalParent. ?interlinkId
ndr:correlationTail ?softgoalSpring. ?interlinkId ndr:contributionKind
?contributionKind.}
```

Basically, this query is selecting all the correlations that somehow affect the *satisficing* of the Transparency *softgoal*. Table 1 demonstrates the internal result of this query execution.

Table 1. SPARQL query execution result

interlinkId	softgoalParent	softgoalSpring	contribution
ndr:UH_correlation2	ndr:Informativiness	ndr:Anonymity	ndr:Hurt
ndr:UH_correlation7	ndr:Integrity	ndr:Data_Share_and_Use	ndr:Help
ndr:UH_correlation1	ndr:Usability	ndr:Anonymity	ndr:Hurt
ndr:UH_correlation6	ndr:Completeness	ndr:Data_Share_and_Use	ndr:Help
ndr:UH_correlation4	ndr:Operability	ndr:Data_Share_and_Use	ndr:Help
ndr:UH_correlation8	ndr:Decomposability	ndr:Data_Share_and_Use	ndr:Help
ndr:UH_correlation5	ndr:Adaptability	ndr:Data_Share_and_Use	ndr:Help
ndr:UH_correlation3	ndr:Availability	ndr:Data_Share_and_Use	ndr:Help

As an outcome of this process, the NDR Tool retrieves the requested information and the result is ready to be sent back to the user. It is noteworthy to mention that the possibility of having results in a graphical way instead of machine-readable format will depend on the level of integration with a given *i** tool.

4.3 jUCMNav Integration

We aim at integrating the NDR Tool with jUCMNav to have SIG catalogs imported/exported into/from our platform.

jUCMNav is an open-source modeling tool that supports the *i** Framework. One of the reasons that we decided to integrate our approach with jUCMNav is its extensibility. Another reason is that jUCMNav is a cross-platform application endeavor. It is well documented and presents a steady process of growth. Lastly, by integrating the NDR tool to jUCMNav, we can provide a graphical visualization to resulting SIGs from queries. Although we will be mainly focusing on jUCMNav at first, all our efforts will keep in mind the need to develop an interactive approach that can work with as many *i** tools as possible.

References.

1. Chung, L., do Prado Leite, J.: On Non-Functional Requirements in Software Engineering. In: Borgida, A., Chaudhri, V., Giorgini, P., and Yu, E. (eds.) Conceptual Modeling: Foundations and Applications. pp. 363–379. Springer Berlin Heidelberg (2009)
2. Boehm, B.W., Brown, J.R., Kaspar, H., Lipow, M.: Characteristics of software quality. North-Holland, Amsterdam (1978)

3. Keller, S.E., Kahn, L.G., Panara, R.B.: Specifying software quality requirements with metrics. System and Software Requirements Engineering. 145–163 (1990)
4. Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. Springer US (1999)
5. Simon, H.A.: The sciences of the artificial. (1996)
6. Cysneiros, L.M.: Evaluating the Effectiveness of using Catalogues to Elicit Non-Functional Requirements. In: Proc. of 10th Workshop in Requirements Engineering, pp. 107–115 (2007)
7. Lopez, C., Cysneiros, L.M., Astudillo, H.: NDR Ontology: Sharing and Reusing NFR and Design Rationale Knowledge. Managing Requirements Knowledge, 2008. MARK '08. First International Workshop on. pp. 1–10 (2008)
8. López, C., Inostroza, P., Cysneiros, L.M., Astudillo, H.: Visualization and comparison of architecture rationale with semantic web technologies. Journal of Systems and Software. 82, 1198 – 1210 (2009)
9. Carroll, J.J, Klyne, G.: RDF concepts and abstract syntax, http://www.w3.org/TR/rdf-concepts/ (2004)
10. Doerr, J., Kerkow, D., Koenig, T., Olsson, T., Suzuki, T.: Non-functional requirements in industry - three case studies adopting an experience-based NFR method. Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on. pp. 373–382 (2005)
11. Zhang, X.-L., Chi, C.-H., Ding, C., Wong, R.K.: Non-functional Requirement Analysis and Recommendation for Software Services. Web Services (ICWS), 2013 IEEE 20th International Conference on. pp. 555–562 (2013)
12. Wang, T., Si, Y., Xuan, X., Wang, X., Yang, X., Li, S., Kavs, A.J.: A QoS Ontology Cooperated with Feature Models for Non-functional Requirements Elicitation. Proceedings of the Second Asia-Pacific Symposium on Internetware. pp. 17:1–17:4. ACM, New York, NY, USA (2010)
13. Al Balushi, T., Sampaio, P.F., Dabhi, D., Loucopoulos, P.: ElicitO: A Quality Ontology-Guided NFR Elicitation Tool. In: Sawyer, P., Paech, B., and Heymans, P. (eds.) Requirements Engineering: Foundation for Software Quality. pp. 306–319. Springer Berlin Heidelberg (2007)
14. Najera, K., Martinez, A., Perini, A., Estrada, H.: An Ontology-Based Methodology for Integrating i* Variants. Presented at the June (2013)
15. Guizzardi, R., Li, F.-L., Borgida, A., Guizzardi, G., Horkoff, J., Mylopoulos, J.: An ontological interpretation of non-functional requirements. Presented at the Formal Ontology in Information Systems: Proceedings of the Eighth International Conference (FOIS 2014) (2014).
16. Sancho, P.P., Juiz, C., Puigjaner, R., Chung, L., Subramanian, N.: An Approach to Ontology-aided Performance Engineering Through NFR Framework. Proceedings of the 6th International Workshop on Software and Performance. pp. 125–128. ACM, New York, NY, USA (2007)
17. McGuinness, D.L., van Harmelen, F.: OWL Web Ontology Language Overview. W3C Recommendation, http://www.w3.org/TR/2004/REC-owl-features-20040210/ (2004)
18. Mussbacher, G., Amyot, D.: Goal and scenario modeling, analysis, and transformation with jUCMNav. Software Engineering - Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on. pp. 431–432 (2009)
19. Brickley D., Guha,R.: RDF vocabulary description language 1.0: RDF Schema, W3C working draft, http://www.w3.org/TR/2002/WD-rdf-schema-20021112/ (2002).
20. Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF.
21. Lohmann, S., Link, V., Marbach, E., Negru, S.: WebVOWL: Web-based Visualization of Ontologies. In: Lambrix, P., Hyvönen, E., Blomqvist, E., Presutti, V., Qi, G., Sattler, U., Ding, Y., and Ghidini, C. (eds.) Knowledge Engineering and Knowledge Management. pp. 154–158. Springer International Publishing (2015)
22. Negru, S., Lohmann, S.: A Visual Notation for the Integrated Representation of OWL Ontologies. Proceedings of the 9th International Conference on Web Information Systems and Technologies (WEBIST '13). pp. 308–315. SciTePress (2013)

From Unknown to Known Impacts of Organizational Changes on Socio-technical Systems

Marília Guterres Ferreira[1, 2], Neil Maiden[2,] Julio Cesar Sampaio do Prado Leite[1]

[1]Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), Rio de Janeiro, Brasil
{mferreira, jcspl}@inf.puc-rio.br
[2]City University London, London, United Kingdom
n.a.m.maiden@city.ac.uk

Abstract. Keeping organizations and their Socio-technical System (STS) aligned over time is a complex endeavour. We believe understanding the organizational dynamics of changes, and of the impacts these changes will have, can support the evolution of STSs. Reasoning on the organizational changes in advance also supports the development of an STS more likely to be aligned to the dynamics of the organization. This work presents the design and the application of a Dynamic Organizational Framework (DOF), constituted of a dynamic organizational model (DOM) on which to base the reasoning, a database of questions (DBQ) to explore possible organizational impacts, and a method to reason on changes and impacts within goal models. We apply this framework to analyse the impacts of the introduction of a system into the customers' attendance process in a Post Office in London. First results show contributions towards to the awareness about the organization, and to the quality and accuracy of requirements.

Keywords: Socio-technical systems, Software Evolution, Requirements Engineering, Organizational Model, Organizational Alignment, Goal Modelling, Goal Analysis.

1 Introduction

In order to enhance their performance in a rapidly changing environment, organizations continuously change, frequently, guided by strategic management plans. In this setting, organizational change creates new requirements for the deployed socio-technical system (STS), which, in turn, may also change the organization [1]. Over time, an STS presents inconsistencies and lack of compliance with new environmental requirements in which it was deployed, i.e. activities and business processes through which the organization intends to generate value; in other words, its business strategy. This lack of compliance is due to unforeseen impacts and demands the evolution of the STS which is a difficult, complex, costly, and time-consuming process. Our research aims to support stakeholders and organizational analysts in understanding likely organizational impacts of proposed organizational changes, as strategic changes, and then gain insights and reasoning on the impacts of these changes on the

STS' requirements evolution. Therefore we propose a Dynamic Organizational Framework (DOF) constituted of a sequence of procedures supported by a Dynamic Organizational Model (DOM), to understand organizational flow of impacts, and by a Database of Questions (DBQ), to elicit knowledge from organizational analysts. These techniques are meant to be used on information acquisition within the context of *goal* and *scenario modelling.* We use goal-oriented requirements engineering, specifically the i* framework, because it is suitable for modelling and analysis in requirements engineering, then we can model and understand stakeholders' underlying motivations for systems, identify the relation between the system and the organizational and business context, clarify and capture organizational changes, impacts and requirements from the analysis [2].

In order to augment our knowledge, we also apply scenario walkthroughs into the organizational impacts to analyse and capture requirements. The idea behind these impact scenarios walkthroughs is that people are better at identifying facts of commission rather than omission. From this, impact scenario walkthroughs offer stakeholders support to think about most likely impacts of organizational changes. If the identified impact is relevant to the system being specified but not yet handled in the specification, then a potential requirement change has been identified, and it is suggested to the developers to acquire and document the relevant requirements [3].

2 The Dynamic Organizational Framework

A project introduces a new STS (the designed thing) into the organization (the environment) and this introduction generates impact on the organization. Thus, the Dynamic Organizational Framework aims to support the elicitation of organizational changes and reasoning about potential impacts on and from both the organizational and the STS. Hence, it is constituted of a sequence of activities assisted by a Dynamic Organizational Model (DOM) and by a Database of Questions (DBQ). These support tools were developed through extensive literature review, application in real cases and recurrent refinements, summarized as follows. First, to understand the flow of changes and impacts in organizations, we initially must understand the organization itself. Hence, we based our model on Jay Galbraith's Star Model, the most widely-used and accepted organizational design framework [4]. This model relies on the following five *dimensions* of an organization: *Strategy:* determines the direction of the organization; *Structure:* defines the placement of power and authority in the organization, the location of decision-making power; *Processes:* outlines the flow of information, cut across an organization structure and determines its functioning; *Rewards:* influences the motivation of people to perform and address organizational goals; *People:* defines and influences the employees' mind-sets and skills to implement the company's chosen direction. Through exploratory literature review and applications in real experiences, we identified *elements* for each of the five ends of the Star Model. The resulting first version of the organizational model, then formed by organizational *dimensions* and respective *elements*, was used as a base in a workshop to discuss organizational changes brought up by a Learning Management System in a University.

Data from the workshop showed a *flow of changes and impacts* within organizational dimensions and the consequent need to incorporate organizational dynamics in the model.

Therefore, we conflated our model to the Configuration Model of Organizational Culture (CMOC) [5], making the necessary amendments. Besides dynamic relationships, the CMOC also maps interactions from the organization with the external environment, which demanded more research on their respective elements. Our final DOM is depicted in Fig. 1. Now, each organizational dimension is connected by *flow of impacts* (arrows left-to-right) and *flow of adjustments* (arrows right-to-left).

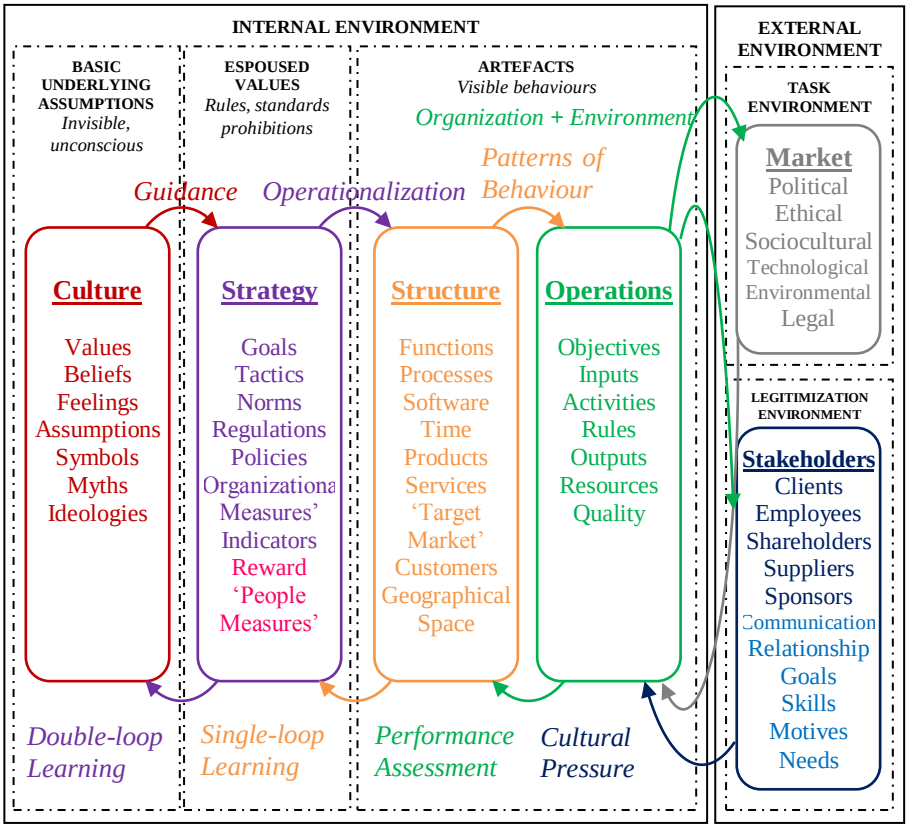


Fig. 1. Dynamic Organizational Model (DOM)

In order to elicit knowledge from the stakeholders about organizational changes and impacts, we constructed a database consisting of *88 questions* (DBQ). These questions are grounded on the organizational *elements* and organized in *10 sets*, corresponding to the *5 flows of impacts* (arrows left-to-right in Fig. 1) and to the *5 flows of adjustments* (arrows right-to-left in Fig. 1) within the organizational *dimensions*. For example, consider the generic organizational change *"Sell new product X"*. First we identify the organizational *dimension* which better fits it: this is a new *Strategy* and by it, we start our *flow of reasoning* according to the flow of impacts on the dimension

Structure. For that, we apply questions from the set: *Operationalization*. One of the questions is: "*What new processes are needed to implement this new strategy?*". Possible answer: "*Sell product X*". Then, flow of impacts on *Operations*, set *Patterns of Behaviours*: "*What are the activities needed to this new process?*", answer: definition of specific activities. Following, flow of impacts on *Stakeholders*, set *Legitimacy Management*: "*What skills are needed from the employees closer to the change?*", identification of necessary skills. Now, we can start a flow of adjustments, regarding the findings. Set: *Cultural Pressure*: "*What operational adjustments are needed to satisfy employees' goals?*". Flow of adjustment, set *Performance Assessment*: "*How does the new functions relate to existing functions?*". And so on.

We summarized the procedure steps of the DOF as follows:

1. **Stimulating Organizational Awareness:** to boost organizational awareness, requirements engineers and organizational analysts model the *As Is* and start modelling the *To Be* contexts using i*.
2. **Identification of organizational changes:** From the comparisons between the models, the participants identify the organizational changes (new elements in the To Be models) between the two contexts.

For each organizational change (new element):

3. **Identification of the type of change:** (i) Participants decide on one change (new element); (ii) Using the DOM, participants chose one *organizational dimension* that better represents the change (strategic, structural, operational, related to people, related to market, or cultural);
4. **Identification of the flow of reasoning to follow:** in order to stimulate a natural flow of reasoning, for each change participants can choose from either the *flow of impacts* or the *flow of adjustment*, according to their own insights regarding the DOM.
5. **Identification of impacts:** (i) According to the type of change and to the chosen flow of reasoning, participants use the *questions* from the matching set in the DBQ to identify the likely organizational impacts. (ii) When necessary, to facilitate the reasoning of the participants, they construct *As Is* and *To Be scenarios* of key use cases of the future system corresponding to the previously identified organizational change. (iii) By (vertically) *walking through the scenarios*, once identified changes between them, participants (horizontally) apply the questions, annotate the organizational changes (the answers of the questions) and the organizational impacts following the flow of reasoning they came up.
6. **Identification of requirements changes:** then, from the identification of likely organizational impacts, participants analyse the possible *impacts on the STS'* requirements.

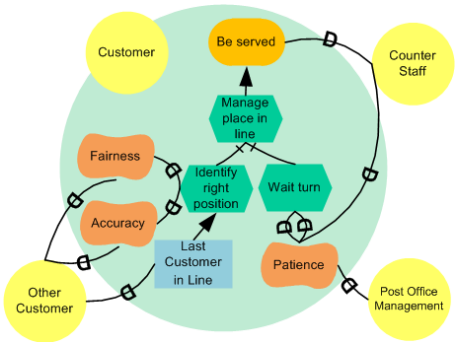


Fig. 2: SR of Post Office (*As Is*)

The procedure ends when analysts are satisfied with the exploration of likely impacts. The flow of reasoning can follow

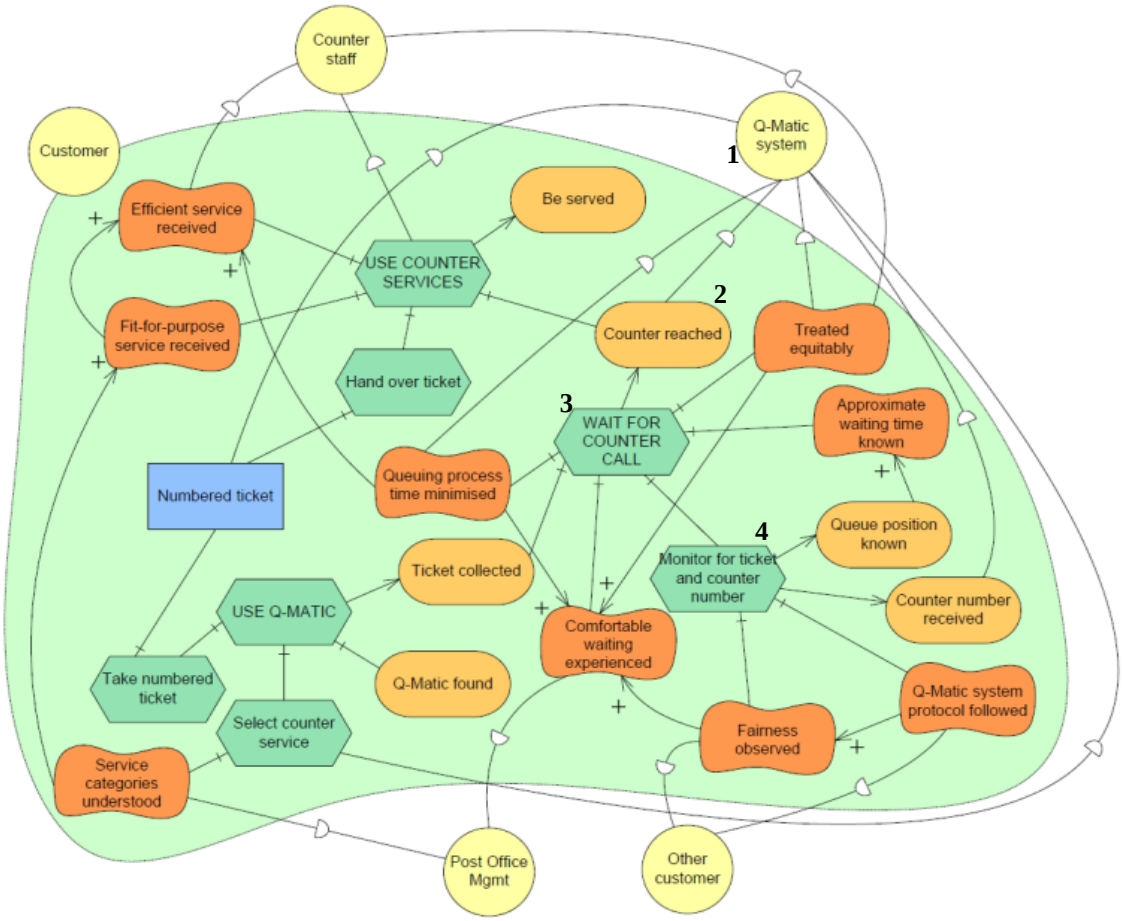


Fig. 3: Strategic Rationale of Post Office (To Be) (with IDs presented in Table 1)

unlimited flows of impacts and flow of adjustments since one change may bring infinite impacts in different organizational dimensions. The last version of the DOF was applied in a real case of organizational change occurred in a Post Office in London, and the method of application is as follows: we present the As Is SR model in Fig. 2 and the To Be in Fig. 3. A summarized flow of reasoning is illustrated in Table 1.

By the end of the study of the Post Office case, the authors identified 18 main changes, explored 6 different flows of impacts, and identified 51 possible organizational changes and consequent 40 STS' requirements changes, which if implemented correctly, will minimise undesirable effects of the impacts. The abstraction level of the requirements varied, for example, we found a need for entire software to support new services, as "Post and Go", and we pointed 10 different specific indicators to be extracted from data gathered by the STS. As the DOF is based on the participant's reasoning, the results and flows of impacts diverse from participant to participant, since it is a representation of the perceptions of the person to whom the DOF is being applied.

3 Conclusion

In this paper we presented our Dynamic Organizational Framework (DOF) to elicit and reason about organizational changes and impacts within goal models and scenario

walkthroughs so that stakeholders can analyse the consequent impacts on STS requirements. First results show contributions towards requirements quality and accuracy; it brings a better understanding of organizational dimensions, elements and impacts of organizational changes, contributes to organizational learning and consequently enables the development of more powerful STS. In the future, we are going to validate this proposal in other cases; make a thorough comparison with related researches [6]; extend the model to address impacts on external organizations; develop a tool to support the DOF; study creative techniques to boost thinking about impacts; and apply the DOF to analyse the relationship between Software Transparency and Power Dynamics in Organizations.

Table 1. Summarized rationale of the application of DOF

ID	Type of change	Direction of Flow	Question	Organizational Answer	Impact	Requirement
	i*					
1	**New system** (Structural)	Patterns of Behaviour	How does this structure relate to the *objectives* of the system?	Controls attendance of customers by counter staff (Be served || counter reached)	New goal (2)	N. A.
2	**New Goal** (Operational)	Patterns of Behaviour	Is any *activity* needed?	The customer should wait for counter call	New task (3)	STS shall call the customer by number.
3	**New Task** (Operational)	Patterns of Behaviour	Is any *activity* needed?	The customer should monitor for ticket and counter number	New task (4)	STS shall print the ticket with the queue number accurately.
Impact Scenarios Walkthrough (continuation)						
Scenario	**New Quality** = *Transparency of the Queue*	Single-loop learning	Does this new structure bring new *organizational measures*?	Total amount of customers: .[day|month|year]. .served on counter .using services. .buying products.	Now it is possible to control the flow of the queue.	STS shall calculate the total amount of customers: .[day|month|year]. .served on counter .using services. .buying products.

References

1. Sousa, H. P., & Leite, J. S. (2014). Modeling Organizational Alignment. In S. I. Publishing (Ed.), *Conceptual Modeling*, (pp. 407-414).
2. Horkoff, J., & Yu, E. (2014). Interactive goal model analysis for early requirements engineering. *Requirements Engineering*, (pp. 1-33).
3. Alexander, I. F., & Maiden, N. (Eds.). (2005). *Scenarios, Stories, Use Cases: Through the Systems Development Life-Cycle.* John Wiley & Sons Inc.
4. Jay, R. G. (2011). *Designing the customer-centric organization: A guide to strategy, structure, and process.* John Wiley & Sons.
5. Dauber, D., Gerhard, F., & Yolles, M. (2012). A Configuration Model of Organizational Culture. *SAGE Open 2.1 DOI: 10.1177/2158244012441482*, 1-16.
6. Danesh, M. H., & Yu, E. (2014). Architecting Enterprise Capabilities: Creating Dynamic Capabilities from IT and Software Architecture. *Seventh International i* Workshop. 1157.* CEUR.

Supporting Creative RE with *i**

Jennifer Horkoff, Neil Maiden

City University London, UK – horkoff,n.a.m.maiden@city.ac.uk

Abstract. Successful software must be both useful and innovative. Techniques for Requirements Engineering (RE) have mainly focused on utility, with a prominent body of work using goal modeling and analysis to ensure that systems meet user goals. However, these techniques are not designed to foster creativity, meaning that resulting systems may be functionally useful but not sufficiently innovative. Further work has focused on applying creativity techniques for RE through workshops. However, the free-form representation of creative workshop outputs (text and informal diagrams), although flexible, is not grounded in user goals, or able to take advantage of goal model analysis, e.g., trade-off analysis. Furthermore, successfully conducting a creative RE workshop requires much experience and soft-skills, as well as a significant economic commitment. In this work, we summarize initial progress aiming to combine goal modeling and creativity techniques for enhanced RE. We focus on methods and tools for introducing creative ideas to goal modeling, and grounding creative outputs in goal-oriented models. Our focus on tooling and methods help to alleviate the need for expert-lead, costly workshops. We outline and illustrate proposed methods.

Keywords: creativity, istar, goal modeling, method, tool support

1 Introduction

Existing work in Requirements Engineering (RE) has focused primarily on software utility, introducing systematic methods such as goal-oriented modeling and analysis to ensure that requirements meet user needs. Although these techniques have been well studied, little emphasis has been placed on goal-oriented creativity: making sure that goal models capture creative ideas and creative design alternatives.

The past decade has seen the application of creativity techniques to software Requirements Engineering (RE), typically in the form of multi-day workshops (e.g., [7,8]). These workshops gather domain experts and, with the help of experienced facilitators, apply a number of creative activities (e.g., Round Robin, Creativity Triggers, Assumption Busting) in order to elicit creative ideas concerning new software.

Although these workshops have been successful in generating creative requirements, feeding into the design and construction of innovative systems, challenges exist. Workshop output is captured in a free and open format (text, use cases). Although this freedom enables capture of creative output, it makes it difficult to transfer these outputs to a format with is more precise and unambiguous, more amenable for downstream development and for transformation into design and code. The free-form nature of the creative output makes it difficult to perform any sort of systematic analysis of alternative ideas, with rationale for the rejection or acceptance of ideas often lost. Furthermore, creativity workshops are costly and require much guidance by experienced facilitators.

In this paper we outline ideas and progress for a multi-year project exploiting synergies between creativity techniques and goal modeling for RE. We aim to effectively use conceptual models in creative RE activities as part of a creativity methodology guided by tool-support. As such, we reduce reliance on creativity experts and expensive workshops, capturing creative ideas in a more structured form, taking advantage of existing RE modeling and analysis techniques, such as those offered by i* ([11,6]).

In addition to the creative RE workshops, individual methods have been introduced to support creativity in RE (e.g., [9,1]). Although these methods may be useful, they are somewhat fragmented and not joined together as part of one, model and tool-supported process. Our aim is to create a tool-supported framework which would allow for the integration of these and future techniques.

In the rest of the paper, we provide an overview of the proposed creativity method (Sec. 2), illustrate part of the method via an example (Sec. 3), then provide conclusions and future work (Sec. 4). Parts of the proposed method have been illustrated in previous short papers, with ([4]) focusing on an initial proposal for a tool-supported method and an exploratory experiment and ([5]) focusing on illustrating the combination of creativity and goal modeling with an air traffic control example. In this paper, we illustrate different parts of the proposed method with a further example.

2 Method Overview

Existing work (e.g., [10]) has identified many techniques which foster creativity. Creativity can be transformational, changing boundary rules to consider transformative ideas, possibly in another paradigm [2], exploratory, exploring a space of possibilities, or combinatorial, combining together creative output. Creativity techniques can be classified along these dimensions (see BeCreative for example classifications `becreative.city.ac.uk`).

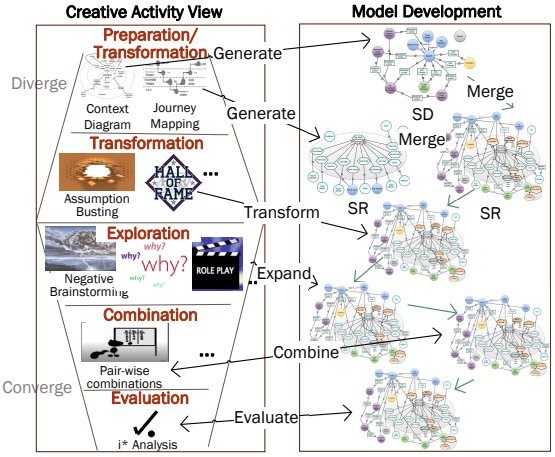


Fig. 1: Overview of Creativity and Goal Modeling Method and Tooling

We propose a method which guides users through a series of creativity activities, with goal models as the output and/or input of such activities. The activity order roughly follows the ordering of activities in previous RE creativity workshops, working through preparatory activities, then activities supporting transformational, exploratory, and combinatorial creativity. See Fig. 1 for an overview. On the left hand side we show the creativity activity view. Here, icons are used to represent various creativity activities, such as assumption busting and role play. Activities

will link to either guided instructions, or to external or integrated creativity tools such as Bright-Sparks (brightsparks.city.ac.uk). We use models such as context diagrams and journey maps to complement our goal-oriented creative process. We believe that the simplicity and different foci of these models, as compared to i*, leads to different types of creative thinking. Starting with transformations from these simpler models can help users to overcome difficulties in creating initial i* models.

On the right, we show the envisioned evolution of a resulting goal model after progressing through activities. In this paper, we illustrate some of these steps, showing the development of an SD then SR model. Further examples have been presented in [4,5]. In the first stages of development, modeling will be done using external RE tools such as OpenOME or online modeling tools such as draw.io; however, we aim to ultimately include modeling support within the tool.

After an iterative process of creative activities and modeling, the resulting goal model can be processed automatically to derive candidate textual requirements or structured Requirement Specifications, feeding into downstream development.

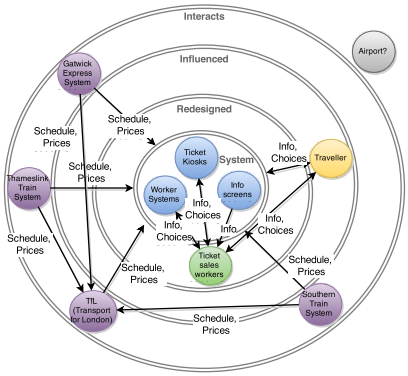
3 Illustrative Example: London Airport Trains

We illustrate parts of our method using a running example of train transport from London airports (inspired by transport from Gatwick Airport), specifically, the purchase of tickets, which offers the possibility of many different train services at different prices and routes, and can be confusing to visitors. Our recent findings as part of an exploratory experiment have shown that new users have trouble beginning to draw an i* model from scratch [3]. As such, in this paper we focus on the initial stages of the creative modeling methodology, which guides users through the creation of a starting, incomplete i* model via the creation of more simpler, intuitive, creativity-inducing models such as context diagrams and customer journey maps.

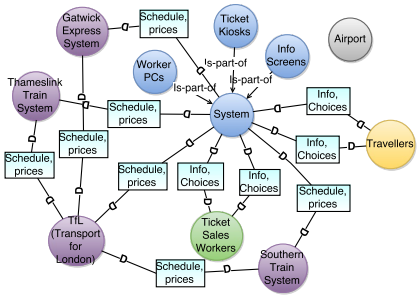
Context to SD. The process starts by urging users to draw a context diagram for the system. In this diagram, actors are drawn in a series of concentric circles, as actors move farther away from the center, they are less impacted by the new system. We show an example Fig. 2a. At the center are the core system actors, then the ticket sales workers, then the travelers, the system for each train and tube company, then, on the outer layer lies the airport, as the relation to the system is not currently well-understood. Actors are connected by arrows showing flows of information.

The creation of a context diagram is not necessarily creative – the model could capture the as-is situation without changes. However, the simple structure of this model well supports transformational creativity when the positions of the actors are challenged, potentially making transformative changes. For example, what is the role of the Airport, could it be moved closer to the core system, could there be a flow of information? Perhaps travelers could be provided information as they depart their flight? Perhaps they could book transport tickets with their flight? For simplicity, we continue the example without applying these ideas.

The information contained in a context diagram can form the beginnings of an i* SD diagram. Context actors are SD actors, while information flows are potential dependencies. We show the result of a proposed automatic conversion to an initial SD model in Fig. 2b. User are encouraged to rearrange, modify or add to this auto-generated figure.



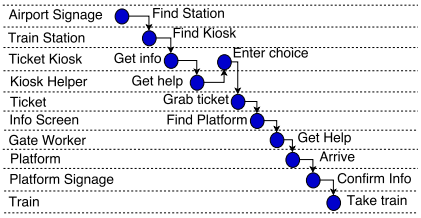
(a) Context Diagram for Airport Train System



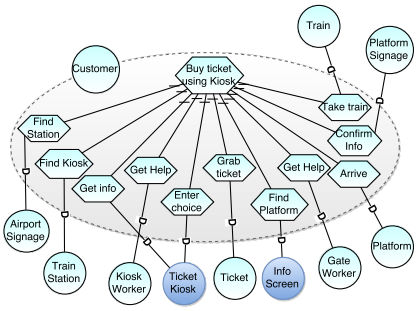
(b) Initial SD Diagram for Airport Train System - Transformed from Context Diagram

Fig. 2: Transformation from Context to SD

Journey Map to SR. A customer journey map is a simple way for users to explore one path through the system, capturing system actors or interaction points, and "touchpoints". We can explore the current path of purchasing a ticket from an airport train kiosk, illustrated in Fig. 3a. As with context diagrams, as-is journey maps are not necessarily creative. However, their structure forces users to explore and question the boundaries of a system, thus they are often used as part of exploratory and transformative creativity. In this case, when drawing the diagram we discovered that the first step in buying a ticket was finding the train station from within the airport, relying on the airport to provide appropriate direction and signage. In this case, we have expanded the boundary of our system as compared to the context diagram, when we did not know exactly what role the airport would play in our journey. Similarly, the ticket buying process ends when the travelers board the train. How can the train be included as part of our system design? Again, for simplicity, we continue the example without deeply exploring these creative prompts.



(a) Customer Journey Map for Buying a Ticket Using Kiosk



(b) Initial SR Diagram for Airport Train System - Transformed from Journey Map

Fig. 3: Transformation from Customer Journey Map to SR (colors match Fig.2)

As with context diagrams, we can make a simple transformation from journey maps to i*, specifically SR diagrams. In this case the columns are potential actors, while

touchpoints are potential tasks. As we can see in the transformed SR model in Fig. 3b some of the transformations are imperfect, for example, a ticket is likely a resource, not an actor. In this case we present the SR model to the user as a starting point and ask for their input in changing and rearranging the elements, again with the theory that it is easier to update and fix an i* diagram than to build one from scratch.

We can see that there are differences in the SD diagram produced via the context diagram and the SR diagram produced via the journey map, for example, Customer vs. Traveler. We can offer users a partial merge of the two diagrams, producing a new SR. Here, elements and actors with matching names are merged, while those with similar names can be clustered. We leave it to the user to finish the merge using their judgment. The end result will be a partial SR model which is roughly consistent with the other diagrams. We show the results of such a human-aided merge on the left of Fig. 4.

Exploratory Creativity applied to SR. We can make a simple mapping between several exploratory creativity techniques and the SR diagram, with the purpose of helping users to fill out the SR detail. For example, one of the existing creativity techniques is called "Why why why?", urging users to constantly question the motivation for each element of a system design. This, obviously, can help the user to move up the i* model, adding higher level intentions until the why question is no longer sensible. Similarly, we can consider "how how how?" and "who who who?", helping to elicit operationalizations and dependencies. Another useful technique, "negative brainstorming", urges user to think of what can go wrong, helping to elicit negative contribution links and other related elements. Conversely, we could apply "positive brainstorming" to help find positive contributions. We show some results of these activities in Fig. 4 in orange/bold.

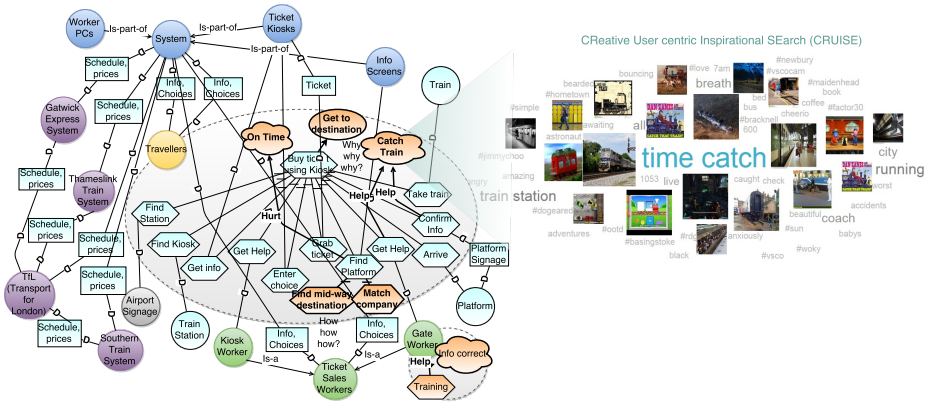


Fig. 4: SR Diagram Resulting from Context and Journey Map with Suggestions for Catch Train from CRUISE (Orange/Bold show Additional Creative Results)

Using i* Structure for Creativity. Thus far we have focused on techniques which help users to build and fill in the details of an i* model. Other techniques can take the i* model as an explicit input. Element names, such as Catch Train, could be used as input for creative search tools such as cruise.imuresearch.eu (shown in Fig. 4), with generated ideas (e.g., warnings for full trains, identifying which trains have snack cars) incorporated back into the goal model. Combinatorial creativity techniques such as pair-

wise comparison, can use the structure of the model to automatically compute different combinations of actors, suggesting potential new dependencies. Techniques, such as the one described in [1], could take as input i* element labels and suggest unfamiliar combinations of elements. Finally, i* analysis, such as in [6] can be used to evaluate and select the outputs of creative activities as captured in the model. We intend for our method and tooling to support such activities via incremental development, with successive releases including support for more and more creativity activities.

4 Conclusions and Future Work

We have outlined a tool-supported methodology for combining creativity and i* modeling. Our proposal incorporates creative ideas into the goal-oriented RE process and grounds creative ideas in user goals, allowing for systematic analysis and (partial) transformation to requirements specifications.

We are currently working on the design and flow of the proposed tooling. We intend to pilot the method internally, then to test successive versions with willing industry contacts, eventually leading to larger-scale, in-situ deployments and case studies.

Acknowledgments. This research is supported by an ERC Marie Skodowska-Curie Intra European Fellow-ship (PIEF-GA-2013-627489) and a Natural Sciences and Engineering Research Council of Canada Postdoctoral Fellowship (Sept. 2014 - Aug. 2016).

References

1. T. Bhowmik, Nan Niu, A. Mahmoud, and J. Savolainen. Automated support for combinational creativity in requirements engineering. In *IEEE 22nd International Requirements Engineering Conference (RE14)*, pages 243–252, 2014.
2. M. A. Boden. *The Creative Mind*. London: Abacus, 1990.
3. J. Horkoff. Observational studies of new i* users: Challenges and recommendations. In *First iStar Teaching Workshop (iStarT15)*, 2015.
4. J. Horkoff and N. Maiden. Creativity and conceptual modeling for requirements engineering. In *CreaRE: Fifth International Workshop on Creativity in Requirements Engineering*, 2015.
5. J. Horkoff, N. Maiden, and J. Lockerbie. Creativity and goal modeling for software requirements engineering. In *Creativity & Cognition (poster)*, 2015.
6. J. Horkoff and E. Yu. Interactive goal model analysis for early requirements engineering. *Requirements Engineering*, pages 1–33, 2014.
7. N. Maiden, S. Jones, K. Karlsen, R. Neill, K. Zachos, and A. Milne. Requirements engineering as creative problem solving: A research agenda for idea finding. In *18th IEEE International Requirements Engineering Conference (RE10)*, pages 57–66, 2010.
8. N. Maiden, C. Ncube, and S. Robertson. Can requirements be creative? experiences with an enhanced air space management system. In *Software Engineering, 2007. ICSE 2007. 29th International Conference on*, pages 632–641, May 2007.
9. L. Mich, C. Anesi, and D. M. Berry. Requirements engineering and creativity: An innovative approach based on a model of the pragmatics of communication. In *Requirements Engineering: Foundation of Software Quality (REFSQ04)*, 2004.
10. M. Michalko. *Thinkertoys: A Handbook of Creative-Thinking Techniques*. Potter/TenSpeed/Harmony, 2010.
11. E. Yu. Towards modelling and reasoning support for early-phase requirements engineering. In *3rd IEEE International Symp. on Requirements Engineering*, pages 226–235. IEEE, 1997.

*i*Star in Practice: On the identification of reusable SD Context Models Elements

Karina Abad, Juan Pablo Carvallo, Catalina Peña

Computer Science Department
University of Cuenca, Ecuador

{pablo.carvallo;karina.abadr;catalina.pena@ucuenca.edu.ec

Abstract. Modern enterprises rely on Information Systems (IS) required both to support their operation and provide information required to endorse strategic decisions. Because of their increasing complexity, such systems are usually constructed by integrating software components of different nature and origins into hybrid systems, for which architectural design plays a fundamental role. However, far from simple, this task is usually cumbersome. In previous work we have addressed this issue and proposed a four steps, pattern-based approach, aimed to help in the solution of this problem. In first steps, patterns are described as Context Models, which include recurring elements (actors and dependencies) identified in several industrial cases. In this work we further address this issue and present an study aimed at the validation and extension of such patterns, and/or the identification of new ones, by reviewing recurring elements appearing in 29 semi-industrial IS architectural design processes.

1 Introduction

Modern enterprises rely on *Information Systems* (IS) specifically designed to manage the increasing interactions with their context. *Enterprise Architecture* (EA) [1], is a new approach involving several levels of architectural design, including IS architecture, which requires deep understanding of enterprise context and strategies. *Enterprise Context Models* (CM) are usually built to support this process, assisting enterprise decision-makers to design and refine their business strategies and enterprise architects to understand what will be required from IS. Far from easy, the construction of such models is usually a cumbersome task, mainly due to communication gaps among technical personnel with limited knowledge of enterprise structure, operations and strategy, and their administrative counterparts imposing pressure and time constraints to the process.

In order to deal with these problems, in the last few years we have intensively used the *i** notation to bridge the gap among technical consultants and non-technical stakeholders [2] and proposed the DHARMA method [3], for discovering IS architecture departing from the construction of CM expressed in *i**. The application of the first activities of this method in several industrial and academic cases, allowed us to identify a catalogue of patterns [4], which could be used as templates for both technical and

managerial personnel in order to improve their understanding. Patterns store knowledge represented by *i** Strategic Dependency models, including generic environmental actors and their strategic dependencies. The catalogue distinguish two levels of abstraction, the higher applicable in general to any kind of enterprise and the lower which considers enterprise strategies describing how a particular enterprise operates.

Although very valuable in practice, we thought that the catalogue could be extended, with additional levels representing knowledge of more specific enterprise domains. In this paper we present initial findings in relation to this belief, which emerged after conducting several semi-industrial cases of applications of the DHARMA method.

2 The Case Studies

In the last three years we have conducted 29 semi-industrial cases of application of the DHARMA method (industrial cases conducted by senior Information Systems Engineering students with support of teachers, for which formal agreements existed, but were conducted with no cost for participant enterprises). Cases were part of a broader study conducted in Ecuadorian enterprises, intended to identify CMs patterns meant to improve the identification of IS architectures (*System Actors* -atomic software domains that structure the system-, services that must be covered by them and their relationships). CMs constructed for these processes were used to validate and extend the patterns presented in [4] (by measuring occurrence of the included elements), and to identify new domain specific ones.

In the study, 25 of the enterprises were small companies, 3 medium size, and the last one a large manufacturing company. This distribution aligns with the Ecuadorian reality, mainly structured with small companies (97,94%) [6]. Enterprises were categorized according to *NACE Rev 2*. Categories included: *Manufacturing* (wood, textiles, food and cardboard processing); *Wholesale and retail trade* (hardware and software, textiles, leather, home appliances, motorized vehicles and general goods); and *Services* (basic, specialized –language- and advanced education, and financial – accounting-)

3 Data Analysis

Actors and dependencies included in the resulting 29 CMs were extracted and placed in tables specifically designed to support the analysis process. Columns represent modelled enterprises whilst rows list the identified actors (table 2) and their corresponding dependencies (table 3). Actors identified in the 29 cases were grouped in relation to 8 of the generic actors identified in [4], *Suppliers*, *Consumers*, *Strategic Partners*, *Distributors*, *Financial Institutions*, *Regulatory Agencies*, *Control Agencies*, *Competitors*. Table cells are used to state the cases in which listed actors/dependency were identified. Total column adds up the number of occurrences of elements in each row, whilst percentage gives the relation among the totals and the number of case studies.

At the end, a total of 54 actors and 189 dependencies were identified in the 29 cases. All of the actors are instances of the generic actors identified in [4], which makes evident the validity of knowledge included in the proposed patterns in relation the this kind of elements. 23 out of 54 actors identified appear in at least 17% of the cases; 14 of them in at least 24% of the cases.

Table 1. The case studies

	Enterprise	Size: Small	Size: Medium	Size: Large	Industry	Dependencies: Actors	Goals	Soft	Resour	Tasks	Total
1	Panadería Centenario	X			C - 10.71 Manufacture of bread; manufacture of fresh pastrygoods and cakes	11	8	10	6	0	24
2	Sport Chavis	X			C - 14. 13 Manufacture of other outerwear	12	10	11	13	1	35
3	FABRICA	X			C - 16.29 Manufacture of other products of wood; manufacture of articles of cork, straw S - 96.03 Funeral and related activities 9603	11	6	12	7	0	25
4	CARTOPEL			X	C - 17.1 Manufacture of pulp, paper and paperboard	16	10	14	0	0	24
5	Forjart	X			C - 24 Manufacture of basic metals	13	7	10	1	3	21
6	ElectroUnion	X			C - 27.5 Manufacture of domestic appliances	20	11	13	12	0	36
7	Muebleria BienStar	X			C - 31.0 Manufacture of furniture	13	8	7	1	1	17
8	FEMUSA Mobiliarios	X				12	4	4	5	1	14
9	SANTANA Muebles	X				9	11	15	6	2	34
10	Importadora Tomebamba		X		G - 45.1 Sale of motor vehicles, G - 46.43 Wholesale of electrical household appliances	14	6	11	8	0	25
11	JCEV Cia. Ltda.		X		G - 45.1 Sale of motor vehicles, G - 46.43 Wholesale of electrical household appliances	10	10	8	13	1	32
12	TECNISUR	X			G - 45.2 Maintenance and repair of motor vehicles	8	7	7	5	3	22
13	Trebol Roses	X			G - 46.22 Wholesale of flowers and plants	15	11	12	8	2	33
14	CAPEDI	X			G - 47.1 Retail sale in non-specialised stores	14	9	13	8	0	30
15	All Design	X			G - 47.41 Retail sale of computers, peripheral units and software in specialised stores	8	8	10	6	2	26
16	Giga Computers	X				17	14	11	8	2	35
17	APC Tecnologia	X				4	5	10	9	0	24
18	HOLIDATSERV	X				8	7	8	7	1	23
19	TOTAL COMPU	X				9	12	10	9	1	32
20	Dress Up Store	X			G - 47.51 Retail sale of textiles in specialised stores 4751	9	10	9	6	3	28
21	KRISTEN	X				12	10	15	12	1	38
22	Sodilibro	X			G - 47.61 Retail sale of books in specialised stores	11	8	5	4	1	18
23	enlinea.com	X			G - 47.7 Retail sale of other goods in specialised stores	10	6	8	5	1	20
24	Calzado Turismo	X			G - 47.72 Retail sale of footwear and leather goods in specialised stores	6	3	3	6	0	12
25	ByB Asesoría contable y tributaria	X			K - 64.99 Other financial service activities, except insurance and pension funding	11	9	13	6	2	30
26	Jardín ABC	X			P - 85.1 Pre-primary education	12	6	3	9	4	22
27	Colegio Técnico Sudamericano		X		P - 85.3 Secondary education	23	6	7	9	1	23
28	CORNATEC Cía. Ltda.	X			P - 85.59 Other education	15	15	12	11	1	39
29	Golden Bridge	X				23	14	15	7	0	36

Table 2. Excerpt of identified actors and their occurrence in the 29 cases conducted.

Generic actor	Actor	Sport Chavis	Muebleria BienStar	Forjart	ElectroUnion	Dress Up Store	All Design	Sodilibro	Jardín ABC	Giga Computers	Calzado Turismo	CAPEDI	APC Tecnologia	FEMUSA Mobiliarios	KRISTEN	Importadora Tomebamba	JCEV Cia. Ltda.	TECNISUR	FABRICA	enlinea.com	HOLIDATSERV	Colegio Técnico	ByB Asesoría contable	Total Compu	Trebol Roses	Panadería Centenario	Comatec Cia. Ltda.	CARTOPEL	Golden Bridge	Santana muebles	Total	Percentaje
Suppliers	Supplier	X		X				X		X		X	X	X	X	X	X	X	X		X		X	X			X	X		X	19	66%
	Raw material supplier	X	X	X								X		X	X				X						X	X		X		X	11	38%
	Parts supplier															X															1	3%
	Finished goods supplier															X				X									X		3	10%
	Supplies	X	X																				X					X	X		5	17%
	Telecomunications supplier		X				X													X			X						X		5	17%
	Technology supplier	X	X								X									X									X		5	17%
	Basic services supplier											X	X		X				X				X					X	X	X	8	28%
	Transport supplier									X		X	X			X	X								X						6	21%
	Insurance and patent supplier									X						X						X		X			X				5	17%
	General Services supplier	X							X	X							X			X		X			X			X	X		9	31%
	Wholesale supplier										X																				1	3%
	Retail supplier										X																				1	3%
	Local supplier	X			X	X				X				X							X										6	21%
	National supplier				X	X		X		X					X	X					X										7	24%
	International supplier							X		X					X						X										4	14%

These statistics point to that fact that they can be used as check list to support the identification of actors in future cases. However we think that a more interesting finding is the fact that actors grouped into generic actors define orthogonal dimensions that can be used to categorize them (see table 4 for an excerpt). For instance, Actors categorized under the *Suppliers* generic actor define at least three dimensions: *Location* (local, national, International); *Kind of supply* (products –raw materials, supplies or technology-, or services); and *Volume* (wholesale or retail). The importance of this finding will be illustrated in section 4.

It is important to notice that CM in most of the cases also included generic actors, (even when more specific instances have been identified) e.g. generic actor *Suppliers*

and the instances *Row Materials*, *Technology*, *Basic Services* etc., included in Table 2. This fact supports the need of the "is-a" generalization-specialization construct included in *i**, as a mean to support the grouping of dependencies shared by instances of a more generic actor. These dependencies representing intentional aspects common to all of them in relation particular organizational processes.

Similarly to actors, some dependencies are instances of more generic ones, included in patterns presented in [4], but also some additional ones were identified. 52 out of the 189 dependencies appeared in at least 17% of the cases; 36 of them in at least 24% of cases. Dependencies are related to specific actors and stored together with them in the patterns catalogue. Therefore, they can also be used as check lists to identify dependencies to be included in CM of future cases, e.g. by using the instantiation rules proposed in [4].

Table 3. Excerpt of generic dependencies found in the 29 cases for the actor supplier.

Dependency	Direction	Type	Sport Chavis	Muebleria BienStar	FORJAR I	Electro Union	Dress up store	ALL DeSIGN	SOLIDBIKO	Jardin ABC	GilSA COMPUTERS	Calzado Turismo	CAPEDI	APC TECNOLOGÍA	FEMUSA	Kristen	Importadora Tomebamba	ICCNSUR	TECNISUR	FABRICA	editrex.com	HOLDAT SERV	Sudamericano	ByE Asesoria	TOTAL COMPU	Trebol Roses	Panaderia Centenario	CORNATEC	CARTOPEL	Golden Bridge	SANTANA Muebles	Total	Percentaje	Actor
Technology, products or services acquired	-->	Goal	X	X	X	X	X	X			X	X	X	X	X	X	X	X	X	X	X		X		X	X	X		X	X	X	24	69%	Supplier / Service supplier
Technology, products or services	-->	Resource	X			X	X	X			X		X	X		X	X	X	X	X	X	X			X	X	X	X	X		X	20	38%	Supplier
Payment made	<--	Goal														X	X		X						X	X	X		X			8	83%	Supplier
Quality of products and services	-->	Soft Goal	X		X	X	X				X	X	X	X		X	X	X	X	X	X	X			X	X			X	X	X	20	21%	Supplier
Timely delivery	-->	Soft Goal	X		X	X	X	X	X		X		X			X			X		X		X		X				X	X	X	16	3%	Supplier Local supplier Transport supplier
Timely billing	-->	Soft Goal														X		X	X													3	34%	Basic services supplier
Timely payments	<--	Soft Goal						X					X	X		X	X	X	X				X	X	X			X			X	12	55%	Supplier
Payment facilities/credits	-->	Soft Goal					X		X		X		X			X	X			X		X	X	X	X							11	17%	National supplier
Llow prices	-->	Soft Goal	X				X				X					X	X	X				X	X	X						X		10	14%	Supplier
Discounts	-->	Soft Goal	X											X											X					X	X	5	17%	Supplier
Catalog	-->	Resource	X				X					X				X		X				X		X				X	X			9	34%	Supplier
Product/Service invoiced	-->	Goal		X									X	X												X	X				X		6	28%	Supplier

Table 4. Dimension found for Customers generic actor.

Generic actor	Dimension	Actor Instances	Associated dependencies	Type	Direction
Customers	Frecuency or Volume	Potencial	Widespread promotions	Goal	-->
			Promocional samples	Resource	<--
		New	Membership card provided	Goal	-->
			Special introduction prices provided	Soft goal	-->
			Membership card	Resource	-->
			Personal information registered	Goal	<--
		Important	VIP benefits granted	Goal	-->
			Personalized attention	Soft goal	-->
			VIP card	Resource	-->
			Important high volume order placed	Goal	<--
	Distribution channel	Wholesaler	Product availability guaranteed	Goal	<--
			Product distribution agreement signed	Soft goal	<--
			Increase sales through the distribution chain	Soft goal	<--
			Product distribution agreement	Resource	<--
			Product distribution chain achieved	Soft goal	-->
		Retailer	Restocking in small quantities provided	Goal	-->
			Approach consumers through an specific location	Soft goal	<--
			Increase sales through individual stores	Soft goal	<--
		Specific market Segment	Specialized customer service infrastructure	Soft goal	-->
			Trained stuff for specific needs	Soft goal	-->
			Specific documents	Resource	-->
	Payment method	Credit	Deferred payments	Goal	-->
			Credit flexibility	Soft goal	-->
			Acceptance of various credit cards	Soft goal	-->
			Voucher	Resource	-->
			Warranty documents	Resource	<--
		Cash	Cash rebates	Goal	-->
			Money	Resource	<--
			Technology, products or services provided	Goal	<--
			Timely payments	Soft goal	<--
			Products, services, technology	Resource	<--
			Invoiced purchases	Goal	-->
			Quality of products or services	Soft goal	-->
			Bill	Resource	-->

Because of problems with *i** semantics, and the descriptions used by modelers in different cases, mapping of similar dependencies is not as straightforward as mapping actors. For instance, for the generic actor *Supplier* we found the objective "*Payment Made*" in 8 out of 29 cases. However, when later analyzed, it became evident that systems engineers were using other types of dependencies to state the same intentional aspect, in order to emphasize aspects that were relevant for their administrative counterparts, e.g. the soft goal "*timely payment*" or the resources "*payment documents*" or "*cash/check*". In addition to semantics, variations can be attributed to lack of experience of engineers, the existence of "unfamiliar" industrial glossaries or the fact that some dependencies were omitted as redundant.

4 Reusing Knowledge Elements

At this point, we have shown important evidence supporting reusability of the proposed patterns and their elements. Because of this, we can sustain that a good way to construct *i** SD-based CM, instead of departing from scratch, is to reuse the elements included in the proposed patterns, going through them as a checklist and adopting those that are relevant for the enterprise context being modeled. Furthermore, in [4] we have defined several pattern instantiation rules specifically designed to support this process.

However, in this paper we argue that there can be and alternative and more systematic way to reuse CM elements (actors and dependencies), to construct complete *i** SD-based CM from scratch and eventually automate this process. An important aspect emerging from this work, introduced in section 2, is the identification of several orthogonal dimensions useful to classify instances of generic the actors (see table 4 for an excerpt in relation to the Customer generic actor). Each of these dimensions has a set of associated value labels, representing potential actor instances (identified from CM of the 29 case studies). These labels have sets of generic dependencies (also identified from the 29 case studies) associated to them. Based on this table, practitioners (system engineers and administrative staff) can systematically identify a large number of actors on their operational context, by selecting and combining labels from each dimension. To illustrate the approach, let's consider the first two labels of three of the *Customer's* categorization dimensions in table 4, *frequency/volume*, *distribution channel,* and *payment method*. In this case, 12 combinations representing potential instances of actors in the context of the organization are possible: *Potential Wholesaler Credit, Potential Wholesaler Cash, New Wholesaler Credit, New Wholesaler Cash, Important Wholesaler Credit, Important Wholesaler Cash, Potential Retailer Credit, Potential Retailer Cash, New Retailer Credit, New Retailer Cash, Important Retailer Credit, and Important Retailer Cash*.

Let's assume that in a particular case the *New Wholesaler Credit Customer* is selected from this set of combinations, then all the dependencies associated to labels included in the name are potential dependencies to be included in the CM of the organization, see figure 2. In this way, identification of dependencies can also be automated. Multi-inheritance shall be used in order to avoid duplication of dependencies in cases were several instances of a same generic actor include occurrences of the same labels on their names. Also dependencies associated to the generic actor have to be included in the model for the reasons explained in section 3.

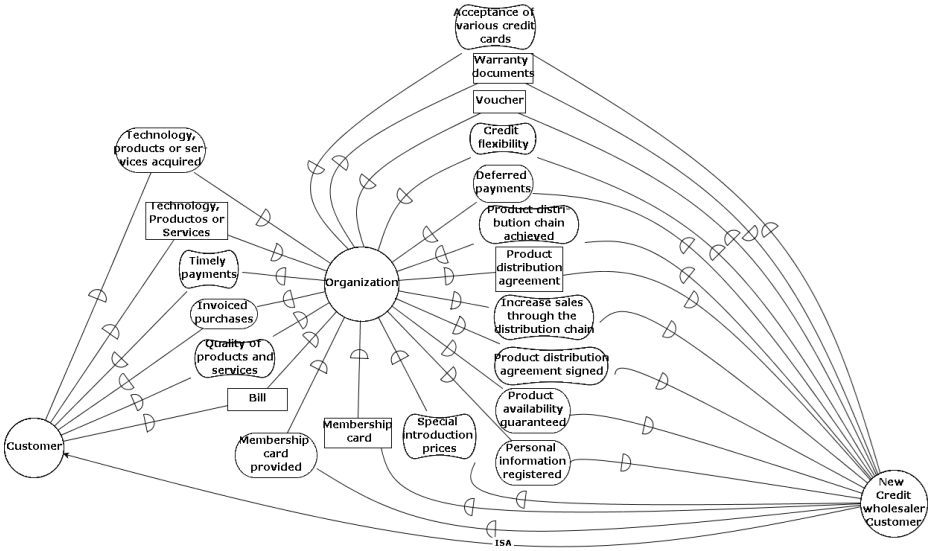


Fig. 1. Final generic i* model

5 Conclusions

In this paper we have presented an approach to automate construction of *i** SD based-CM, which reuses elements (actor and dependencies), included in the patterns presented in [4]. Elements in these patterns have been validated, and patterns have been extended with the results of 29 semi-industrial IS architectural design process, conducted in the last three years. All of these projects used the DHARMA method, which requires enterprise CM to be constructed as departing activity for a IS architectural design.
We have also proposed a method to systematize the identification of context actors and dependencies, and eventually automate the construction of *i**-based CM. it is important to remark that the proposal is based in a significant amount of empirical evidence which makes it highly useful. We are currently finishing the construction of a tool to support the method and exploring the ontological representation of patterns in order to improve CM construction, by automatically recommending the elements to be included in them.

References

1. The Open Group. *The Open Group Architecture Framework (TOGAF) version 9.* The Open Group, 2009
2. Carvallo, J.P. *Supporting Organizational Induction and Goals Alignment for COTS Components Selection by Means of i**. ICCBSS 2006
3. Carvallo, J.P., Franch, X. *On the Use of i* for Architecting Hybrid Systems: A Method and an Evaluation Report*. PoEM 2009.
4. Carvallo, J. P., & Franch, X. *Building Strategic Enterprise Context Models with i*: A Pattern-Based Approach*. TEAR 2012.
5. Steinberg, M. *Enterprise Applications: A Conceptual Look at ERP, CRM, and SCM*. Hill Associates Inc., 2006.
6. http://aplicaciones2.ecuadorencifras.gob.ec/dashboard2/pagina3.php

A Holistic Approach to Security Attack Modeling and Analysis *

Tong Li[1], Jennifer Horkoff[2], Kristian Beckers[3], Elda Paja[1], and John Mylopoulos[1]

[1] University of Trento, Trento, Italy
{tong.li,paja,jm}@unitn.it
[2] City University London, London, UK
horkoff@city.ac.uk
[3] Technische Universität München, München, Germany
beckersk@in.tum.de

Abstract. Protecting socio-technical systems is a challenging task, as a single vulnerability or exposure of any component of the systems can lead to serious security breaches. This problem is exacerbated by the fact that the system development community has not kept up with advances in attack tactics. In this paper, we present ongoing research on the development of a holistic attack analysis technique. Our approach adopts a goal modeling technique to capture attacker malicious intention as anti-goals, which are systematically refined and operationalized into concrete attack actions which target various assets (e.g., human, software, and hardware). A comprehensive attack pattern repository (CAPEC) is seamlessly integrated into our approach in order to provide analysts with practical security knowledge and assist them in identifying potential attacks under specific contexts. Finally, a set of security controls is provided for mitigating identified attacks.

1 Introduction

Socio-Technical Systems (STSs) consist of human, software and physical elements that together fulfill system requirements. Due to their heterogeneity and complexity, such systems are exposed to a broader range of attacks than their software cousins. Attackers are able to breach system security by targeting any vulnerable component of STSs, such as human, software applications, or physical infrastructure. Consider a smart meter system as an example [1]. An attacker can access energy consumption data by performing social engineering against the stakeholders, by intercepting communication data transmitted between software applications, or even by probing the physical smart meter device. The larger attack surfaces of STSs can also lead to multistage attacks that combine attacks on different parts of an STS [2].

Thinking like an attacker has been proposed as an effective solution to discover attacks that are most likely to be performed by an attacker [3]. As such, security analysis can consider alternative countermeasures to mitigate identified attacks and ensure the satisfaction of security requirements. Many approaches have been proposed for analyzing security requirements from an attacker's perspective, such as anti-goal analysis [4] and misuse cases [5]. However, these approaches are not designed for STSs but for software, i.e., do not explicitly capture inter-dependencies between software and other system components (e.g., business processes, hardware). As a result, multistage attacks that target several system components cannot be appropriately captured.

Another obstacle to STS security is that attack analysis lacks knowledge of impending attacks. Barnum and Sethi have pointed out that the software engineering community has not kept up with advances in attack knowledge, resulting in less effective, and sometimes useless security designs [3]. Attack patterns, as solutions to this problem, document reusable attack knowledge in support of system security solutions. Specifically, CAPEC (Common Attack Pattern Enumeration and Classification) is a comprehensive attack knowledge repository, which includes 463 attack patterns[1]. However, without an efficient method to use this large set of patterns, analysts are reluctant to adopt them in practice [6].

We have proposed a holistic approach for modeling and analyzing attacks for STSs in a companion poster [7]. In this paper, we describe recent progress regarding this work. In particular, we present and illustrate a refined analysis process. We base our approach on a three-layer requirements framework [8] in order to consider threats from various system viewpoints and provide a holistic security analysis. Specifically, our approach takes an attacker's viewpoint to generate attack strategies by systematically capturing and refining attacker malicious intentions. Moreover, we seamlessly integrate CAPEC attack patterns into our approach to effectively identify operational attacks, based on which corresponding security controls are applied. Finally, a supporting tool is under development, and we also describe how the tool can support the proposed analysis process.

2 Background

Three-layer requirements modeling framework. Li et al. [8] proposed a three-layer requirements framework, which models and analyzes requirements of STSs at the business layer, software application layer, and physical infrastructure layer, respectively. In our proposal, we take the three-layer requirements model as input, which allows us to capture threats that originate in different layers of a system, and analyze attacks from a holistic viewpoint.

Contextual goal modeling. Ali et al. [9] have extended Tropos with context-related concepts in order to model and analyze stakeholder requirements in different contexts. In this paper, we propose to model attack patterns as contextual

[1] https://capec.mitre.org

goal models in order to (semi-)automate the selection of attack patterns during anti-goal analysis and operationalization.

Attack patterns. Inspired by design patterns, attack patterns were first proposed by Moore et al. [10] in order to reuse proven attack knowledge. Notably, CAPEC has been under development for years and currently includes 463 attack patterns. Each attack pattern is specified in terms of *Attack Prerequisites*, *Attack Motivation-Consequences*, *Solutions and Mitigations* etc. However, it is difficult to use the CAPEC repository, as analysts have to manually navigate and select appropriate patterns. In this paper, we model attack patterns as contextual goal models in order to semi-automate the corresponding analysis.

3 A Holistic Attack Analysis Approach

Our approach takes a three-layer system requirements model as input, which includes both functional requirements and security requirements, and eventually produces a list of security controls that can effectively protect the system from being damaged by attackers. An overview of the holistic attack analysis process is shown in Fig. 1. Each step of the analysis will be specified in the following subsections.

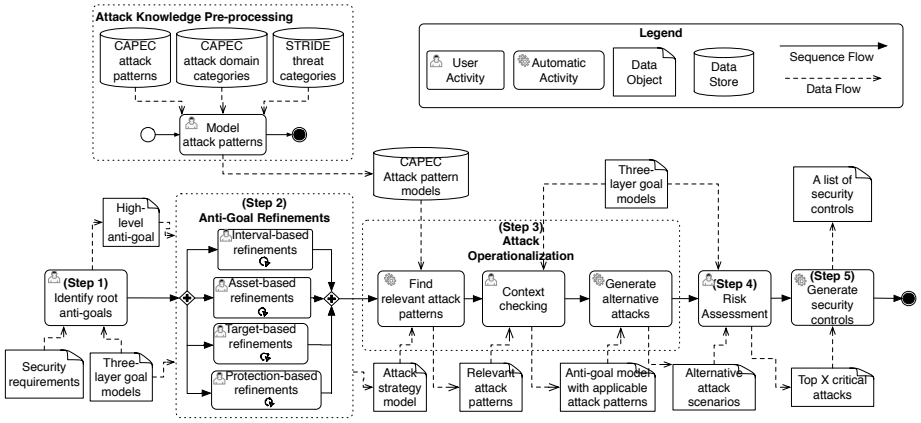


Fig. 1: An overview of the holistic analysis process

Step 1: Root anti-goal identification. We capture an attacker's high-level malicious intentions against a system as anti-goals, which are systematically analyzed to explore alternative attacks. In order to (semi-)automate anti-goal analysis, we propose to characterize attacker's anti-goals as a quadruple, consisting of four attributes: *Asset*, *Threat*, *Target*, and *Interval*.

- *Asset* is anything of value to stakeholders. Attackers can benefit from attacking assets.
- *Threat* indicates an undesired condition of an asset, which attackers try to achieve to fulfill their malicious desires. In this work, we leverage the STRIDE threat categories [6] to specify threats.

– *Target* is a component of a system, which involves assets and has vulnerabilities that are exploitable by attackers. Within the three-layer system structure, targets vary from layer to layer.
– *Interval* represents the time period, during which attackers carry out attacks. In this work, an interval is specified in terms of a system functional task, which indicates the execution period of the task. Note that a goal can also be specified as an interval, which means the execution period of all the operationalized tasks of this goal.

As shown in Fig. 2, a root anti-goal *AG1* is derived from the root security goal *SG1* that is captured in the three-layer goal model. In particular, *AG1* capture the malicious intention "Tampering (*Threat*) energy demand (*Asset*) when the real-time pricing is applied (*Interval*) by attacking the energy supplier (*Target*)", which negates *SG1*.

Step 2: Anti-goal refinement. When attacking a complex system, an attacker can have various attack strategies to achieve his root anti-goal. An attack strategy sheds light on which system components to attack and when to attack, but does not mention concrete techniques and attack actions. Once root anti-goals are identified, we propose to systematically refine them in order to explore various attack strategies across three layers.

To this end, we investigate several attack scenarios (reported in [2]) to understand how attackers generate attack strategies to achieve their malicious intention. Based on the investigation, we identify four refinement methods to simulate the generation of attack strategies, as presented in Fig. 1. Take the interval-based refinement pattern as an example. As shown in Fig. 2, the root anti-goal *AG1* applies during the interval *G1*, and the interval *G1* is "and-refined" into two sub-interval *G2* and *G3*. Thus, *AG1* is "or-refined" into *AG2* and *AG3*, which apply during intervals *interval(G2)*, *interval(G3)* respectively.

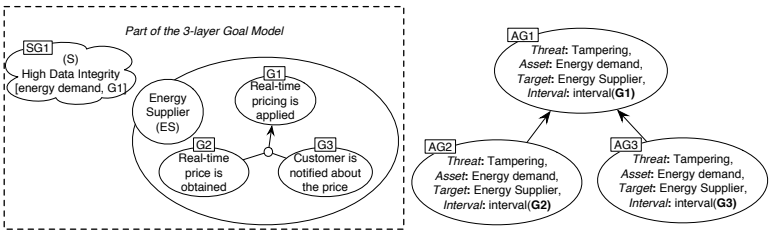


Fig. 2: An example of interval-based refinement

Step 3: Anti-goal operationalization. Anti-goal refinements address when and what to attack in order to achieve attacker malicious intentions. In this step, we leverage the attack knowledge from the CAPEC repository to analyze whether leaf anti-goals can be achieved by known attacks. In order to (semi-) automate the analysis, we construct a contextual goal model for each CAPEC attack pattern according to its textual description. An example is shown in Fig. 3, capturing the JSON Hijacking pattern.

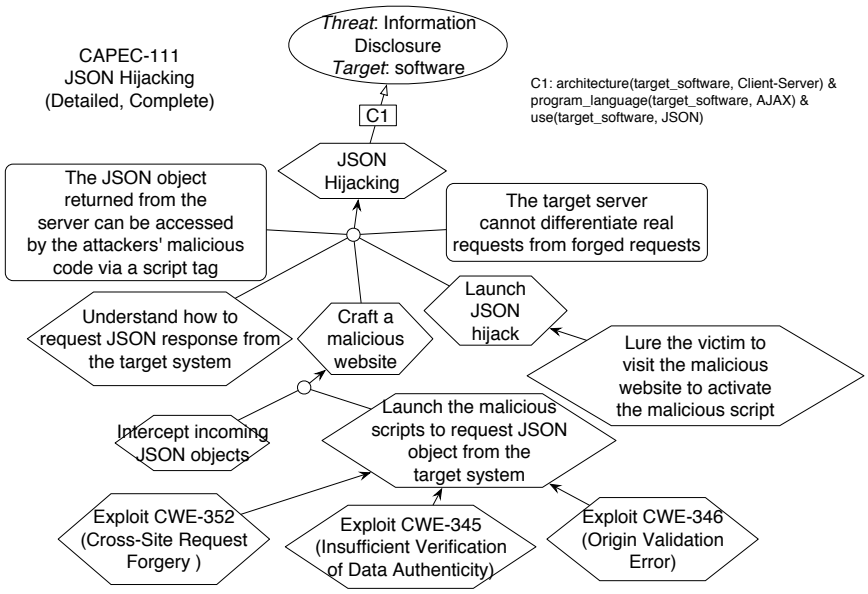


Fig. 3: The attack pattern model of JSON Hijacking (CAPEC-111)

Operationalization analysis consists of two steps: relevance analysis and applicability analysis. Given a leaf anti-goal, we first automatically identify all relevant attack patterns from the attack repository. In particular, if the leaf anti-goal and the anti-goals modeled in an attack pattern model concern the same threat and the same type of target (software, hardware etc.), then the that attack pattern is relevant to the leaf anti-goal. After identifying the relevant attack patterns, we further check their applicability, i.e., whether the contexts required by the attack patterns are held in the target system. Specifically, the context of each relevant pattern will be automatically checked against the three-layer requirements goal model. If the information captured in the goal model is not enough to determine whether the context applies, then the supporting tool will interactively ask analysts to check them. Once the applicable attack patterns are determined, we can automatically generate all alternative attacks according to the derived anti-goal model.

Step 3: Risk assessment. For alternative attacks identified in the previous step, we assess their risk by analyzing the vulnerabilities exploited by those attacks. To this end, we propose to use external vulnerability assessment services to detect vulnerabilities that exist in the target system [2]. Based on the result of vulnerability analysis, we can assess the risk of alternative attacks and further prioritize them.

Step 4: Generate mitigation controls. Given prioritized alternative attacks, an analyst can choose how many to tackle, depending on her budget. For each

[2]https://cve.mitre.org/compatible/product_type.html

attack to be addressed, our approach finally generates corresponding mitigation controls according to security knowledge documented in the CAPEC repository.

4 Conclusions and Future Work

We present ongoing research on a holistic attack analysis technique, which takes an attacker's viewpoint by capturing their malicious intents as anti-goals. The approach takes into account threats that originate from various system components, identifies alternative attacks that target the vulnerable components, and finally provides effective security controls to satisfy security requirements.

Apart from the analysis process we have investigated, we are working on the tool-supported implementation of each step. In particular, we seek to deeply integrate practical attack knowledge (e.g., CAPEC) into our approach in order to deal with real world security problems. To this end, we need to process a reasonable amount of the attack patterns (as presented in Section 3). Furthermore, we will improve the integration of external vulnerability assessment services into the risk assessment step of our analysis process. Once all the analysis steps are well designed and can be supported by our tool, we plan to perform a case study to validate our approach.

Acknowledgements This work was supported by the ERC advanced grant 267856, titled "Lucretius: Foundations for Software Evolution".

References

1. Flick, T., Morehouse, J.: Securing the smart grid: next generation power grid security. Elsevier (2010)
2. Mitnick, K.D., Simon, W.L.: The art of deception: Controlling the human element of security. John Wiley & Sons (2011)
3. Barnum, S., Sethi, A.: Attack patterns as a knowledge resource for building secure software. In: OMG Software Assurance Workshop: Cigital. (2007)
4. Lamsweerde, A.V.: Elaborating security requirements by construction of intentional anti-models. In: ICSE. (2004) 148–157
5. Sindre, G., Opdahl, A.L.: Eliciting security requirements with misuse cases. Requirements Engineering **10**(1) (2005) 34–44
6. Shostack, A.: Threat Modeling: Designing for Security. John Wiley & Sons (2014)
7. Li, T., Paja, E., Mylopoulos, J., Horkoff, J., Beckers, K.: Holistic security requirements analysis: An attacker?s perspective. In: Requirements Engineering Conference (RE), 2015 IEEE 23nd International, (to be published) (2015)
8. Li, T., Horkoff, J.: Dealing with security requirements for socio-technical systems: A holistic approach. In: CAiSE'14. (2014)
9. Ali, R., Dalpiaz, F., Giorgini, P.: A goal-based framework for contextual requirements modeling and analysis. Requirements Engineering **15**(4) (2010) 439–458
10. Moore, A.P., Ellison, R.J., Linger, R.C.: Attack modeling for information security and survivability. Technical report, CMU-SEI-2001-TN-001. (2001)

Modeling the monitoring and adaptation of context-sensitive systems

Jéssyka Vilela, Jaelson Castro

Centro de Informática
Universidade Federal de Pernambuco
Recife, Brazil
{jffv, jbc}@cin.ufpe.br

Abstract. [Context] Context-sensitive systems (CSS) must detect variations in their operating context and adapt their behavior in response to such variations. Hence, their development requires the support of appropriate methods of software engineering. [Objective] This paper describes the activities of the GO2S systematic process to specify the adaptation and monitoring as well as the flow expressions of CSS. [Results] This process guides the software engineer to model the adaptive behavior through contextual design goal models and contextual refinements. [Conclusion] These models explicitly capture what changes in the environment and in the system to be monitored, what to adapt, when to adapt and how to adapt. We illustrate our proposal by applying it to the smart home exemplar.

Keywords: Adaptation, Context, Design Goal Model, Monitoring, Behavior.

1 Introduction

Context-Sensitive Systems use context to provide services and relevant information to their users. They are flexible, able to act autonomously on behalf of users and dynamically adapt their behavior. Hence, these systems must have the following characteristics: monitoring, awareness and adaptability [1]. Considering the inherent complexity and variability of context-sensitive applications, their development requires the support of appropriate methods of software engineering.

The specification of adaptive behavior is an issue addressed with different perspectives. The contexts are used to represent and analyze the variations in i* models resulting from the domain variability in [10]. The work of [8] supports the design and runtime execution of adaptive software systems both at a requirements and architectural level. Another work [2] describes a systematic methodology to design adaptive software systems. Finally, a method to derive the adaptive behavior of Dynamically Adaptive Systems (DAS) from a set of i*models is presented in [7].

In previous works [4][5], we proposed the GO2S (GOals to Statecharts) process, a systematic approach for deriving the behavior (expressed in statecharts) of context-sensitive systems, from requirements models (described as goal models).

In this paper, we detail two critical sub-process of the GO2S process: the modeling of the monitoring and adaptation as well as the specification of the flow expressions. These sub-process define systematic methods to model the system's adaptation and monitoring through the elements of an extended (contextual) design goal model (DGM) and flow expressions [6]. We illustrate our proposal by applying it to the smart home exemplar.

The remainder of this paper is organized as follows. In Section 2, we present our approach to perform the specification of the adaptation and monitoring as well as the behavior of CSS following the GO2S process. Section 3 discusses the contributions of this work and present the venues for future works.

2 Our proposal

The GO2S is an iterative process centered on the incremental refinement of a goal model, obtaining different views of the system (design, contextual, behavioral). The GO2S process consists of six sub-processes: 1) *Construction of design goal model; 2) Specification of contextual variation points; 3) Specification of monitoring and adaptation; 4) Specification of flow expressions, 5) Statechart derivation and refinement* and *6) Prioritization of variants*. In the next subsections, we detail how to specify the adaptation and monitoring (sub-process 3) as well as the flow expressions (sub-process 4) of CSS. The GO2S process assumes that the requirements elicitation and analysis activities were previously performed and a goal model was generated. It is out of scope of this paper to present and discuss all activities of the GO2S process.

2.1 Modeling the adaptation and monitoring of context-sensitive systems

We propose that the specification of the adaptation and monitoring of context-sensitive systems (sub-process 3 of the GO2S process) is performed through refinements in the design goal model [2] extended with contextual annotations [3] which we call contextual design goal model.

Accordingly, in this sub-process we add adaptation design tasks in the contextual DGM. These tasks are required for the adaptation of each requirement that needed to be monitored. Then, they are refined through tasks in AND/OR decompositions that represent the adaptation strategies. The activities required for the modeling the adaptation and monitoring are presented in Fig. 1.

The input of this sub-process is the contextual design goal model that is used by the software engineer to define the critical requirements that requires adaptation. The next activity is the representation of the adaptation management, which we propose to perform through the following activities:

1. **Add a new design task in the root node for adaptation management** (This activity is necessary when the system requires more than one adaptation).
2. **Add design tasks in the parent node previously created for the management of each requirement that must be monitored and adapted** (*ex: Manage gas leak (t3)* in Fig. 2).

3. **Add design tasks to represent the adaptation strategies for each monitored requirement** (ex: *Turn off the oven (t1)* and *Call fire department (t2)* for the task *Manage gas leak (t3)* in Fig. 2).

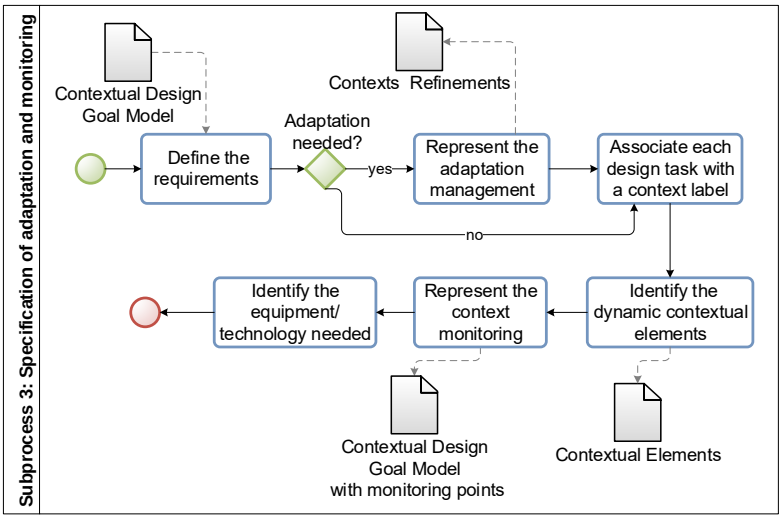


Fig. 1. Activities of specification of adaptation and monitoring sub-process.

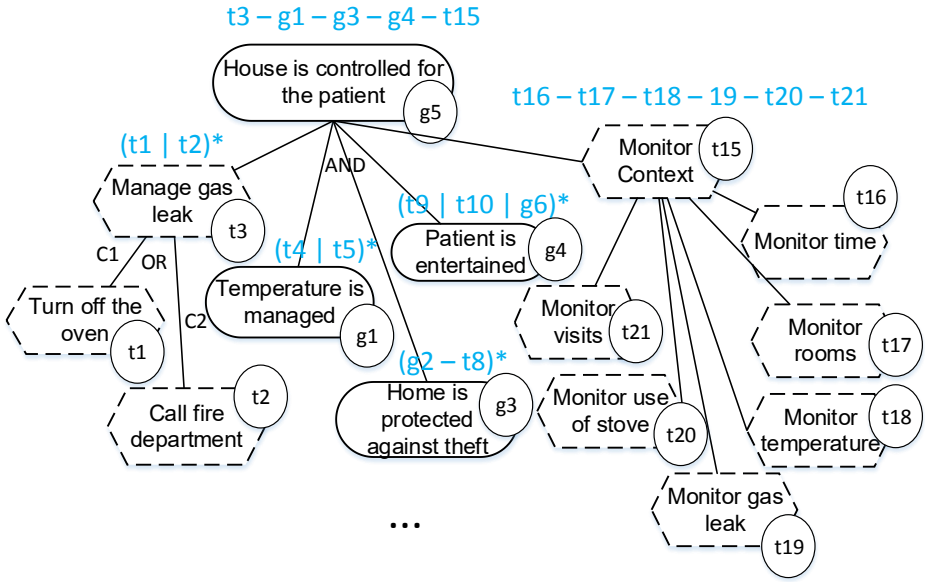


Fig. 2. Excerpt of the behavioral contextual DGM of smart home example.

Note that we should add at least two adaptation design tasks since the variants are the cornerstone for adaptability, a system with only one variant cannot be adaptable.

After the identification of the tasks necessary for the system adaptation, the next activity is to associate each adaptation design task with a context label since these tasks will executed only in certain contexts. In the smart home example, the *Turn off the oven (t1)* design task will be executed when the context C1 holds (*the patient finished using the oven*) and the *Call fire department (t2)* design task will be executed when the context C2 holds (*a gas leak is detected*).

The next step is the identification of the dynamic contextual elements. The dynamic contextual elements are the properties of real-world presented in the facts of context refinements that change their values dynamically. Therefore, the changes in the contextual elements imply in changes in the system context. In our running example, the dynamic contextual elements are the time, rooms, temperature, gas leak, use of the stove, and visits for the patient.

The next activity corresponds to the representation of the context monitoring. Accordingly, we propose the following activities in order to achieve this:

1. **Add a new design task in the root node** (*ex: Monitor Context (t15)* in Fig. 2).
2. **Add design tasks to monitor each dynamic contextual element** *(ex: Monitor time (t16), Monitor rooms (t17), Monitor temperature (t18), Monitor gas leak (t19), Monitor use of the stove (t20)* and *Monitor visits (t21)* in Fig. 2).

We propose to add the adaptation and monitoring activities in the root node since we want to improve the system's modularity and separation of concerns. Accordingly, the related design tasks will be executed concurrently with the system's requirements.

The last activity of this sub-process is the specification of the equipments/technology necessary for monitoring the contexts. In the smart home, the technologies needed are some mechanism to information storage and different types of sensors (presence, temperature, gas leak, stove and luminosity sensors). The outputs of the sub-process 3 of GO2S are the contextual design goal model refined and the contexts refinements.

Having defined the adaptation strategies and the contextual elements that need to be monitored, we can now move on to specify the order of execution of tasks and goals. For this, we rely on flow expressions. This sub-process is described in the next section.

2.2 Modeling the behavior of context-sensitive systems

The goal of the sub-process 4 (Specification of flow expressions) of the GO2S process is to refine the contextual DGM with flow expressions. Flow expressions are a set of enrichments to a goal model that allow specification of the runtime behavior through the execution order of its elements [2]. These expressions are used in the GO2S process as an intermediary model in order to derive the statechart [4-5].

The input of this sub-process is the contextual DGM previously obtained. The first activity is to assign an identification (ID) to each goal and task in the model. Their identification is necessary for reference in the flow expression later. *Gi* was used as ID for goals and *Ti* for tasks and design tasks where *i* is the number of the task.

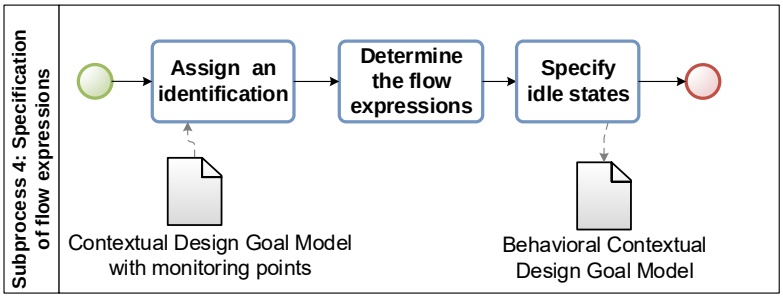


Fig. 3.Activities of Specification of flow expressions sub-process.

After the IDs assignment, the next activity is to define the flow expression for each parent node which describes the behavior of its children elements using the symbols proposed by [2]. The strategy of specifying the children behavior of a parent node can be bottom-up or top-down, the result will be the same. Thereafter, when we reach the root goal, we have the flow expression from the entire system. The resulting flow expressions should be annotated in the contextual DGM as demonstrated in Fig. 2.

A common practice when creating statecharts is to use intermediate states as a point where the system is idle, waiting for some input, such as input selection by the user or for a context to hold. Considering how frequently these states appear, and aiming to reduce visual pollution in the behavioral contextual DGM, such states must be inserted directly in the flow expressions identified as iX, where X is an integer.

The output of this activity is the behavioral contextual DGM. It is the contextual design goal model annotated with flow expressions. This model can represent in unified way all the views developed in the GO2S (contextual, design and behavioral).

3 Ongoing and Future Work

We conducted a controlled experiment in order to evaluate our process. This study was performed using 18 undergraduate and graduate students enrolled in a requirements engineering course divided into two groups with nine subjects each. Each subject of the first group constructed a statechart of the smart home system following the GO2S process (the GO2S group) and each subject of the second group built/developed a statechart without guidance (the control group).

The experiment results are encouraging since the structural complexity of the experimental group was lower and the mean of behavioral similarity was higher than control group. Besides, the subjects agreed that the GO2S process is easy to use indicating that it is understandable. The results of this experiment can be found at [12].

While statecharts are the industry standard and provide an intuitive representation of behavior models, formal analysis is limited and difficult. Hence, we are currently working on an approach to analyze properties of the statecharts generated with the GO2S process. Moreover, we are also investigating the contributions of using ontolo-

gies to the verification of statecharts considering their empirical benefits for requirements engineering identified in a previous systematic literature review [9].

We also expect to develop a case tool to support the process. It should be used to generate the different views (design, contextual and behavioral) of our process. Besides, it is important to derive systematically the other architectural views of CSS. The structural view of context-sensitive systems was already addressed in the work of [11] and the GO2S process addressed the behavioral view. The other views can be incorporated in our process in order to obtain a complete architecture specification.

References

1. Klein, C. et al. A Survey of Context Adaptation in Autonomic Computing. In: 4th International Conference on Autonomic and autonomous Systems (ICAS), 2008. pp. 106–111.
2. Pimentel, J. et al. From requirements to statecharts via design refinement. In: proceedings of the 29th SAC, 2014, pp. 995–1000.
3. Ali, R.; Dalpiaz, F.; Giorgini, P. A goal-based framework for contextual requirements modeling and analysis. Requirements Engineering, v. 15, n. 4, 2010, pp. 439-458.
4. Vilela, J.; Castro, J.; Pimentel; J.; Soares, M.; Lima, P.; Lucena, M. Deriving the behavior of context-sensitive systems from contextual goal models. In: Proceedings of the 30th SAC, 2015.
5. Vilela, J.; Castro, J.; Pimentel; J.; Lima. On the behavior of context-sensitive systems. In proceeding of 18th Workshop on Requirements Engineering (WER), 2015.
6. Boudaa, B.; Hammoudi, S.; Chikh, M.A. ODM-based modeling for user-centered context-aware mobile applications. IN: 3rd International Conference on Information Technology and e-Services (ICITeS), 2013, pp. 1-7.
7. Welsh, K. and Sawyer, P. Deriving Adaptive Behaviour from i* models. In: proceedings of the 4th International i* Workshop (istar 2010). pp. 98-102.
8. Pimentel, J.; Angelopoulos, K.; Souza, V. E. S.; Mylopoulos, J.; Castro J. From Requirements to Architectures for Better Adaptive Software Systems. In: proceedings of the 6th International i* Workshop (iStar 2013). pp. 91-96.
9. Dermeval, D.; Vilela, J.; Bittencourt, Ig.; Castro, J. Isotani, S.; Brito, P.; Silva, A. Applications of ontologies in requirements engineering: a systematic review of the literature. Requirements Engineering, 2015, pp. 1-33.
10. Lapouchnian A. and Mylopoulos, J. In: proceedings of the 5th International i* Workshop (iStar 2011). pp. 96-101.
11. Pimentel. J. et al. Deriving software architectural models from requirements models for adaptive systems: the stream-a approach. Requirements Engineering, v.17, n.4, 2012, pp.259–281.
12. J. F. F. Vilela. A systematic process to derive the behavior of context-sensitive systems from requirements models. MSC Dissertation, Universidade Federal de Pernambuco, Centro de Informática, 2015. Available at: http://www.cin.ufpe.br/~ler/supplement/istar2015/.

A Textual Syntax with Tool Support for the Goal-oriented Requirement Language

Vahdat Abdelzad, Daniel Amyot, Sanaa A. Alwidian, Timothy C. Lethbridge

EECS, University of Ottawa, Ottawa, Canada
{v.abdelzad, damyot, salwidia, Timothy.Lethbridge}@uottawa.ca

Abstract. Most goal-oriented modeling languages, including *i**, Tropos, KAOS and the Goal-oriented Requirement Language (GRL), offer a graphical syntax, sometimes accompanied by a textual interchange format (e.g., in XML). Graphical representations of goal models excel at supporting discussions and at visualizing analysis results. However, creating/modifying goal models is often a tedious task with current graphical environments. Textual languages are often more efficient for creating/ modifying models, in particular large ones. This paper proposes a programming-like textual syntax for GRL supported by an advanced editor for the Eclipse platform. Such syntax and editor enable modelers to create GRL models with complex features (e.g., strategies and contribution overrides) in a way that is simpler than with the most popular GRL editor, namely jUCMNav. The paper also introduces a converter from the GRL textual syntax to jUCMNav, so that models can be visualized and analyzed.

Keywords: Editor · Goal-oriented Requirement Language · Textual Language.

1 Introduction

Graphical modeling languages bring benefits over textual languages in that they can represent information in two dimensions (rather than linearly) using intuitive pictograms and other visual clues. Unsurprisingly, goal modeling languages such as *i**, Tropos, Techne, KAOS and the Goal-oriented Requirement Language (GRL) have chosen graphical syntaxes for supporting visual modeling and analysis activities [5]. However, there are still two important issues in that context: 1) it is difficult to design a graphical modeling language that offers good cognitive fitness for different types of users and purposes, and most goal modeling languages have much room for improvement in that regard [8]; and 2) graphical editors are often cumbersome to use and inefficient for *creating* goal models [9]. Textual syntaxes, although not very useful for visualizing analysis results, are often less cognitively challenging than graphical syntaxes, and are often faster to use for creating/modifying models via intelligent editors and simpler copy/pasting semantics. This is something we have observed with Umple, a textual language that integrates concepts from UML class/state diagrams and patterns with programming languages such as Java [3].

In this context, we have decided to explore the design of a textual syntax for GRL, which is part of the User Requirements Notation (URN) standard [6]. GRL modeling, analysis and transformations are currently supported by jUCMNav [10]. The designed language, called TGRL, supports full GRL, including basic concepts such as intentional elements, links and actors, but also advanced features such as indicators, metadata, strategies, and contribution overrides.

Section 2 provides an overview of our textual syntax, based on standard GRL and jUCMNav's metamodel. The Eclipse-based editor and the converter that transforms TGRL models into jUCMNav ones are introduced in Section 3. Section 4 discusses early experience with the language and its editor, and provides pointers to future work items. Please note that the full grammar and the editor are available online [1].

2 TGRL: A Textual Syntax for GRL

To define the TGRL textual syntax, we used guiding principles inspired from the design of Umple, including simplicity, consistency, and a programming language-like look and feel. In addition, we aligned the syntax and especially keywords with jUCMNav's metamodel (which served as inspiration for GRL in the standard URN metamodel [6], except for several exploratory features) in order to simplify the conversion from/to GRL models in jUCMNav. In addition:

- GRL elements are usually defined through keywords using CamelCase boundaries (e.g., a softgoal intentional element is represented by a `softGoal`).
- String values are surrounded by quotation marks.
- Model element properties and sub-elements (if any) are set inside curly brackets.

```
grl SimpleExample {
    comment "This is a simple TGRL illustrative model"; // Model comment

    actor User {
      // Default name is the ID name, "User" in this case.
      // Goal with specific name and quantitative importance.
      softGoal EasyToUse {name = "Have a system that is easy to use";
                          importance = 100;}
      indicator LowLearningTime; // Indicator definition
    }
    actor System {
      // Goal with qualitative importance, and OR decomposition type
      goal ProvideMainFunctionality {importance = high;
                                     decompositionType = or;}
      task FirstOption {metadata stereotype="SomeValue";}
      task SecondOption {description = "Better alternative";}

      ProvideMainFunctionality decomposedBy FirstOption, SecondOption;
      FirstOption contributesTo User.EasyToUse {hurt}; //Inside element
    }

   // Links defined outside its elements, with quantitative value
   System.SecondOption contributesTo User.EasyToUse {name=C1;50};
   User.LowLearningTime contributesTo User.EasyToUse {name=C2;40};
}
```

Fig. 1. Simple illustrative TGRL model

- Every definition ends with a semicolon except when a pair of curly brackets is utilized to include either sub-elements or properties.

Most elements have a textual identifier (ID) as well as optional *metadata* (name-value pairs). Intentional elements (goals, softgoals, tasks and resources) also have qualitative/quantitative importance values (to their containing actor). For example, Fig. 1 shows the TGRL representation of a simple GRL model with two actors, their intentional elements, and various links. IDs are used as names unless specified otherwise (e.g., a `name` attribute can be quite long, with special symbols). Qualitative values (e.g., `high` for importance, `make`/`hurt` for contributions) and quantitative values (between –100 and 100) can be used interchangeably. Lists can be used for definitions and usages (e.g., see the `decomposedBy` relationship in the example).

As in Umple [3], links can be specified *inside* one element or *outside* the relevant elements, depending on the modeler's preference. In Fig. 1, one contribution is defined inside the `System` actor, whereas two other contributions are defined outside both actors. Note that scoping is also used to resolve potential naming issues. For example, in the contribution inside the `System` actor, task `FirstOption` is local but softgoal `EasyToUse` is defined elsewhere, and hence must be prefixed by its containing actor (leading to `User.EasyToUse`). Dependency links are handled similarly.

TGRL also supports evaluation *strategies* (initial values given to some intentional elements before invoking a propagation algorithm for model analysis) and handles advanced constructs such as strategy inclusion (for reuse), indicator initialization, and value ranges. For example, Fig. 2 adds a group of three strategies to the model in Fig. 1. The first one selects the first task option in the system, and sets the parameters of the `User.LowLearningTime` indicator. During analysis, an indicator converts a strategy's `eval` value to a satisfaction value by comparing it to the `target`, `threshold`, and `worst` values [6]. The second strategy extends the first one (and hence includes its initializations) but overrides existing initializations or adds new ones. The third strategy refines the first one through a *range* of values (in this example: 10, 15, 20, 25, 30, 35, 40), leading to ranges of results for all intentional elements and actors after evaluating the strategy against the model. In jUCMNav, the creation and management of GRL strategies is rather complicated and not user-friendly, especially if indicators and ranges are involved. This is handled in a much simpler and explicit way in TGRL.

```
strategy SelectFirst {
   System.FirstOption = satisfied;
   User.LowLearningTime = {unit="minutes"; target=30.0; threshold=60.0;
                           worst=120.0; eval=90.0;}
}
strategy SelectSecond extends SelectFirst { // Strategy inclusion
   System.FirstOption = none; // Overridden
   System.SecondOption = 100; // Added, quantitatively this time
}
strategy RangeExample extends SelectFirst {
   System.FirstOption = {start = 10; end = 40; step = 5;}
}
strategyGroup MyGroup includes SelectFirst, SelectSecond, RangeExample;
```

Fig. 2. Sample GRL strategy definitions in TGRL

TGRL also supports advanced model modifications such as *contribution overrides*, which can be applied to a GRL model to change the weights of existing contribution links, prior to the evaluation of strategies [6]. Overrides are useful when stakeholders disagree on the weights of contributions; different options can be evaluated while having only one model to manage [9]. As shown in Fig. 3, contribution overrides can be grouped and their new values can be specified via a name given to the modified contribution (in Fig. 3, one contribution was named `C1` and another one `C2`). The syntax is quite similar to those of TGRL strategies; as illustrated by `SecondOverride`, contribution overrides can `extend` others, possibly with ranges of values.

The last example is shown at the bottom of Fig. 3, and is concerned with URN links, which are user-specified typed links that can connect any pair of elements in a URN model [6]. In this example, a new type of link (`independentFrom`) is defined and then used to connect two actors in the model. URN links and metadata are constructs that are useful in extending or *profiling* URN to specific domains.

```
// Contribution overrides
contributionGroup SomeOverrides includes FirstOverride, SecondOverride;
contribution FirstOverride {
    C1 = 30;
    C2 = make;
}
contribution SecondOverride extends FirstOverride {
    C1 = {start = -40; end = 0; step = 10;}
}

// URN links
link independentFrom; // Link type definition
User independentFrom System; // Link instance between two actors
```

Fig. 3. Sample GRL contribution overrides and URN links

3 TGRL Editor

We developed a TGRL editor for the Eclipse environment. We specified our grammar with *Xtext* [11], often used for the development of textual domain-specific languages. The TGRL editor supports syntax highlight (as shown in the code snippets from the previous figures), an outline view, annotation of syntactic errors, content assistance, and code formatting. Fig. 4 gives an overview of the editor.

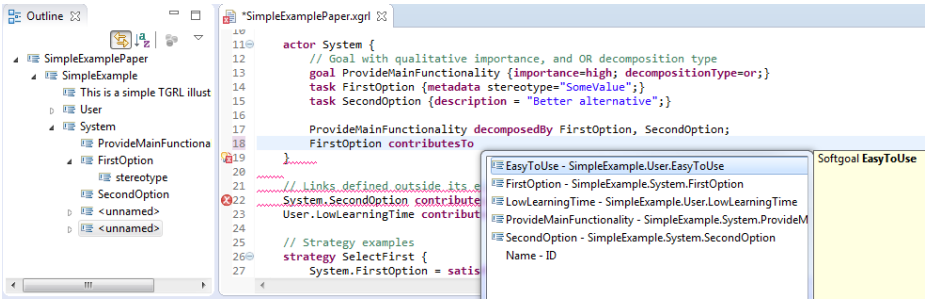


Fig. 4. Overview of the TGRL editor, with content assistance

The modeler, using Control-Space, can invoke code completion at any moment. Not only is this available for the keywords found in the grammar, this is also available for references to existing elements. For example, in Fig. 4, several suggestions are provided as potential targets of an incomplete contribution link.

Together with the editor, we provided a mechanism (*converter*) that enables the transformation of a textual model to a graphical model in jUCMNav, ready to be analyzed and visualized. The transformation was developed using *Acceleo* [2], which is a pragmatic model-to-text transformation language. We have chosen to use such transformation rather than a model-to-model one because jUCMNav stores its models in textual (XML) files. Further validation of the models, based on GRL's semantics, is performed through rules deployed in the body of the converter. This transformation does not handle the layout of diagrams, but jUCMNav has several features for creating views of a model and for automatically laying out elements. For example, Fig. 5 shows the GRL model corresponding to the ongoing example, as imported by jUCMNav (with automatic and some manual layout). The evaluation of the strategy `SelectFirst` is also shown, using quantitative values.

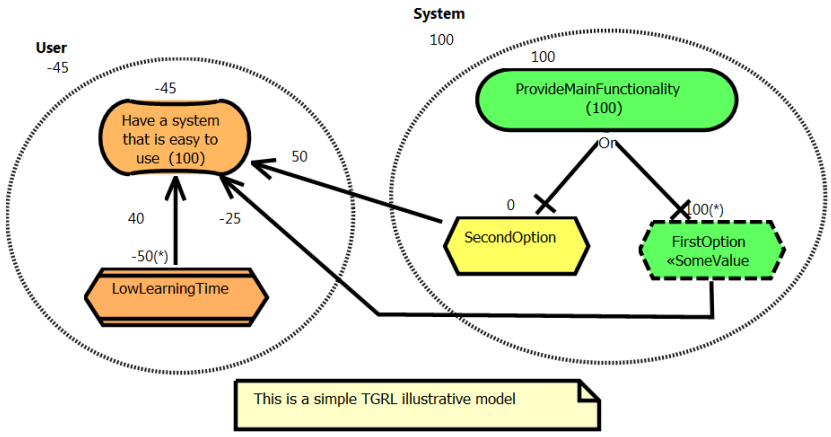


Fig. 5. Sample GRL model imported in jUCMNav, with a strategy evaluated quantitatively

4 Discussion and Future Work

In this paper, we illustrated a new textual syntax for GRL, called TGRL, with a full coverage of the language. TGRL is supported by a feature-rich Eclipse-based editor, supplemented by an automated conversion to GRL models readable by jUCMNav. This work contributes a simple and practical way of creating/modifying GRL models.

The idea of having a textual syntax for GRL is not new. In the first draft of the GRL language (from 2001), Liu and Yu provided a textual grammar and an XML-based interchange format [7]. TGRL is however not based on this earlier attempt. Rather, it focuses on adding a textual syntax to an already existing metamodel definition. TGRL also covers many concepts that did not exist in [7], such as indicators, strategies, contribution overrides, metadata and URN links. Formal Tropos also has a textual syntax [4] but its goal modeling syntax (outer layer) is more declarative, ver-

bose, and limited in scope. Formal Tropos however supports an inner layer for declaring constraints on attributes and supports temporal logic properties. Tool support for Formal Tropos (T-Tool) does not include a feature-rich editor. To our knowledge, TGRL is the first tool-supported textual syntax for an *i**-like modeling language.

In terms of coverage and usability, one of us (S.A. Alwidian, not involved in the design and implementation of the grammar and tool, to avoid bias) validated the language and the tool through two simple examples from the GRL literature. Her feedback was overall positive, and solutions to some issues raised with earlier versions were incorporated in the grammar and the tool to improve their usability.

This new technology opens the door to many future opportunities. The language and the tool obviously require further and more rigorous validation, for example based on how well existing models are supported. However, they also enable comparisons with graphical tools (e.g., jUCMNav) in terms of efficiency and usability for model creation and manipulation tasks. One important feature currently missing is the availability of a transformation from jUCMNav to TGRL, which would enable modelers to go back and forth between the two representations. The editor could also be improved by the inclusion of additional static semantic rules to ensure the correctness of the GRL models created (e.g., to prevent cyclical contribution links or the mixed use of quantitative/qualitative values, or to detect bad smells and anti-patterns). We also envision opportunities to combine TGRL (for goals) with Umple (for design and implementation) as they provide complementary concepts.

References

1. Abdelzad, V.: Textual Modeling Language for GRL (2015) https://github.com/vahdat-ab/TGRL
2. Acceleo (2015) http://www.eclipse.org/acceleo/
3. Forward, A., Badreddin, O., Lethbridge, T.C., Solano, J.: Model-driven rapid prototyping with Umple. Softw., Pract. Exper. 42(7), 781–797 (2012) http://umple.org/
4. Fuxman, A. et al.: Specifying and analyzing early requirements in Tropos. Requir. Eng. 9, 2, 132–150 (2004).
5. Horkoff, J., Yu, E.S.K.: Comparison and evaluation of goal-oriented satisfaction analysis techniques. Requir. Eng. 18(3), 199–222 (2013)
6. ITU-T: Recommendation Z.151 (10/12): User Requirements Notation (URN) – Language definition. Geneva, Switzerland (2012)
7. Liu, L., Yu, E.: GRL – Goal-oriented Requirement Language. University of Toronto, Canada (2001) http://www.cs.toronto.edu/km/GRL
8. Moody, D.L., Heymans, P., Matulevičius, R.: Visual syntax does matter: Improving the cognitive effectiveness of the *i** visual notation. Requir. Eng. 15(2), 141–175 (2010)
9. Mussbacher, G., Amyot, D., Heymans, P.: Eight Deadly Sins of GRL. 5th International *i** Workshop (iStar 2011). CEUR-WS, Vol-766, 2–7 (2011)
10. Roy, J.-F. Kealey, Amyot, D.: Towards Integrated Tool Support for the User Requirements Notation. SAM 2006. LNCS 4320, 198–215. Springer (2006) http://softwareengineering.ca/jucmnav
11. Xtext (2015) http://www.eclipse.org/Xtext/

Using i^* for Transformational Creativity in Requirements Engineering*

Sushma Rayasam and Nan Niu

Department of EECS, University of Cincinnati
Cincinnati, OH, USA 45221
rayasasa@mail.uc.edu, nan.niu@uc.edu

Abstract. Requirements engineering (RE) techniques that promote creativity can lead to product innovation and business competitiveness. To investigate the role of i^* in creative RE, we report a study involving nine analysts who generate creative requirements for the meeting scheduler in a transformational way. Our results reveal the interdependency of exploratory creativity and transformational creativity, and uncover tasks as starting points for creative goal modeling. Our work also offers process insights which can guide the development of automated support for transformational creativity in RE.

1 Introduction

In today's tech-savvy world, it is crucial for companies to leverage innovation and creativity to come up with products which will sustain the test of time. The importance of creativity in requirements engineering (RE) is recognized and specially emphasized for developing software-intensive systems which address critical business challenges and which are in highly competitive contexts [6].

Creativity, in general, is the ability of an individual or a group to think of new and useful ideas; however, because creativity plays a role in many fields (e.g., business, arts, etc.), defining creativity can be context-dependent. Creativity in RE, according to Maiden *et al.* [7], is the capture of requirements that are both novel and appropriate. Maiden *et al.* [7] also distinguish between creativity and innovation by relating innovation to downstream software development, that is, implementation of creative requirements leads to system innovation.

Current creativity techniques tend to rely heavily on expert facilitation and manual effort. An example is the creativity workshop where stakeholders are asked to perform brainstorming and creative thinking during requirements elicitation [6]. Manual work often results in undocumented rationales behind the produced requirements, making the creativity process less systematic.

Goal models, such as i^* [12], offer structure which can provoke systematic creative exploration and enable a wide variety of analyses (e.g., [4]). As will be surveyed in Section 2, researchers have developed creativity methods to support

RE. Most contemporary support is of *exploratory* nature aiming to traverse a space of possibilities. Support for *combinational* creativity in RE also emerges recently, attempting to make unfamiliar connections between familiar possibilities. Less supported is *transformational* creativity which challenges the constraints on the search space and seeks new ideas in different domains or paradigms.

In this paper, we report a preliminary study on how human analysts perform transformational creativity using i^* models. Specifically, we recruited 9 upper-division computer science students and asked them to individually generate creative requirements of the meeting scheduler i^* models in a transformative way. Our research goal is to uncover the patterns used and the challenges faced by the analysts. Our observations and lessons learned can contribute to a more systematic process, along with the identification of potential automated support, of transformational creativity in RE. In what follows, we review related work in Section 2. Section 3 presents our study design, Section 4 analyzes the results, and finally, Section 5 concludes the paper.

2 Background and Related Work

RE, framed as a creative problem solving process, plays a key role in product innovation and system sustainability [7]. Being novel and being appropriate are intrinsic to creativity [3]. Creativity in RE can be categorized into three groups on the basis of the techniques and heuristics employed [3]. The first is *exploratory creativity* obtained by traversing a search space of partial and complete possibilities. Techniques, such as brainstorming, snowballing, and serial association, can be used. Operating on the KAOS goal models, Lutz *et al.* [5] perform analysis to explore requirements in a space defined by obstacles for a safety-critical, autonomous system.

The second is *combinational creativity* that is performed by making unfamiliar connections between objects in the same search space. Random and fixed stimuli are two techniques through which combinational creativity can be achieved [3]. Bhowmik *et al.* [1, 2] leverage topic modeling to identify concepts familiar to stakeholder groups and exploit part-of-speech tagging to automatically generate unfamiliar combinations.

The third, and the highest form of creativity according to Boden [3], is *transformational creativity* which can be achieved by changing the rules that govern and structure the conceptual space. Compared to exploratory creativity and combinational creativity, less (automated) support for transformational creativity exists. An exception is the semantic service search & composition (S^3C) tool developed by Zachos and Maiden [13] that retrieves Web services in domains analogical to a current requirements problem. Even the S^3C tool does not fully take advantage of the structural information embedded in goal models. We argue that more structure (e.g., overview of the current problem domain, dependencies within the domain, etc.) is needed to facilitate transformational creativity in RE. Next we describe the research design to investigate the use of i^* for transformational creativity.

Table 1. Excerpt of existing modeling constructs (see [11] for the complete version)

Actor	Goal	Softgoal	Task	Resource
Mtg. Initiator (7)	Mtg. Be Scheduled (14)	Low Effort (10)	Attend Mtg. (4)	Details (3)
Mtg. Scheduler (5)	Agreeable Mtg. Date (4)	Quick (4)	Organize Mtg. (3)	Proposed Date (2)
Mtg. Participant (5)	Solicit Response (4)	Accuracy of Constraints (4)	Determine Mtg. Date (3)	Agreement (2)
Important Participant (5)	Collect Timetables (4)	Collection Effort (3)	Participate in Mtg. (3)	Facilities Confirmed Room (1)
...	...	...	...	...

3 Study Design

Our main research objective is to examine how human analysts use i^* to perform transformative creativity, paying special attention to the structural elements of i^* in creative RE. We set out to uncover the commonly used strategies and to identify areas where additional support can be provided to facilitate creative goal modeling. To accomplish the objective, we chose participant observation as our research method. To recruit participants, we sent e-mail invitations to the upper-division students (juniors, seniors, Master's, and Ph.D.s) in our department who have already learned or practiced goal modeling (i^*, KAOS, or other forms deemed as appropriate by the respondents). We regarded having industrial experiences as desired but not as mandatory. Nine participants (3 females and 6 males; 4 seniors and 5 graduates) were recruited to voluntarily take part in our study, all of whom knew goal modeling from their educational backgrounds. Five participants reported 1 to 4 years of software development experience in industry, though none had used goal modeling in their industrial projects.

We selected the meeting scheduler i^* models for the participants to perform transformational creativity. Our rationale is three-fold. First, meeting scheduler is a common problem scenario, allowing the participants to readily gain familiarity and practice creativity. Second, meeting scheduler serves as a canonical example in goal modeling, making it relatively easy for us to depict the current domain by consulting to the relevant literature. Third, meeting scheduler is framed in the early-RE phase [12] in which business goals and alternatives are still explored. We therefore consider the early-RE phase is where the requirements tend to be most creative.

We prepared 3 types of materials to help participants begin the transformational creativity task: (1) thirteen references from the literature containing meeting scheduler in i^* notations; (2) three representative graphical models; and (3) existing modeling constructs sorted by their frequencies of occurrence. All the materials are available in [11]. The 13 references range from a conference presentation to a dozen peer-reviewed papers. We identified them by manually searching the proceedings of the RE conference and i^* workshop series, and by following the references cited in relevant papers.

We then chose 3 graphical models — 1 strategic dependency model and 2 strategic rationale models — to illustrate the goal-modeling constructs and their relationships [11]. Finally, we extracted the structural elements from the 13 references, grouped those elements by their types (actor, goal, softball, task, and resource), and ranked the elements by number of appearances. Table 1 shows an excerpt of the extraction results. We expect our prepared materials (references,

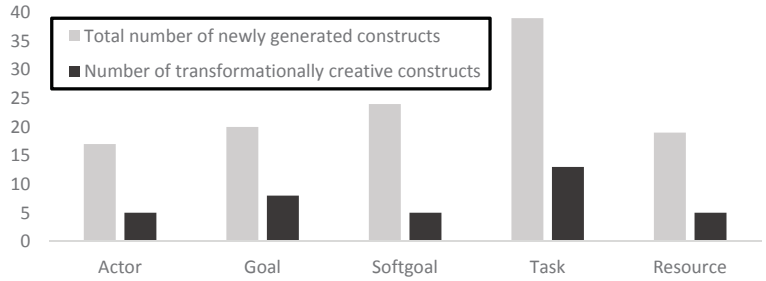


Fig. 1. Modeling constructs grouped by i^* categories.

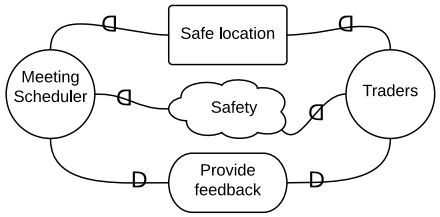
graphical models, and extracted constructs) to act as a depiction of the current i^* meeting scheduler domain — a baseline that the participant-analyst can use to conduct transformational creativity.

4 Results and Analysis

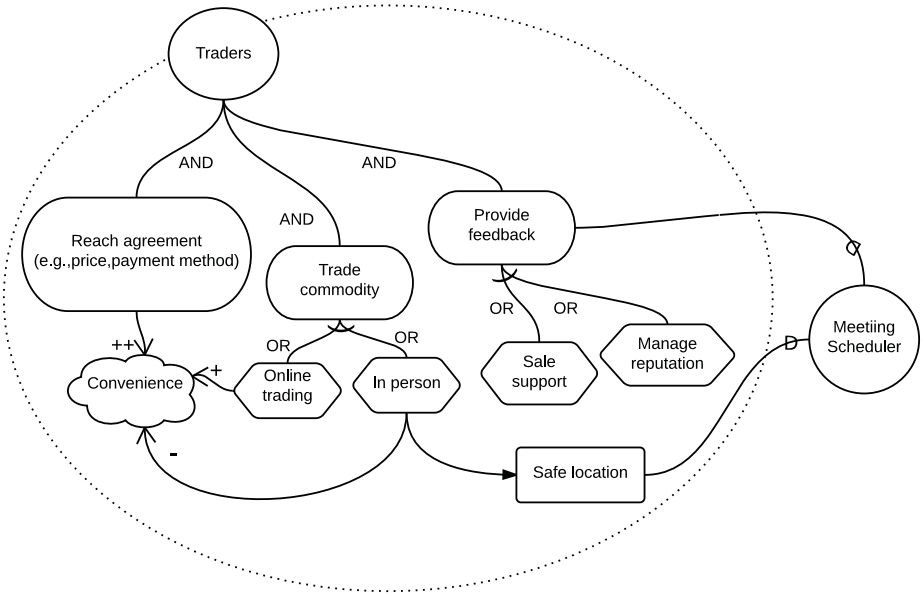
From the i^* models generated, we identified *new* constructs to be those that had not appeared in the literature. This was done via a manual comparison with the complete version of Table 1 [11]. We further judged whether a new construct is transformationally creative. The judgments were drawn based on *how* the constructs were generated; specifically, if a new domain (e.g., catering) was behind a newly generated construct, then the construct was regarded as *transformationally creative*. Figure 1 compares these numbers.

Several observations can be made from Figure 1. First, although a total of 119 new i^* constructs were generated, the transformationally creative ones accounted for a small portion (30.2%). A closer inspection revealed that a majority constructs were of *exploratory* nature, that is, they were identified by surfacing the possible elements within the *same* meeting scheduling domain. Examples resulted from exploratory creativity include the actor "Meeting Secretary" and the softgoal "Quick Supply". This suggests that exploratory creativity is not only easier to conduct, but potentially a precondition for transformational creativity. In another word, the analyst would need to explore the current domain before transforming to a new one. Second, softgoals are less likely to be transformationally creative ($\frac{5}{24} = 20.8\%$) than other i^* modeling types (32.3% on average). We speculate a main reason is that a typical domain has about a dozen softgoals significant to software architecture [10] and these softgoals already appear in existing i^* models. Another reason is due to softgoal's terminological interference [9], e.g., different modelers use the same term to label different softgoals. Third, the participants in our study produced the greatest number of tasks (39), among which 1/3 were creative in a transformational manner. The rich set of tasks not only helps to manage softgoal's terminological interference [8], but also represents a common starting point for transformational creativity.

In addition to the structural analyses, we examined the newly generated i^* constructs based on the semantics. A majority of the constructs could be grouped into 3 semantic clusters: remote participants (e.g., video conferencing, virtual



(a) strategic dependency



(b) strategic rationale

Fig. 2. Sample transformationally creative i^* models.

meeting), services (e.g., technology, food, transportation), and pre/post work (e.g., document sharing and distributing). As mentioned earlier, these modeling elements fall into the meeting scheduler domain and therefore their identification counts more as exploratory creativity than transformational creativity.

One of the most transformationally creative i^* models in our study linked meeting scheduling with online trading. For illustration, Figure 2 shows strategic dependency (SD) and strategic rationale (SR) models. According to the analyst, the key was to identify some *bridging node* to enable domain transformation. In this case, the "safety" (softgoal) "safe location" (resource) concerns helped connect meeting to the trading domain. Further modeling the core constructs in trading (the new domain) not only required temporary ignorance of meeting scheduling (the current domain), but also facilitated discovery of new transformative relationships. For instance, "provide feedback" of Figure 2b could be used by the traders (meeting participants who do transactions instead of just attending) to influence meeting scheduling (e.g., having a new requirement for reputation management). Our work thus indicates a process: identifying bridging node $\rightarrow$ modeling new domain $\rightarrow$ refining transformative relationship, through which automated support can be developed.

5 Summary

Our study is designed to better understand how i^* is used for transformational creativity in RE. The results show that transformational creativity intertwines with (and possibly requires) exploratory creativity, and that tasks (as opposed to softgoals) serve as common starting points for transformational creativity. In addition, some concrete process insights (e.g., connecting new domain via bridging node) are obtained that could direct automated tool development.

Future work can be carried out in several avenues to overcome the limitations of our preliminary study. First, involving more diverse and heterogeneous participants will uncover more patterns used and struggled faced in creative goal modeling. Second, different ways of depicting (visualizing) the current domain can be researched and compared. Finally, measures and metrics could be defined to help guide the (transformationally) creative RE process.

Acknowledgments. *This research is partially supported by the U.S. NSF (National Science Foundation) Grant CCF-1350487.*

References

1. T. Bhowmik, N. Niu, A. Mahmoud, and J. Savolainen. Automated support for combinational creativity in requirements engineering. In *RE*, pages 243–252, 2014.
2. T. Bhowmik, N. Niu, J. Savolainen, and A. Mahmoud. Leveraging topic modeling and part-of-speech tagging to support combinational creativity in requirements engineering. *Requirements Engineering*, (to appear).
3. M. Boden. *The creative mind: myths and mechanisms*. Routledge, 2003.
4. J. Horkoff and E. Yu. Comparison and evaluation of goal-oriented satisfaction analysis techniques. *Requirements Engineering*, 18(3):199–222, 2013.
5. R. Lutz, A. Patterson-Hine, S. Nelson, C. Frost, D. Tal, and R. Harris. Using obstacle analysis to identify contingency requirements on an unpiloted aerial vehicle. *Requirements Engineering*, 12(1):41–54, 2007.
6. N. Maiden, A. Gizikis, and S. Robertson. Provoking creativity: imagine what your requirements could be like. *IEEE Software*, 21(5):68–75, 2004.
7. N. Maiden, S. Jones, I. Karlsen, R. Neill, K. Zachos, and A. Milne. Requirements engineering as creative problem solving: a research agenda for idea finding. In *RE*, pages 57–66, 2010.
8. N. Niu and S. Easterbrook. Analysis of early aspects in requirements goal models: a concept-driven approach. *Transactions on Aspect-Oriented Software Development*, III:40–72, 2007.
9. N. Niu and S. Easterbrook. Managing terminological interference in goal models with repertory grid. In *RE*, pages 296–299, 2006.
10. N. Niu, L. Xu, J. Cheng, and Z. Niu. Analysis of architecturally significant requirements for enterprise systems. *IEEE Systems Journal*, 8(3):850–857, 2014.
11. S. Rayasam. Transformational creativity in requirements goal models. *Master's thesis*, Department of EECS, University of Cincinnati, 2015.
12. E. Yu. Towards modeling and reasoning support for early-phase requirements engineering. In *RE*, pages 226–235, 1997.
13. K. Zachos and N. Maiden. Inventing requirements from software: an empirical investigation with Web services. In *RE*, pages 145–154, 2008.

Analyzing Second-Order Dependencies in i*

Mohammad Hossein Danesh[1], Eric Yu [2, 1]

[1]Department of Computer Science, University of Toronto, Toronto, Canada
[2]Faculty of Information, University of Toronto, Toronto, Canada

{danesh,eric}@cs.toronto.edu

Abstract. Dependencies among intentional actors is a fundamental feature of i* Modelling. By depending on others, an actor can achieve much beyond what it can by itself. At the same time, the dependent actor becomes vulnerable to the failings of dependees. However, even when dependees are fully fulfilling expectations, over time, depending on other actors can result in structures that are hard to change. By analyzing second-order dependencies, i.e., dependencies among (first-order) dependencies, we determine the extent to which a dependency is depended on by other dependencies. The more a dependency is depended on by other dependencies, the more likely it is to become a barrier to change. Our approach is a model-based formulation of the concept of rigidity in the study of dynamic capabilities in strategic management. The i* models are used to analyze resistance to change in socio technical structures.

Keywords: i* Dependencies, Barriers to Change, Second-order Dependencies

1 Introduction

The importance of dealing with change and enabling adjustment to changing requirements has been studied in both management and Information System (IS) design [1, 2]. The importance of alignment and realignment of business and technical architectures with respect to changes is identified by the literature [3]. As a result, it is crucial to consider the intertwined nature of business and IS when modeling and representing enterprise requirements [4]. The challenge of dealing with change is twofold: (1) the ability to identify changing conditions and adjust to satisfy new requirements (either automated or with human intervention); and (2) the flexibility of enterprise capabilities and organizational settings to accommodate change, create new services or information systems and support their deployment [5].

While many researchers in IS and software engineering have attempted to overcome the first challenge, not many approaches exist that can analyze social and technical inflexibilities in an enterprises [6]. In this paper a model-based formulation of potential inflexibilities is presented using i* models that describe enterprise capabilities, their dependencies and alternatives [7]. The formulation investigates the structure of dependencies among capabilities (modeled as specialized actors) and other actors within the organization to analyze the commitments resulting from networks of de-

pendencies. This formulation is motivated by research in strategic management about the positive and negative consequences of collaboration [8]. While collaboration can produce better qualitative and quantitative achievements, it also entails vulnerability as actors committed to such dependencies become confined in their future alternatives [8, 9]. The analysis of potential inflexibilities in this paper is demonstrated on a hypothetical educational institute presented in our earlier work [5].

2 Related Work in IS Design to Enable Change

Two classes of research are presented that deal with adaptation of information systems. The first category focuses on the context and changing requirements while the second category addresses architectural reconfiguration and modification.

Souza et al [2] deal with the adaptation challenge from a requirement perspective and propose capturing evolutionary requirements of information systems in order to enable automated or semi-automated adjustments. Zdravkovic et al [10] propose using enterprise models to capture the business context and its variation points to enable runtime adjustment of services in accordance to changes in the capability context.

Researchers in software architecture analysis address change with a particular focus on the effort and process required to enable implementation and modification of a software system to accommodate changes in stakeholder requirements. For example, Bengtsson et al [11] propose a scenario oriented analysis to enable evaluation of alternative software architectures with regards to specified change scenarios. Bohner [12] proposes structural analysis of the software architecture to study the rippling effect of a change, this enabling estimation of effort and time required to implement changes. Building on impact analysis approaches, De Boer et al [3] propose identification of rippling effects of a change in an Archimate model to allow realignment of the technical and business architectures.

Both of the discussed categories enable adjustment of information systems in accordance to changing context, hence focusing on overcoming the first adaptation challenge. However enterprises face emergent needs that arise as a result of interactions of social and technical entities within the organization and its ecosystem [13, 14]. Studies indicate that enterprise capabilities can resist to changing context and implementation of emerging requirements if changes contradict the capability evolution path (the evolution path is shaped by the history of decisions made over its lifetime) [8, 9]. Accommodating such changes requires architectural governance that can identify socio-technical inflexibilities that constitute the second adaptation challenge [13, 15].

3 Uncovering Potential Inflexibilities using Second-order Dependencies in i*

The i* modeling framework is known for its ability to capture intentions of different actors when modeling enterprise requirements. As part of i*, dependencies among actors is modeled to enable analysis regarding how actors rely on one another to satis-

fy goals and softgoals, acquire resources, and perform tasks. However such dependencies entail commitment among actors and can introduce barriers to change. In this section we propose a method to investigate the most influential elements in i* networks of dependencies to enable identification and analysis regarding highly influential dependencies that can cause inflexibilities.

To determine which dependencies have a higher potential of being barriers to change the degree of coupling among dependencies is (algorithmically) computed using second-order dependencies in i*. The approach enables identification of dependencies that have high impact on the overall network of dependencies. In addition it can enable qualitative and quantitative analysis regarding how a certain change will impact the network of dependencies and enterprise capabilities. An example of qualitative impact analysis on IT and organizational capability alignment is discussed in a related work [5].

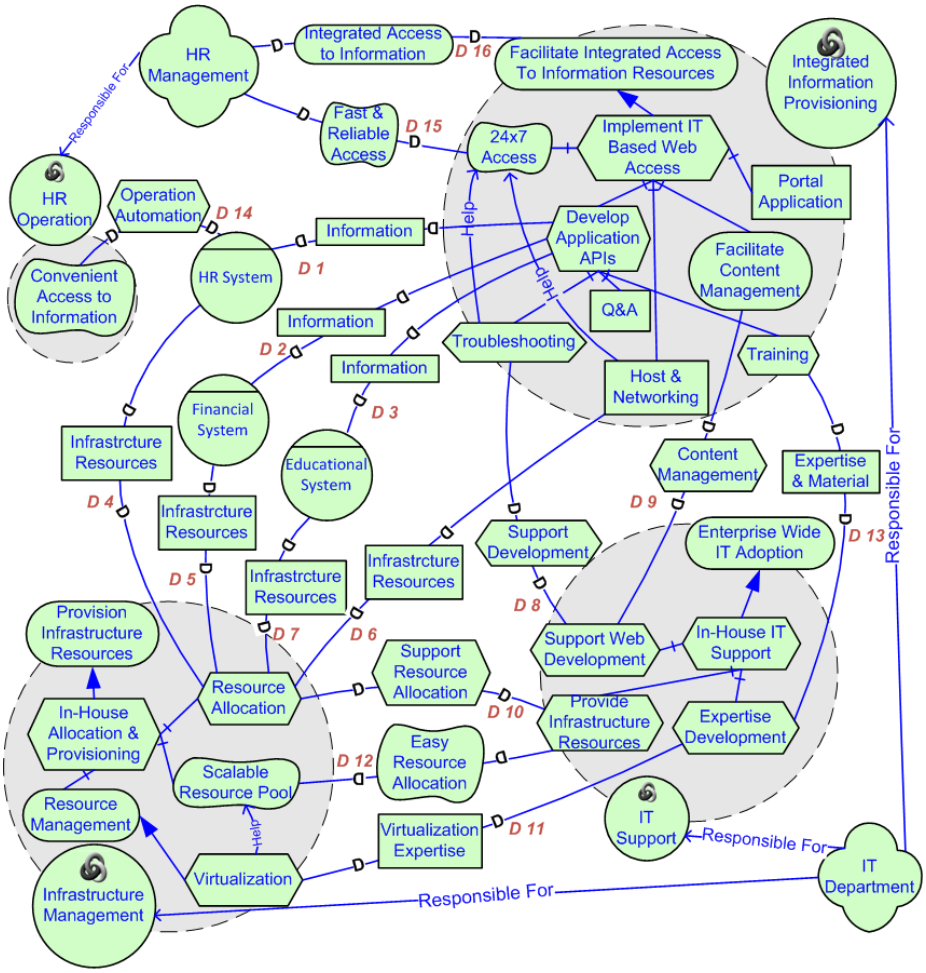


Fig. 1. An i* model of IT Capabilities

In an earlier work [7], an extended version of i* was introduced to enable the modeling of enterprise capabilities and their development, orchestration and deployment alternatives. Using that extension a method for analyzing second-order dependencies is introduced in the context of a hypothetical educational institute. Figure 1 depicts a snapshot of the enterprises IT capabilities and their relations to organizational actors and information systems. A capability is depicted as a specialized type of actor.

A second-order dependency is defined as the reliance of one dependency to another to the extent that it cannot perform with the required quality unless the former dependency is satisfied. In other words second-order dependencies refer to dependencies among (first-order) dependencies.

To extract second-order dependencies one can investigate the strategic rationale model of i*. If the dependee-side element of an i* dependency (element which resides in dependee) such as *D 7* in Fig. 1 (the dependee-side element in *D 7* is *Resource Allocation*), and that element itself is dependent on some other actor as is the case with *Resource Allocation* (which is dependent on the *IT support* capability), then *D 7* is dependent (second-order) on *D 10*.

If the source element of a dependency is comprised of sub-elements where sub-elements are identified through contribution, decomposition and mean-end links (for softgoals, tasks and goals respectively); then a second-order dependency exists from the dependency to each of the dependencies of sub-elements. For example *Virtualization* (a task of *Infrastructure Management* capability) contributes to the dependee-side element of *D 12* (*Scalable Resource Pool*) and depends on *Virtualization Expertise* (resource) which is provided by the *IT Support* (capability), therefore a second-order dependency exists from *D 10* to *D 12*. The second-order dependency exists as *Easy Resource Allocation* (D 12) relies on setup and engineering of the virtualization infrastructure which provides *Scalable Resource Pool*.

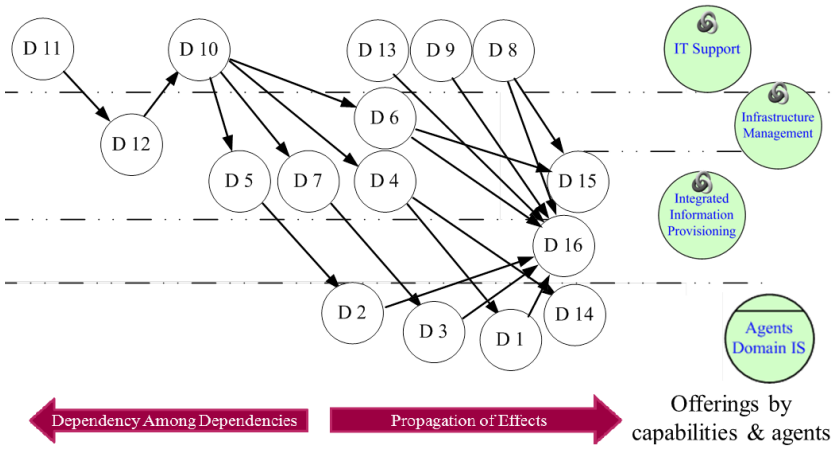


Fig. 2. Dependency Propagation Graph

The dependency propagation graph presented in Figure 2 enables analysis of the rippling effects of dependencies among actors in i*. Its construction can be automated

in a tool using the rules described earlier in this paper. The directions of arrows depict the path in which rippling effects of a change can propagate, i.e., the opposite direction of the dependencies in the SR model. In this graph the dependencies are grouped into rows according to the dependee actors. The grouping facilitates visual analysis regarding how changing certain capabilities or systems will impact the overall network of dependencies. With additional information the graph can serve as a roadmap to quantify economical contribution of each dependency and its role in value creation.

According to graph presented in Figure 2, *D 11* which refers to *Virtualization Expertise* provided by the *IT Support* capability to the *Infrastructure Management* capability of Figure 1, is a sensitive element as deficits in resources to design and govern a virtual infrastructure can have extensive impacts on the functionality and use of information systems across the enterprise. Furthermore making changes to the process (i* task) by which this resource is provided, i.e., *Expertise Development* in *IT Support*, can impact many other applications and organizational dependencies. Hence when making decisions regarding its evolution, one should carefully consider consequences and alternatives.

4 Conclusion and Future Work

Building the flexibility required to enable enterprise transformation is a major concern in both management and IS research. While many have proposed approaches to deal with automated adjustment of IS, there is a lack of methods that allow analysis regarding inflexibilities that arise in a socio-technical context. An approach that enables analysis and identification of potential inflexibilities is introduced by investigating second-order dependencies in an i* model of enterprise capabilities.

As future work, in order to fully recognize causes of rigidity, one needs to investigate the degree of impact that a certain sensitive dependency has. This can be achieved through assignment of quantitative measures to the edges of the dependency propagation graph. The measures can be assigned as a weight to depict the importance of the second-order represented by the edges. If the dependency is resulted from a softgoal, the contribution links can serve as a roadmap for assigning values. Such quantitative measures assist human judgement regarding the sensitivity of an element and how it can cause barriers to change.

The results of the analysis can be used at design time to enable accurate planning and mitigation of the risks imposed by any potential inflexibility. In the case presented in this paper, careful planning and consideration in training human resources with the skillsets to manage a virtual infrastructure should be a major concern at the design time. Furthermore the dependency graph can be used at runtime to monitor and measure potential inflexibilities in order to alert the changes in the probability of some dependency causing inflexibility.

Analyzing and interpreting the significance of second-order dependencies without tool support that points to the source i* elements is difficult and reduces the practical usage of the method. Furthermore as the models scale and enterprises grow creation

of the graph requires automated tool support that can take an i* model and produce second-order dependencies based on the proposed algorithm.

References

1. Combs, J.G., Ketchen Jr., D.J., Ireland, R.D., Webb, J.W.: The role of resource flexibility in leveraging strategic resources. J. Manage. Stud. ***48***, 1098–1125 (2011)
2. Silva Souza, V.E., Lapouchnian, A., Mylopoulos, J.: (Requirement) Evolution Requirements for Adaptive Systems. In: Proc. of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, pp. 155–164. IEEE (2012)
3. de Boer, F.S., Bosanque, M.M., Bass, L., Garlan, D., Ivers, J., Little, R., Nord, R., Stafford, J.: Change impact analysis of enterprise architectures. In: Proceedings of the 2005 IEEE International conference on Information reuse and integration (IRI 2005), Las Vegas, Nevada, USA. IEEE Computer Society, Los Alamitos (2005)
4. Jarke, M., Loucopoulos, P., Lyytinen, K., Mylopoulos, J., Robinson, W.: The brave new world of design requirements. Information Systems 36, 992–1008 (2011)
5. Danesh, M., Yu, E.: Analyzing IT Flexibility to Enable Dynamic Capabilities. In: Persson, A. and Stirna, J. (eds.) Advanced Information Systems Engineering Workshops. pp. 53–65. Springer International Publishing (2015).
6. Buckl, S., Schweda, C.M.: Classifying Enterprise Architecture Analysis Approaches. In: Poler, R., van Sinderen, M., Sanchis, R. (eds.) IWEI 2009. LNBIP, vol. 38, pp. 66–79. Springer, Heidelberg (2009)
7. Danesh, M.H., Yu, E.: Modeling enterprise capabilities with i*: reasoning on alternatives. In: Iliadis, L., Papazoglou, M., Pohl, K. (eds.) CAiSE Workshops 2014. LNBIP, vol. 178, pp. 112–123. Springer, Heidelberg (2014)
8. Teece, D.J., Pisano, G., Shuen, A.: Dynamic capability and strategic management. Strateg. Manag. J. ***18***, 509–533 (1997)
9. Leonard-Barton, D.: Core capabilities and core rigidities: a paradox in managing new product development. Strateg. Manag. J. 13, 111–125 (1992)
10. Zdravkovic, J., Stirna, J., Henkel, M., Grabis, J.: Modeling business capabilities and context dependent delivery by cloud services. In: Salinesi, C., Norrie, M.C., Pastor, Ó. (eds.) CAiSE 2013. LNCS, vol. 7908, pp. 369–383. Springer, Heidelberg (2013)
11. Bengtsson, P., Lassing, N., Bosch, J., van Vliet, H.: Architecture-level modifiability analysis (ALMA). J. Syst. Softw. ***69***, 129–147 (2004)
12. Bohner, S.A.: Software change impacts-an evolving perspective. In: Proceeding of International conference on Software Maintenance, (ICSM 2002), pp. 263–272 (2002)
13. Dreyfus, D., Iyer, B.: Managing architectural emergence: a conceptual model and simulation. Decis. Support Syst. 46, 115–127 (2008)
14. Furukawa, M., Minami, A.: A Study on the "Flexibility" of Information Systems (Part 1): Why Do They Need to Be Flexible? Int. J. Bus. Manag. 8, p48 (2013)
15. Sirmon, D.G., Gove, S., Hitt, M.A.: Resource Management in Dyadic Competitive Rivalry: The Effects of Resource Bundling and Deployment. Acad. Manage. J. 51, 919–935 (2008)

GRL for Representing the Sustainability Reference Model

Birgit Penzenstadler

Department of Computer Engineering and Computer Science
California State University Long Beach
birgit.penzenstadler@csulb.edu

Abstract. [**Context**] Goal modeling is an important method to get a grip on fuzzy concepts, to understand the relations between objectives, and to reason about trade-offs. One of the concepts that is currently discussed with often fuzzy arguments is sustainability. In previous work, we developed a reference model for sustainability in order to be able to break it down into more specific goals and relate those to activities and indicators. [**Problem**] Even though sustainability is now captured and differentiated in a reference goal model, it has no standardized notation that would provide a means for formal reasoning. [**Contribution**] This paper proposes to use GRL to represent our sustainability reference goal model and shows first results. [**Impact**] By using the more wide-spread notation technique based on i* for the sustainability reference model, we hope to facilitate the discussion and application of the model. Integrating the model with a standard technique allows for a broader understanding of how to decompose and handle sustainability as a major objective for systems' development, as well as a formalized basis for reasoning. We hope to engage in community discussion.

1 Introduction

Why consider Software Engineering for Sustainability? How is current SE not sustainable? Current SE practice aims at making systems faster and more encompassing, which can lead to exponential growth in resource consumption. That is unsustainable as we only have limited natural resources available. However, this is driven by the business behind the software and, therefore, the economy. To include sustainability as a concern in the businesses, the added value for the costs caused by sustainability has to be proven, for example improvements of the company image, which is primarily the responsibility of the business analysts. However, as software engineers we are responsible for the long-term consequences of our designs [1].

In previous work, we presented a sustainability goal reference model [10] that helps to break down the abstract concept of sustainability into more tangible goals by means of dimensions and values. It is used as a checklist for software engineers who want to include the goal of sustainability into their development. As the original model didn't have a standard notation and didn't provide means

for formal reasoning, we chose to remodel it in the Goal-oriented Requirement Language (GRL) that allows for reasoning with qualitative and quantitative data. In its new form, the reference model is more accessible (due to a better know notation) and provides more means for guidance and analysis.

2 Background and Related Work

This section explains the background of sustainability and its relation to software engineering, our own preliminary work on a reference goal model for sustainability, and presents an overview of the most relevant related work.

Background: Characterization of Sustainability. The first known European use of the word Nachhaltigkeit ("sustainability") occurred in 1713 for "sustained-yield forestry" [4]. Since then, sustainability has been defined as, inter alia: (i) "The capacity to endure" [13], indicating simply endurance over time. (ii) "Preserving the function of a system over an extended period of time" [6], to serve as a starting point for scoping in systems analysis. (iii) "Ethics expanded in space and time" [7], adding the notion of ethics and values. (iv) "The possibility that all forms of life will flourish forever" [3], adding a notion of prosperity and quality of life that includes non-human forms of life. The first ones ([13] and [6], without the notion of values) are domain-independent, and the latter ones ([7] and [3], the ones that refer to values) are domain-dependent and therefore specifically relevant for the application domain context. To define what sustainability means for any kind of system, exact scoping needs to be performed by answering the questions of what to sustain, for whom, over which time frame, and at what cost [13].

Prior work: Reference Goal Model for Sustainability. To analyze sustainability in detail, we decompose it into five different dimensions [10], see top of Fig. 1. Most concisely, the dimensions of sustainability are characterized as follows: *Individual* sustainability refers to maintaining human capital (e.g., health, education, skills, knowledge, leadership, and access to services). *Social* sustainability aims at preserving the societal communities in their solidarity and services. *Economic* sustainability aims at maintaining capital and added value. *Environmental* sustainability refers to improving human welfare by protecting the natural resources: water, land, air, minerals and ecosystem services. *Technical* sustainability refers to domain-independent longevity of systems and infrastructure and their adequate evolution with changing surrounding conditions. The reference goal model, shown in an excerpt in Fig. 1, structures sustainability by its dimensions, which are represented by a set of values, which can be contributed to by activities and are approximated by indicators (details see [10,11]).

Related Work. There are two works related to sustainability and goal-oriented modeling that are the most important related work for the paper at hand. *Cabot et al.* [2] modeled sustainability goals focusing on conference organization, and

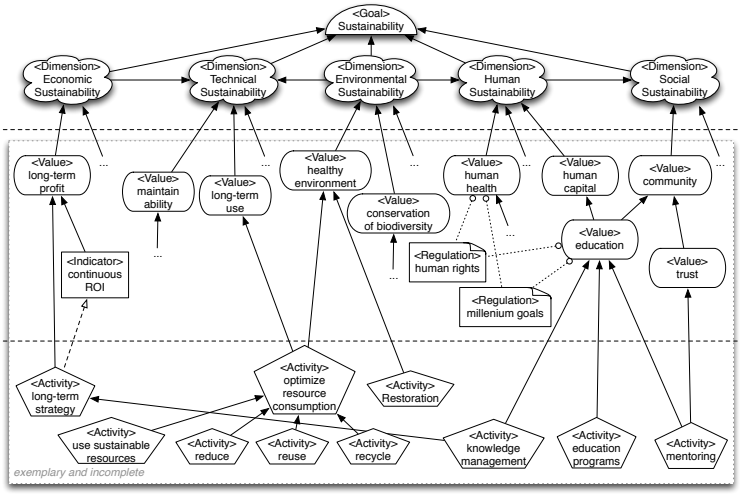


Fig. 1. Old Notation of the Reference Goal Model for Sustainability

their sustainability taxonomy was one of the original inspirations for our sustainability goal reference model [10]. *Mussbacher and Nuttall* [8] modeled sustainability in GRL and related it to time cost, thereby combining quantitative and qualitative data for a richer analysis. Because of that support for indicators and reasoning, we chose to use GRL for our model. Their work is an example case for a sustainability goal model, though, as opposed to a reference model.

3 The Reference Model in GRL

The sustainability reference goal model introduced in Sec. 2 is now represented in its new GRL form in Fig. 2. The elements visible in the figure are the same ones as in the excerpt in Fig. 1 plus a few more activities. While the actual mapping takes place on the level of the meta model, we could not include a figure of the meta model in this paper due to limitations of space, but it is available in [10].

The mapping of the concepts and their visual representation is as follows: Overall goal, Dimensions, and Values are mapped onto Soft goals with metadata for differentiation and preservation of knowledge.

(Sub-)Goals become Goals, Activities become Tasks, and Regulations become Resources.

Indicators remain Indicators, and Contributions remain Contributions.

This now "standardized" representation in a better known notation provides for the possibility to perform richer analyses and reasoning with qualitative and quantitative data.

IV

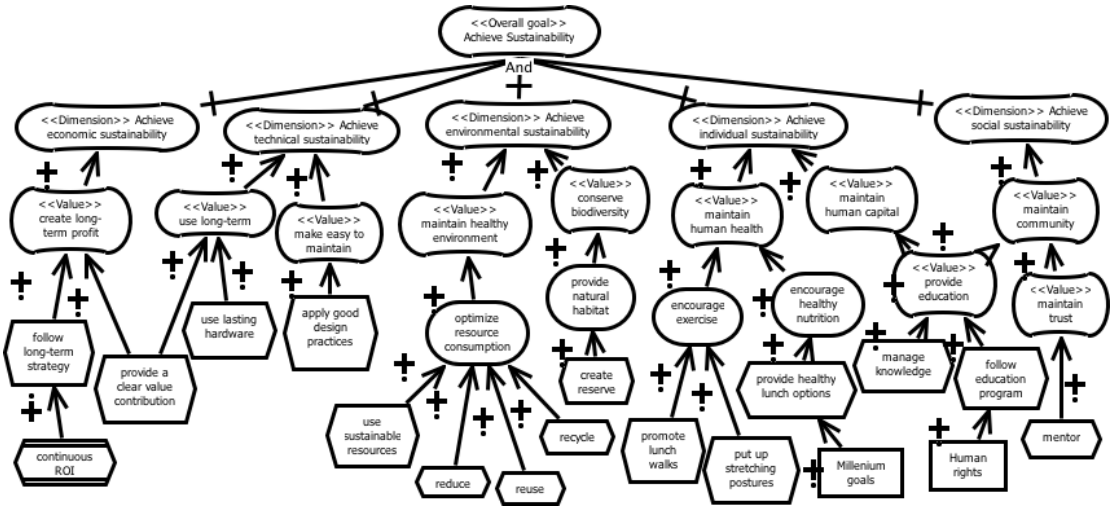


Fig. 2. New GRL Version of the Reference Goal Model for Sustainability.

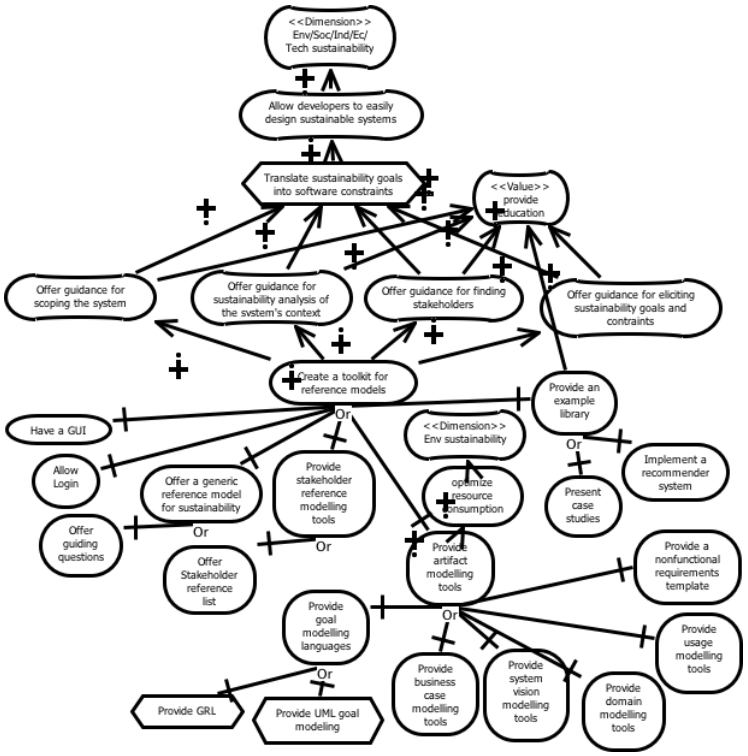


Fig. 3. Goal Model for the Toolkit.

Further Application and Usage. The reference model is intended to serve as a checklist for a systems engineer who tries to integrate sustainability as a major goal into her development context. That means, a goal model is developed while consulting the reference model as one additional input source, because sustainability often lacks explicit stakeholders. The toolkit goal model shown in Fig. 3 is one example where the sustainability goal reference model was considered. It is for a toolkit that helps developers use sustainability reference models (for different artifacts in RE) [9]. When taking a critical look at Fig. 3, there is the *toolkit goal* acting as the only contribution to the four guidance goals. However, this is a known problem in goal modeling. Consequently, the model might be improved in different ways, but it was helpful for progressing with the tool development. The *Values* that the respective *Goals* are related to are referenced in the metadata, which is not visible in the figure. Only for illustrative purposes we have added the example of the $<< Value >>$ *Provide Education* in the upper half and $<< Dimension >>$ *Environmental Sustainability* in the bottom half of the figure.

4 Discussion and Conclusion

Choice of GRL Notation. We use GRL because of its direct support for indicators, which is missing in i*, KAOS, and Tropos, making it difficult to capture real-life measurements from the sustainability assessment in the goal model and then reason about them in the goal model. We could have used BIM [5], which includes *indicators* and *situations* but it is more complex than needed for our purpose.

Expressiveness of New Model. GRL provides a better known notation and simplified the model. Due to this simplification, there initially occurred some information loss during the mapping, for example, that the *dimensions* are now simply *soft goals* as opposed to having a separate concept. However, this was mitigated by using the metadata available in GRL that allowed for adding $<< Dimension >>$ back in.

Analysis and Reasoning. Qualitative and quantitative reasoning are both possible using GRL, and the combination of the two will provide for a richer analysis. That way we are now able to perform a weighting of the goals and by approximating qualitative values to the level of least precision of the quantitative values, we can do first overall assessments.

Future Work on Actors. We chose to show representations without actors in this paper in order to keep things simple for the initial discussion of the sustainability reference model in this notation. One of the steps for future work is to extend the model with views per actor, where the actors are common stakeholders, for example users, project managers, requirements engineers, etc. We plan to extend our previous work on stakeholders for that [12].

Future Work on Tooling. We are currently building a tool (based on jUCMNav) to make this model available as online reference model that can be instan-

tiated and related to other requirements engineering artifacts. For details, see the tool vision in [9].

Take-away Message. This paper presented a sustainability goal reference model in GRL that allows for reasoning with qualitative and quantitative data. It is intended to serve as guidance and checklist for software engineers who want to include the goal of sustainability into their development.

Acknowledgments. We thank Bertrand Ithurburn for his work on the toolkit model, Jennifer Horkoff for a helpful discussion and feedback on a draft version of the model, Gunter Mussbacher for feedback on an earlier version of the model, and Daniel Mendez and Leticia Duboc for feedback on the draft paper. Furthermore, we would like to thank Reviewer 1 for detailed feedback and suggestions for improvement.

References

1. C. Becker, R. Chitchyan, L. Duboc, S. Easterbrook, B. Penzenstadler, N. Seyff, and C. Venters. Sustainability Design and Software: The Karlskrona Manifesto. In *Proceedings of the International Conference on Software Engineering*, 2015.
2. J. Cabot, S. Easterbrook, J. Horkoff, L. Lessard, S. Liaskos, and J. Mazón. Integrating sustainability in decision-making processes: A modelling strategy. In *31st Intl Conf on Software Engineering*, pages 207–210. IEEE, 2009.
3. J. R. Ehrenfeld. The roots of sustainability. *MIT Sloan Management Review*, 46(2):23–25, 2005.
4. R. Heinberg and D. Lerch. What is sustainability? *The post carbon reader: managing the 21st century's sustainability crises*, 2010.
5. J. Horkoff, D. Barone, L. Jiang, E. Yu, D. Amyot, A. Borgida, and J. Mylopoulos. Strategic business modeling: representation and reasoning. *Software & Systems Modeling*, 13(3):1015–1041, 2014.
6. L. Hilty et al. The relevance of information and communication technologies for environmental sustainability. *Env. Mod. & Softw.*, 21(11):1618 – 1629, 2006.
7. S. Mann, K. Costello, M. Lopez, D. Lopez, and N. Smith. An ethical basis for sustainability in the worldviews of first year students. In *ICT for Sustainability 2014 (ICT4S-14)*. Atlantis Press, 2014.
8. G. Mussbacher and D. Nuttall. Goal modeling for sustainability: The case of time. In *Model-Driven Requirements Engineering Workshop (MoDRE), 2014 IEEE 4th International*, pages 7–16. IEEE, 2014.
9. B. Penzenstadler. A toolkit for SE for sustainability - a design fiction. In *Proceedings of the HCI International 2015*, 2015. accepted for publication.
10. B. Penzenstadler and H. Femmer. A Generic Model for Sustainability with Process- and Product-specific Instances. In *First Intl. Workshop on Green In Software Engineering and Green By Software Engineering*, 2013.
11. B. Penzenstadler and H. Femmer. RE at 21: Time to Sustain! In *2nd International Workshop on Requirements Engineering for Sustainable Systems, RE, Rio*, 2013. CEUR-WS.org/Vol-995.
12. B. Penzenstadler, H. Femmer, and D. Richardson. Who Is the Advocate? Stakeholders for Sustainability. In *2nd International Workshop GREENS*, 2013.
13. J. Tainter. A framework for sustainability. *World Futures: The Journal of General Evolution*, 59(3):213–223, 2003.

Making Means-End-Maps Workable for Recommending Teaching Methods

Michael Koch, Dieter Landes

Faculty of Electrical Engineering and Informatics
University of Applied Sciences and Arts
96450 Coburg, Germany
{michael.koch, dieter.landes}@hs-coburg.de

Abstract. Finding appropriate didactical approaches for a specific purpose in software engineering education is difficult. Our work focusses on a recommendation engine for teaching methods. This encompasses modeling teaching goals and suitable teaching methods. To that end, we translated Reich's pool of domain independent constructive teaching methods into a concept map which also includes educational goals or skills at which these methods aim. We started out from Means-End-Maps (ME-Maps), i.e. simple concept maps based on *i** which aim at modeling goals and tasks to achieve these goals. Modeling Reich's pool of methods revealed several shortcomings of ME-Maps. This article presents experiences we made with ME-Maps, discusses necessary changes and extension, and outlines an editor to create such models. Our extension to ME-Maps is expected to significantly improve readability and overview by providing a visual map to quite complex models. Further, such concept maps establish a basis for a goal-oriented search engine for teaching methods in software engineering education.

Keywords. Means-End-Maps (ME-Maps), Concept Maps, Teaching Methods, Educational Goals, Recommender System, *i**

1 Introduction

Nowadays, it is commonly accepted that successful learning requires advanced teaching methods which reach far beyond traditional instructive formats. In particular, a large variety of active learning methods has been developed over the years in pedagogy. Yet, instructors are experts in their particular domain, say in software engineering, but often lack a profound pedagogical background. Therefore, they need support in choosing appropriate didactical methods for a specific purpose. In order to offer a wider variety of teaching methods, thus enhancing interaction in software engineering lessons, useful didactical methods need to be modeled jointly with goals that they may help to achieve and experiences related to their application in a specific setting.

Yet, it is still an open issue which modeling notation is most appropriate for that purpose, striking the balance between clarity and simplicity on the one hand and sufficient expressive power on the other.

This is why we explore in more detail whether Means-End-Maps (ME-Maps) [9] are appropriate for supporting instructors in a manual search for suitable didactical approaches, but also provide a basis for automated reasoning on promising methods.

Our work is primarily directed towards building a goal-oriented search engine which allows instructors to enter their intent and, in return, provides a set of teaching methods ranked by their suitability to meet these goals in a given context. A crucial component of our work consists in establishing a basis for recommendations on didactical approaches in software engineering education [2, 3]. To that end, we started out to model Reich's pool of constructivist teaching methods [6] with ME-Maps. This pool is an extensive collection of teaching methods which are described explicitly and domain-independently and also pays attention to general prerequisites for a meaningful use of a given method, such as minimum and maximum numbers of participants, timeframe etc. Since the method pool is domain-independent, method descriptions neglect technical outcomes in favor of skills that a specific method will foster.

In the following section, we summarize an adaptation of syntax, semantics, and pragmatics of ME-Maps to make them suit our needs. These adaptations are based on experiences made when modeling Reich's pool of constructivist methods with classical ME-Maps. We also briefly highlight some features of a modeling tool for our variant of ME-Maps before a summary and outlook concludes the paper.

2 Modified Means-End-Maps

2.1 Why Using a Concept Map Based Approach in General?

Concept maps in general are intended to capture domain knowledge by describing concepts and their relationships [5] concisely, thus making the notation fairly intuitive. In our particular context, namely software engineering education, we need to consider teaching methods and their contribution to foster competencies in general. For making good decisions, it is also necessary to pay attention to the instance level, e.g. aspects of the instructor's personal attitude and the contents. Association rules are promising candidates to capture this particular aspect.

Therefore it seems to be a good choice to employ a hybrid recommendation algorithm using concept maps for domain knowledge and association rules for context knowledge.

It is worth noting that we explicitly do not want to build a recommendation engine for the one and only "perfect" method in a specific setting, but providing the instructor with a targeted list of promising methods based on matching the primary and secondary goals as well as the context.

Transparent recommendations are more accepted than non-transparent ones [8]. Thus, for better user acceptance, we also want to make the recommendation process comprehensible to the instructor by being able to explain how and why the system generated a recommendation.

2.2 Introducing a Modified ME-Map Approach Based on Our Experiences

We started out by trying to model Reich's pool of constructivist methods with the strict version of ME-Maps and the recommended CmapTools [1] presented in [9]. The original approach is intended to be minimalistic and comes along with only a few language elements based on *i** [10]. There are just two node types: tasks covering the concept of tasks and hard goals from *i**, and qualities covering the concept of soft goals from *i**, yet in a more focused fashion as quality attributes associated to tasks. To express relationships, achieved-by links, consist-of links, association links and contribution (+, -) links are offered. Here we were confronted with some obstacles regarding to our purposes, leading to some syntactic and semantic modifications, which we will describe and explain in the following.

2.2.1 Task and Method Node

Tasks describe actions performed by participants involved in a specific method.

Methods are a special kind of task which represents teaching methods and plays a central role for our purposes. Methods are derived from the task element and are associated with additional attributes for classification and filtering. For highlighting teaching methods and better distinction from "regular" tasks, their label is printed in bold-face type.

2.2.2 Soft Goal and Quality Nodes

Quality nodes in the original ME-Map approach are intended to cover the concept of soft goals from *i**. They express desired quality attributes associated with tasks [9]. In our point of view, however, these concepts are different: for soft goals, methods are a means to achieve the goal while qualities denote constraints on methods, i.e. a second-order concept. Hence we slightly adjust their appearance to emphasize their characters.

Soft Goals mainly represent competencies or intended outcomes [4] fostered by carrying out a given method or performing a given task. In analogy to the concept of misuse cases [7], we want to be able to express outcomes from contradictive teaching approaches that should be explicitly avoided by inverting their color. This explicit syntactic finesse reduces the effort for sentiment analysis significantly.

Qualities in our definition represent quality attributes that a method or task requires in order to be performed meaningfully. To distinguish qualities from soft goals, their label is printed in bold-face type and in italics.

2.2.3 Generalization Links

The *generalizes* link may be used to model more specialized variants of a task, method, or goal. Derived elements inherit all aspects from their parents and allow for the definition of additional aspects in a specialized context. Thus, replicated parts of the model may be avoided and redundancy be reduced.

2.2.4 Containment Links

The *contains* link express either that a task has multiple sub-tasks or that a goal has multiple sub-goals. In contrast to the *consists-of* link described in [9], the containment relationship may be incomplete. This semantic redefinition was necessary in the context of education and distributed modeling, since it might not be useful to model all sub-competencies of a higher competency if these aspects are not relevant for a particular method, but matter for other methods in another (partial) model.

2.2.5 Achievement Links

The *achievedBy* link is semantically identical to the *achieved-by* link from the original ME-Map approach and is intended to be an equivalent of means-end links in *i**. It indicates tasks respectively methods – which are derived from tasks – offering solutions for a parent task. Sibling tasks respectively methods are alternative means to the end represented by the parent task.

2.2.6 Requirement Links

The *requires* link is used to describe required goals needed to achieve another goal or to carry out a task respectively a method. Since this concept is closely related to the concept of association links between tasks/methods and qualities, we replaced association links from the original ME-map approach by requires links. In contrast to association links, the latter link type is also directed to emphasize the roles.

2.2.7 Contribution Links (+, -)

Positive (+) and *negative (-) contribution* links are used to describe the impact of a task, method, or goal on the acquirement of a competency. These links have an outstanding importance for recommending suitable teaching methods. Since the influence of a given method to the achievement of a given competency is hard to express by a quantitative value, it seems more reasonable to use a qualitative value. This contribution can either be positive or negative, in contrast to *i** with its contribution links break, hurt, some-, some+, help, and make. We also decided to not use a qualitative scale like -- or ++ since this might suggest higher precision, yet might cause vagueness if the criteria leading to a rating are not defined or disputable. A fine-grained qualitative scale would also aggravate the occurrence of semantic conflicts when distributed partial models are merged since it is likely that most instructors classify positive and negative aspects similar but weigh them in a different manner. There is also no unknown or neutral contribution intended, since it would have no effect on the generation of recommendations, but would only increase the complexity of the model at the expense of readability.

2.3 An Example of Our Modified Means-End-Map Approach

Figure 1 shows an exemplary (incomplete) model of an in-tray exercise – which is also popular in job application assessment centers – using our modified notation.

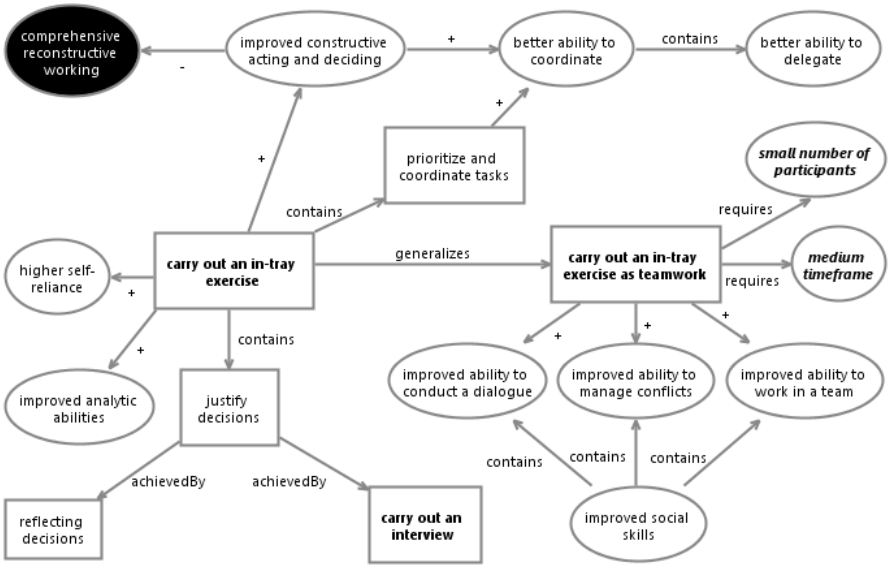


Fig. 1. Modified Means-End-Map for In-Tray Exercises

3 Tool support

The CmapTools recommended for drawing ME-Maps in [9] has many advantages and allows collaborative modeling. Our adjusted approach can be modeled with this tool. However, the CmapTools lack capabilities for custom extensions, such as defining custom attributes. Hence, we developed an Eclipse plugin offering all tools to draw modified ME-Maps, element-based and attribute-based filters as well as some common features such as, e.g., the ability to add hyperlinks or to store a detailed description for each element that will appear on mouse-over events.

Since distributed modeling is important in our setting to create a joint method pool, all elements are tagged with a content-independent globally unique identifier (GUID). Partial models may be merged by creating an alias for multiple GUIDs. This approach allows us to retain the original models, which facilitates correcting wrong associations done by either an algorithm or an instructor, and it can be used to train the merging and recommendation algorithms by implicitly defining synonyms.

4 Summary and Future Work

Modeling didactical methods and the outcomes that they intend to produce is an important prerequisite for supporting instructors in choosing those methods that best suit

their needs. This paper explored whether Means-End-Maps are an appropriate notation for that purpose by modelling Reich's pool of constructivist methods. As it turns out, ME-Maps cannot reasonably be used out-of-the-box since some notational elements do not really fit our needs, both syntactically and semantically. In particular, there is a semantic mismatch between ME-Map's qualities and goals in an educational setting. Therefore, we propose a variation of ME-Maps that seems to be better adapted to the modeling requirements in the educational domain. As future work, this notation will be further explored in the context of an intelligent recommendation system for didactical methods.

Acknowledgements

Our research is supported by the German Ministry of Education and Research (Bundesministerium für Bildung und Forschung) as part of the project EVELIN under grant no. 01PL12022A. For additional information see http://www.evelinprojekt.de.

References

1. Cañas, A. J. et al.: CmapTools: A Knowledge Modeling and Sharing Environment. In: Proc. 1st International Conference on Concept Mapping, Pamplona, Spain (2004)
2. Koch, M., Landes, D.: A Recommender System for Didactical Approaches in Software Engineering Education. In: Proc. DeLFI Workshops 2014 (DeLFI 2014), CEUR Workshop Proceedings Volume 1227, pp. 140-143, Freiburg, Germany (2014)
3. Koch, M., Landes, D.: Design and Implementation of a Competency Repository. In: Rocha, A. et al. (eds.): New Perspectives in Information Systems and Technologies, Volume 1, pp. 249-255, Springer, Heidelberg, Germany (2014)
4. Koch, M., Landes, D.: Notations for Modeling Educational Goal Profiles. In: Proc. 1st European Conference of Software Engineering Education (ECSEE 2014), pp. 45-58, Shaker, Aachen, Germany (2014)
5. Novak, J.D., Cañas, A. J.: The Theory Underlying Concept Maps and How to Construct and Use Them, Technical Report IHMC CmapTools (2008) – available at: http://cmap.ihmc.us/docs/theory-of-concept-maps (last visited on: 2015-06-10)
6. Reich, K.: Konstruktivistische Didaktik. 4th ed., Beltz Verlag, Weinheim, Germany (2008) – The method pool is available in German at: http://methodenpool.uni-koeln.de (last visited on: 2015-06-10)
7. Sindre, G., Opdahl, A. L.: Capturing Security Requirements through Misuse Cases. In: Proc. Norsk Informatikkonferanse (NIK'2001), pp. 219-230, Tromsø, Norway (2001)
8. Swearingen, K., Rashmi, S.: The Role of Transparency in Recommender Systems. In (Terveen, L. G. et al. Eds.): Extended abstracts of the 2002 Conference on Human Factors in Computing Systems (CHI'02), pp. 830-831, ACM press, New York (2002)
9. Wang, J., Sturm, A., Yu, E.: Know-How Mapping: From *i** to ME-maps. In: Proc. 7th International *i** Workshop (istar'14), CEUR Workshop Proceedings Volume 1157, Thessaloniki, Greece (2014)
10. Yu, E.: Modeling Strategic Relationships for Process Reengineering. PhD thesis, Department of Computer Science, University of Toronto (1995)

Designing adaptive systems

João Pimentel, Jaelson Castro

Centro de Informática
Universidade Federal de Pernambuco
Recife, Brazil
{jhcp, jbc}@cin.ufpe.br

Abstract. In this work, we investigate the interplay between requirements and architecture in the context of adaptive systems. Furthermore, we propose the Multi-Level Adaptation for Software Systems (MULAS) framework. It is centred on the iterative and incremental refinement of a goal model, towards the creation of a design goal model, which can be used at runtime to drive adaptation on a system that is properly instrumented. Moreover, the framework includes a tool-supported process for generating statechart behavioural models from a design goal model.

Keywords: Requirements-driven software adaptation, architecture-driven software adaptation, goal-oriented requirements models, model-driven development.

1 Introduction

Different approaches to support the development of self-adaptive systems have been proposed in the literature. However, those are often restricted to a single aspect of software development. For instance, the Zanshin framework [1] provides support for handling adaptation at the requirements level, enacting a monitoring-diagnosis-compensation cycle. With Zanshin, adaptation is specified in terms of stakeholders' goals, tasks, quality constraints, and other elements.

On the other hand, Rainbow [2] provides similar capabilities, but addressing architectural models. Thus, it is concerned with properties of systems' components and connectors, e.g., response time, number of servers and load balancing. The differences between requirements-based and architecture-based approaches are discussed in [3].

Requirements engineering and architectural design, while addressing the system specification at different abstraction levels, comprise intertwined activities [4]. The former focuses on the problem at hand, whereas the latter provides solutions for that problem.

Approaches that only support requirements-based or architecture-based adaptation thus, lack relevant elements of the adaptation space. For instance, architecture-based

approaches might ignore stakeholders' goals and preferences, while requirements-based ones may not address concerns related to the system implementation, such as algorithms and components.

Hence, the investigation of how to support seamless adaptation mechanisms across the different phases of software development seems to be a promising venue to improve the development of self-adaptive software systems. In this paper we provide an overview of a design process centred on an extended goal model, which incorporate elements aiming to support requirements-based and architectural-based adaptation. To illustrate, we adopt a meeting scheduler exemplar.

The remainder of this paper is organized as follows. In Section 2, we present our process to design adaptive systems, focusing on behavioural specification. Section 3 discusses the limitations of this work. Later, we present ongoing and future work in Section 4.

2 Design process

The proposed process, which is a part of the Multi-Level Adaptation for Software Systems (MULAS) framework, comprises eight steps (Fig. 1). The first five steps are related to the refinement of design goal models: *Identify design tasks, constraints and assumptions*; *Assign tasks*; *Define basic flows*; *Identify indicators, parameters and relations*; and *Specify adaptation strategies*. The other three steps are related to statecharts: *Generate base statechart*; *Specify transitions*; and *Include adaptation elements*. While these steps may be followed mostly sequentially, waterfall-like, in realistic settings it is expected that the architect will go back and forth, by introducing additional refinements to already refined elements.

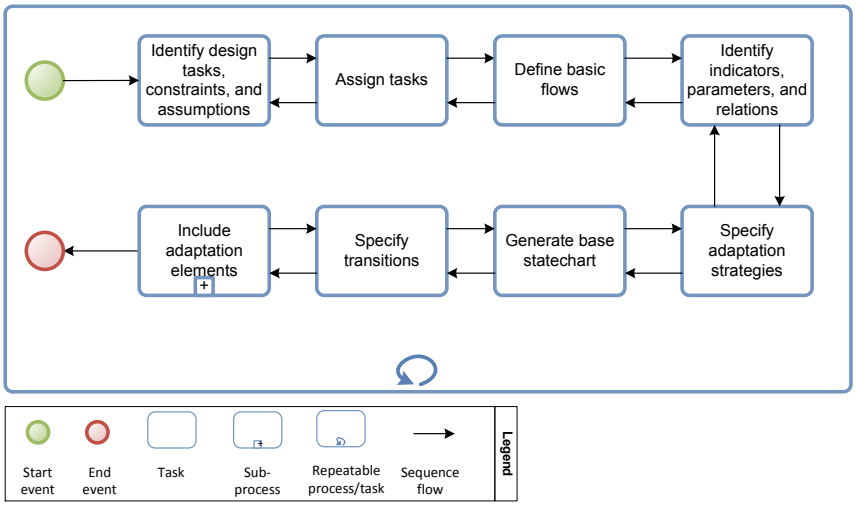


Fig. 1.MULAS design process

The first step, *Identify design tasks, constraints and assumptions*, supports the refinement of a goal model by including elements that are not initially required by stakeholders, but are relevant from the architectural point of view, expressed as design tasks, design constraints and design assumptions. The second step, *Assign tasks*, consists of assigning the responsibilities for the execution of tasks — e.g., tasks that will be performed by an external actor (human or otherwise). This assignment is helpful for defining the scope of the system.

In the next step, *Define basic flows*, the architect introduces possible flows for every sub-tree in the goal model. Roughly, these flows describe the order that the sub-elements are going to be fulfilled or executed, so that their parent element can be considered fulfilled or executed. These flows are expressed as alternative flow expressions, introduced as annotations to a goal model using a top-down, bottom-up, or middle-out strategy. These expressions are later used to automatically generate a statechart that represents the system's behaviour.

The next two steps are related to the adaptation capabilities of the system: *Identify indicators, parameters and relations* and *Specify adaptation strategies*. The former is related to the addition, in the design goal model, of some elements proposed by Zashin [1], in light of the design elements previously included in the first step. In the *Specify adaptation strategies* step it is considered how the system will react to failures — e.g., by retrying the execution of a task, or by changing the parameters described in the goal model.

The second part of the process is related to system behaviour. The first step, *Generate base statechart*, makes use of derivation patterns to automatically create a statechart from the flow expressions previously defined. Although flow expressions are a useful intermediate abstraction between goal models and statecharts, they are not as expressive as statecharts. Thus, in the next step, *Specify transitions*, the transitions of the statechart are refined with their events and conditions, which are identified by analyzing when any given transition should take place.

An example of a resulting (Design) Goal Model is shown in Fig. 2, which is an excerpt from a Meeting Scheduler system [10]. Besides the scheduling itself, the system supports the characterization of meetings, the gathering of timetables and the management of meetings, while satisfying the non-functional requirements of scalability and portability. The excerpt on Fig. 2 depicts the sub-tree of the *Define Schedule* goal, which is refined with the *Schedule Manually* and *Schedule Automatically* tasks. Both tasks must be supported by the system, thus it is an AND-refinement. The automatic scheduling can only be performed if the *Rooms Available* assumption is satisfied, since the system is not able to book additional rooms. Moreover, the *Schedule Automatically* task is refined with design elements – elements that result from design decisions, i.e., they are not mandated by customers or users.

The tasks defined during architectural design (the so called design tasks) in this example are: *Brute Force Algorithm*, *Heuristics-based Algorithm*, and *Select Date*. The first two tasks define algorithms that can be executed to perform the scheduling, while *Select Date* is a task that must be performed once the algorithms find a set of possible dates. Additionally, the automatic scheduling presents two design constraints: it must be performed in less than ten minutes and it must be implemented with web-services. In particular, the selected web-service must be available at least 90 % of the time.

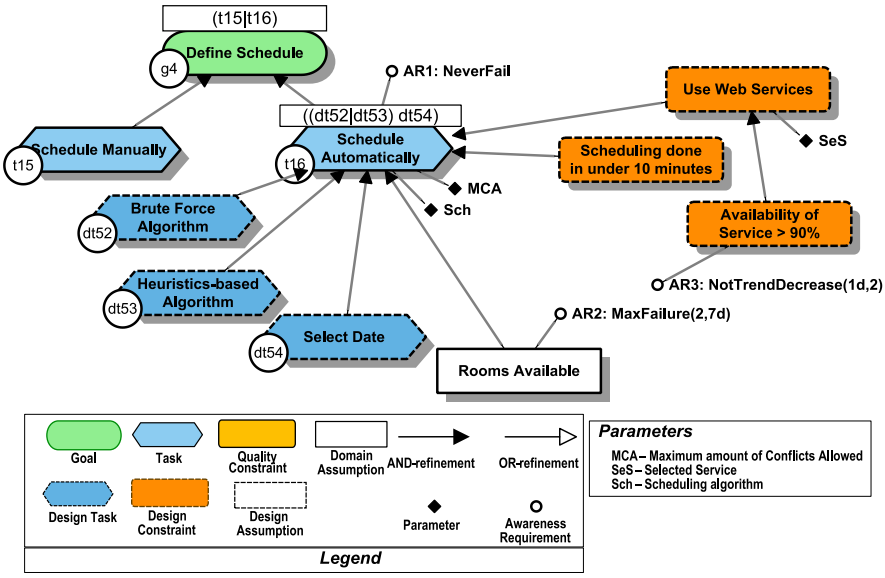


Fig. 2.Excerpt of the design goal model of the Meeting Scheduler system.

Besides goals, tasks, constraints and assumptions, the DGM also contains *flow expressions*, an extension of regular expressions that allow the definition of the execution flow of the system. Six constructs can be used in these expressions: alternative (vertical bar, |), optional (question mark, **?**), sequence (blank space,), repetition (star or plus symbol, * or +), parallelism (hyphen,–), and idle states (*iX*, where *X* is a natural number).

Each flow expression defines the behaviour for the element which it is on top of. For instance in Fig. 2, the *(t15|t16)* expression states that, when the system needs to *Define Schedule*, it may either perform *Schedule Manually (t15)* or *Schedule Automatically (t16)*. Moreover, for the execution of *Schedule Automatically*, the expression is *((dt52|dt53) dt54),* with this meaning: after performing either *Brute Force Algorithm (dt52)* or *Heuristics-based Algorithm (dt53)*, the system will perform the *Select Date* task *(dt54)*.

Lastly, the DGM defines what must be monitored during the system execution (the so called *awareness requirements*), and what can be modified in the system (the *parameters*). In our example, the AR1 awareness requirement, linked to the *Schedule Automatically* task, states that it must never fail. AR2, linked to the *Rooms Available* assumption, indicates that it should be false no more than twice a week (*Max Failure 2, 7d*). On the other hand, AR3 linked to the *Availability of Service* design constraint, defines that its success rate should not decrease for two days in a row (*NotTrendDecrease 1d, 2*).

As illustrated with the aforementioned example, the design goal model allows the integration of requirements and architectural concerns in a single model. Both requirements and architecture elements can be used to specify the system adaptation, with awareness requirements, parameters, relations, and adaptation strategies. In the next subsection we discuss some of the limitations of the proposed process and the design goal model.

3 Limitations

This MULAS design process, as well as the design goal model, presents a series of limitations, regarding the following aspects: expressiveness of the design goal model, heuristics for selecting optimal flows, tool support, and compositional adaptation.

Expressiveness of the design goal model – The design goal model proposed in this paper is based on goal model extension [1]. That extension includes awareness requirements and parameters, which are relevant as they correspond to the control theory concepts of *reference value* and *control input*. However, as a result of the focus on these control theory concepts, two other important concepts have been partially neglected: contribution links and context. The explicit use of contribution links and context annotations may improve the expressiveness of the design goal model. Nonetheless, it is necessary to balance this expressivity with the complexity of the proposed model.

Heuristics for selecting optimal flows – In the MULAS framework, we propose the use of flow expressions to define the possible flows of the system. However, we do not provide any guidance that helps the architect in the decision of which flow may be best in different contexts and scenarios. Further investigation is required in order to identify heuristics, patterns, or techniques to facilitate such decision.

Tool support – A supporting tool was developed specifically to support the MULAS framework. Even though this tool is functional, more effort is required in order to make the tool suitable for public use, related not only to actual development but also to the creation of user documentation, such as user guides or tutorials.

Compositional adaptation – Parameterized adaptation is adaptation related to the modification of variables. In contrast, compositional adaptation is related to modifying structural parts of the system. While we have conducted early endeavours on the latter [6][7] during this research, the MULAS framework is focused only on the former.

4 Ongoing and Future Work

This is an ongoing work, with early results presented in [9][10]. Its most recent results composed a doctoral thesis [8] which includes: detailed description of the MULAS framework; description of a support tool; case studies; experiments. We were able to use this framework for developing information systems, which were verified by means of simulation. Moreover, a mobile differential drive robot was designed and developed using the MULAS framework, providing satisfactory results.

Through an experiment with 15 requirements engineering students, we were able to obtain evidence in favour of the feasibility of the framework. Nonetheless, further experimentation is required in order to properly evaluate and evolve the proposal, specifically in the context of large industrial system.

Another interesting line of research is to adapt the MULAS framework for developing context-sensitive systems [11]. As future work, we intend to investigate the integration of a control theoretic approach (Zanshin) with a context-based one, aiming to expand the expressiveness of the proposal.

Acknowledgments

This work has been supported by the ERC advanced grant 267856 “Lucretius: Foundations for Software Evolution”, as well as by the following Brazilian institutions: FACEPE, CAPES and CNPq.

References

1. Souza, V.E.S.: Requirements-based software system adaptation, Ph.D. Thesis (2012).
2. Garlan, D., Cheng, S.-W., Huang, A.-C., Schmerl, B., Steenkiste, P.: Rainbow: architecture-based self-adaptation with reusable infrastructure. Computer, 37, 2004, pp. 46–54.
3. Angelopoulos, K., Souza, V.E.S., Pimentel, J.: Requirements and Architectural Approaches to Adaptive Software Systems: A Comparative Study. 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, 2013, pp. 23–32.
4. Castro, J., Lucena, M., Silva, C., Alencar, F., Santos, E., Pimentel, J. Changing attitudes towards the generation of architectural models. Journal of Systems and Software, 85, 2012, pp. 463–479.
5. Yu, Y., Mylopoulos, J., Lapouchnian, A., Liaskos, S., Leite, J.C.S.P. From Stakeholder Goals to High-Variability Software Design. Technical report csrg-509, University of Toronto, 2005.
6. Pimentel. J., Lucena, M., Castro, J., Silva, C., Santos, E., Alencar, F. Deriving software architectural models from requirements models for adaptive systems: the STREAM-A approach. Requirements Engineering Journal, 17-4, 2012, pp.259–281.
7. Dermeval, D., Soares, M., Alencar, F., Santos, E., Pimentel, J., Castro, J., Lucena, M., Silva, C., Souza, C.: Towards an i*-based Architecture Derivation Approach. Fifth International i* Workshop, 2011, pp. 66-71.
8. Pimentel, J.: Systematic Design of Adaptive Systems — A Control-Based Framework, Ph.D. thesis (2015). Available at http://www.cin.ufpe.br/~ler/supplement/istar2015/
9. Pimentel, J.; Angelopoulos, K.; Souza, V. E. S.; Mylopoulos, J.; Castro J. From Requirements to Architectures for Better Adaptive Software Systems. 6th International i* Workshop, 2013, pp. 91-96.
10. Pimentel, J. et al. From requirements to statecharts via design refinement. 29th Symposium on Applied Computing, 2014, pp. 995–1000.
11. Vilela, J.; Castro, J.; Pimentel; J.; Soares, M.; Lima, P.; Lucena, M. Deriving the behavior of context-sensitive systems from contextual goal models. 30th Symposium on Applied Computing, 2015.

Formalization of the i* Mapping Rules for Class Diagram

Josenildo Melo[1], Aêda Sousa[1], Celso Agra[1], Fernanda Alencar[2]

[1]Departamento de Engenharia de Computação, Universidade de Pernambuco, Recife, Brazil
{jasm,amcs,clasf}@ecomp.poli.br
[2]Departamento de Eletrônica e Sistemas, Universidade Federal de Pernambuco, Recife, Brazil
fernandaalenc@gmail.com

Abstract. The phase of requirements gathering of a project is extremely essential because it identifies all the features that the project should have. After this phase, they must be modeled to be better understood. To model solutions, UML (Unified Modeling Language) is one of the most used languages, but it is not developed to capture domain requirements for quality. To capture these requirements, models based on Goal-Oriented Requirements Engineering (GORE) are used, such as i* (iStar). This paper presents a formalization of i* mapping rules for class diagram in the context of Model-Driven Development (MDD), aiming to create more complete class diagram, where quality requirements are captured.

Keywords: Transformation between models, i*, class diagram, Model-Driven Development.

1 Introduction

Companies need to respond quickly to new market demands, building new solutions or performing maintenance on existing systems. So must update your processes and working properly, without neglecting the quality requirements [1]. It is necessary that the end of the requirements specification phase, all stakeholders is acutely aware of the features and system behavior. For this, are proposed and used various models, especially models of Unified Modeling Language (UML).

The UML is efficient to specify "what" a system does and "how" it does something, but it is not to describe the "why" it does [2]. It is not designed to capture the domain requirements (early requirements) [3]. To minimize these problems, came the Goal-Oriented Requirements Engineering (GORE) [4]. In goal-oriented approaches, requirements engineering is responsible for discovering, formulating and analyzing the problem to be solved, as well as conclude because the problem must be solved and who is responsible for solving the problem. [5]. The need to have more precise specifications of requirements that they consider the reasons, motivations and intentions captured by GORE approach led to the initial proposal of models mapping rules i* (goal-oriented) for class diagrams [6] in UML, which subsequently been extended [7].

This time, contributing to a possible automatic transformation between models could be thought. The formalization of transformation rules between these models was initialized in [8] and making it necessary to formalize and test all the rules, to allow automatic transformations between models (i* to class diagrams).

This paper aims to demonstrate a transformation between models in the context of Model Driven Development (MDD) [9], which is obtained through the formalization of mapping rules described above. For this, the present article is organized as follows: Section 2 briefly define the objectives of the research; Section 3 we discuss the scientific contributions; Section 4 provides the conclusions, and Section 5 presents ongoing and future works.

2 Objectives of the research

This work aims to demonstrate a transformation between models in the context of MDD. This will be achieved through the formalization of the guidelines proposed by [6] and extended by [7]. This guidelines was created to map i* into UML class diagram. The objective of this transformation is to keep the consistency between the desired software system and the organization objectives, as well to establish the impact that any change of objectives will be able to cause in the system and vice versa.

3 Scientific contributions

Using templates to design complex systems is standard in traditional engineering disciplines. We cannot imagine the construction of a building, a bridge or a car, without first constructing a variety of designs and simulate them. Models help us understand a complex problem (and possible solutions) through abstraction.

Currently, the Model-Driven Development (MDD) [9] has proved to be a highly reputable trend [10]. In fact, MDD aims to accelerate the development of software by automating the development of products and employing reusable models or abstractions to view the code (or the problem domain). By using the models, or abstractions, we can describe complex concepts more legibly than computer languages do. This improves communication between stakeholders, because models are often easier to understand than the code [11].

The most important contribution of this work is the development of a transformation between models that covers the MDD. This transformation will be responsible for creating the most complete class diagrams, which cover better user requirements.

This transformation will be achieved through the formalization of the guidelines proposed by [6] and extended by [7]. This guidelines are shown in the Table 1. Is not part of the scope of this study to discuss these rules, but the formalization of them.

Table 1. Mapping Guidelines

Number	i*	UML
1.1	Agents, roles or position	Class.

Number	i*	UML
1.2	Relationship ISPART-OF between positions, agents or roles.	Class aggregation.
1.3	Relationship ISA between positions, agents or roles.	Class generalization/specialization.
1.4	Relationship OCCUPIES between an agent and a position.	Class association named OCCUPIES.
1.5	Relationship COVERS between a position and a role.	Class association named COVERS.
1.6	Relationship PLAYS between an agent and a role.	Class association named PLAYS.
2.1	Tasks defined in SD model.	Methods with public visibility.
2.2	Tasks defined in SR model.	Methods with private visibility.
3.1	Resources defined in SD model.	Class if this dependence has the characteristics of an object.
3.1	Resources defined in SD model.	Attribute with private visibility in class that represents the dependee actor if this dependence cannot be characterized as an object
3.2	Resources (sub resources) defined in SR model.	Attribute with private visibility in the class that represents the actor in which the sub resource belongs (if this sub resource cannot be understood as an object).
3.2	Resources (sub resources) defined in SR model.	An independent class, otherwise.
4.1	(Soft)Goals in SD model.	Attribute with public visibility in the class that represents the dependee.
4.2	(Soft)Goals in SR model.	Attribute with visibility public in the class that represents the actor in wich the sub goal belongs.
5	Task Decomposition.	Represented by pre and posconditions (expressed in OCL) of the corresponding pUML operation.
6.1	(Soft)Goals-(Soft)Goals.	The disjunction of the means values implies the end value.
6.2	(Soft)Goal – Task, Resource-Task.	The post-condition of the means task implies the value of end.
6.3	Task – Task.	The disjunction of the post-condition of the means imply the pos-conditions of the end.

The formalization process is shown in Figure 1. The process starts with data input obtained by iStarTool [12] tool. In iStarTool, the i* element is designed and the tool generates a corresponding file XMI. In the second step ("transformation between models"), this XMI file is imported and rules described in ATL language [13] are applied. In the last step, an output model is generated containing the elements of class

diagram generated. This output model is another XMI file, that must be imported by a CASE tool for the class diagram can be viewed.

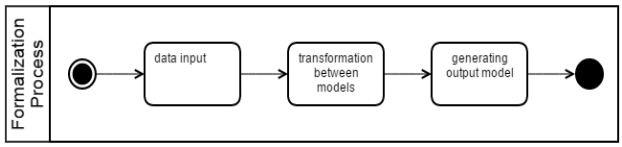


Fig. 1. Formalization process.

4 Conclusions

Requirements elicitation is essential for a system to be developed with all the features and functionality needed, and the application templates can help us visualize the system before its construction begins. Several models exist, the UML is more used. However, UML does not capture all system requirements, indicating "as" a system should be done and not "why" should be done. Among the approaches that care about the needs of the system, stands out i*.

This work presented the formalization of rules mapping i * to class diagram in order to create class diagrams that addressed user requirements more fully.

5 Ongoing and the future work

The objective of this work is to create more complete class diagrams using the MDD features, covering the features i* and based on previously created rules. For this, these rules are being formalized in the ATL language. Furthermore, we are performing a comparison between the ATL language and others three transformation languages (QVT, ETL and MOFScript).

As future work, it is planned to finalize the formalization of the rules, as well as the creation of a tool to automate the whole process. Also is planned the adaptation of XGOOD tool [14]. This tool decides which i* elements must be mapped.

References

1. T. C. Pereira, F. M. R. Alencar, J. R. F. Silva, and J. F. B. Castro, “Requisitos Não-Funcionais em Modelos de Processos de Negócio : Uma Revisão Sistemática,” *Proc. do IX Simpósio Bras. Sist. Informação*, 2013.
2. G. A. A. C. Filho, A. Zisman, and G. Spanoudakis, “A Traceability Approach for i * and UML Models,” in *2nd International Workshop on Software Engineering for Large-Scale Multi-Agent Systems*, 2003.
3. F. M. R. Alencar, F. Pedroza, J. F. B. Castro, and R. C. O. Amorim, “New Mechanisms for the Integration of Organizational Requirements and Object Oriented Modeling,” in *VI WORKSHOP DE ENGENHARIA DE REQUISITOS*, 2003.

4. A. van Lamsweerde, “Goal-oriented requirements engineering: a guided tour,” in *Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on*, 2001, pp. 249–262.
5. J. Pimentel, M. Lucena, J. Castro, C. Silva, E. Santos, and F. Alencar, “Deriving software architectural models from requirements models for adaptive systems: the STREAM-A approach,” *Requir. Eng.*, vol. 17, no. 4, pp. 259–281, Jun. 2011.
6. F. M. R. de Alencar, “Mapeando a Modelagem Organizacional em Especificações Precisas,” UFPE, 1999.
7. J. F. Castro, John Mylopoulos, F. M. R. Alencar, and G. A. C. Filho, “Integrating Organizational Requirements and Object Oriented Modeling,” in *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering*, 2001, p. 146–.
8. U. D. E. A. Oliveira, “Uma Metodologia DSDM para Integração de Requisitos Organizacionais e UML no Desenvolvimento de Sistemas de Agentes Utilizando i* (i-Star),” Universidade Federal de Sergipe, 2009.
9. B. Selic, “The pragmatics of model-driven development,” *IEEE Software*, vol. 20, no. 5, pp. 19–25, Sep-2003.
10. J. M. Vara, V. A. Bollati, Á. Jiménez, and E. Marcos, “Dealing with Traceability in the MDD of Model Transformations,” vol. 40, no. 6, pp. 555–583, 2014.
11. E. Yu, “Why Agent-Oriented Requirements Engineering,” in *4th International Workshop on Requirements Engineering: Foundations of Software Quality*, 1998, pp. 15–22.
12. Á. Malta, M. Soares, E. Santos, J. Paes, F. Alencar, and J. Castro, “iStarTool: Modeling requirements using the i; framework,” *CEUR Workshop Proc.*, vol. 766, no. iStar, pp. 163–165, 2011.
13. F. Jouault, F. Allilaire, J. Bézivin, and I. Kurtev, “ATL: A model transformation tool,” *Sci. Comput. Program.*, vol. 72, no. 1–2, pp. 31–39, Jun. 2008.
14. F. Pedroza, F. Alencar, J. Castro, F. R. C. Silva, and V. F. A. Santander, “Ferramentas para Suporte do Mapeamento da Modelagem i * para a UML : eXtended GOOD – XGOOD e GOOSE,” *Proc. 7th Work. Requir. Eng. - WER 2004*, pp. 164–175, 2004.

US2StarTool: Generating i* Models from User Stories

Renato Mesquita[1], Aline Jaqueira[1], Celso Agra[2], Márcia Lucena[1], and Fernanda Alencar[23]

[1]Departamento de Informática e Matemática Aplicada – UFRN
{rmsnatal,alineopj}@gmail.com; marciaj@dimap.ufrn.br
[2]Programa de Pós-Graduação em Engenharia da Computação – UPE
[3]Departamento de Eletrônica e Sistemas - UFPE
celsoagra@gmail.com; fernanda.ralencar@ufpe.br

Abstract. In agile methods, the requirements are represented by user stories. However, this model does not allow a good visualization of context in which a story is inserted, reducing the understanding of the system as a whole. On the other hand, the i* model presents dependencies among organizational actors, and the understanding of the context in which a requirement is inserted. This paper presents an implementation of an automated solution as a tool to mapping user stories into i* models, US2StarTool, adding a support for agile development environment. US2StarTool can help requirements engineer in agile development environments contextualizing the environment in which user stories are inserted and showing relationships between the actors and the system.

Keywords: User Story, i* Model, Meta-model, Model Mapping.

1 Introduction

To build a software project, it must be specified the requirements to be satisfied. The requirements elicitation aims to specify the software completeness and correctness, besides guarantee the quality, validation and acceptance. In agile methods, the requirements are developed of incremental way, according to stakeholders demands. The artifacts used are user stories. To represent a user story, Cohn [2] suggests a common format: As <role>, I want <action>, to <goal>. However, in user stories you cannot visualize dependencies among stories [3], besides it is hard to assimilate the context that they are included within a system [8].

The i* modeling technique [9] is one of the most relevant Goal-Oriented Requirements Engineering (GORE) approaches and provides a view of involved actors and their dependencies. Thus, the requirement model described from i* models provides a complete representation of requirements. The dependencies between actors are represented and by this model it is possible understand the context.

The benefits of using visual models to describe the requirements are presented in [1]. Thus, this paper presents the needed stages to develop a tool named US2StarTool, able to map user stories into i* model based on a set of mapping heuristics. In the Section 2 we define the research objectives. Then Section 3 explains how

the tool has been developed. In Section 4 we discuss the scientific contributions. Section 5 provides the conclusions and Section 6 presents future works.

2 Objectives of the research

This paper presents an automated solution, named US2StarTool, that the main objective is mapping user stories into i* models. The results are used as a complementary tool for agile development process, providing a better understanding for context in the projects based on user stories, dependencies between actors involved and the whole system-to-be.

3 Development and required tools

To implement the US2StarTool, we used the EuGENia tool, a small part of the Epsilon Framework (Extensible Platform for Specification of Integrated Languages for Model Management). We consider that EuGENia tool is used as code generation, models transformation, validation, comparison and refactoring with EMF (Eclipse Modeling Framework) and other types of models. The EMF is a modeling framework and code generation to build tools based on a structured data model. An EMF model is named as Ecore and defines a meta-model language that can be implemented using EMFatic. EMFatic is a language used to represent EMF Ecore models in a textual form and has the *.emf extension. The EuGENia tool works as a front-end for GMF (Graphical Modeling Framework) and facilitates the Ecore models handling.

Thus, it was possible to generate the editors in Java class for the EMF meta-models that represents user story and i* model. Then the object-oriented structures were used based on the transformation between the models to perform mapping and implementation of the heuristics. The packages *.us2star.us and *.us2star.istar have interfaces and enumeration classes. The packages *.us2star.us.impl and *us2star.istar.impl have the implementation classes related to interfaces.

To implement the tool, we separate in a set of stages. In the first stage, we develop the user story metamodel using the Emfatic language. In addition, we used the meta-model of the i* model based on Paes [7]. From the .emf files were generated .ecore extension files. With the .ecore files and EuGENia tool a structure that represents the meta-models of both user story and i* are generated. The US2StarTool is an executable file based on java environment, named "US2StarTool.jar", and doesn't require any installation process.

The US2StarTool requires to the user an input file, in XLS format, that has been imported by an external tool responsible for managing user stories. If the user does not have this file, it will have to access this tool in order to get it and will upload it in the US2StarTool that will map to imported user stories. Then the US2StarTool generates an output file in XMI format capable to be imported by an i* model editor, so that it generates the graphical representation of this model.

The *.reader.xls package is responsible for obtaining the input file with *.xls* extension, and interpret it in order to read the user stories. Then, the package

*mapping.us can communicate to obtain user stories and create objects from OO structure, that represents the user story model. This structure is composed by packages in *.us. After build the user stories objects, the mapping process is performed, using a set of heuristics. Thus, the package *.mapping.istar communicates with the packages *.mapping.istar.command, that contains the implemented heuristics, and *.mapping.us that contains the objects from user stories, and perform the mapping, creating OO objects structures that represents the i* model, composed by packages in *.istar. Finally, the package *.writer.xmi communicates with *.mapping.istar, which contains objects to create elements of output file in the *.xmi format.

The set of mapping heuristics were implemented based on the proposed heuristics from Jaqueira [5]. The user stories are imported into US2StarTool, resulting in the creation of objects represented by the elements in the model of user stories. These models are stored by UsData class, localized in the package *.mapping.us. Therefore, the heuristics are able to map elements that represents the i* model and keep in memory by the IstarData class, localized in the package *.mapping.istar. The mapping between models are performed through the UsData2IstarData class that reads the data in UsData and safe the objects in the IstarData. The heuristics are presented at Table 1 according to the sequence in which they are executed during mapping. Were used the concepts and notations according to i * Wiki [4] that is a simplified version of the technique.

Table 1. Proposed mapping heuristics from Jaqueira [5].

ID	Description
SD-H1	Create the System Actor;
SD-H2	Create an actor in i* model for each different role of user stories;
SD-H3	Create a goal in i* model for each goal of user stories. If there are repeated goals they will be defined only once in the model;
SD-H4	If there are repeated goals for different actors, create a generic actor;
SD-H4.1	Create a IS_A relationship of the generic actor for other specific actors who share the same goal;
SD-H5	Relate the dependencies of each actor with his goals;
SR-H1	Create a task in the Actor System for each action of user stories;
SR-H2	If there are different actions for the same goal, create a generic task;
SR-H2.1	Decompose the generic task into subtasks that represent the actions associated with the same goal;
SR-H3	Relate the dependencies of each goal with the corresponding tasks according to user stories;
SR-H4	If there are tasks that depend on itself actor to which they are related, generated a resource to the task name;
SR-H5	Relate the resource created, depending on the actor.

To use the US2StarTool, it's needed that user has the user stories to upload in the application. We perform an integration of US2StarTool with a tool based in the management of backlogs, named Easybacklog (EB). The EB tool is useful for agile development and customer teams, to manage user stories. Using this tool it's possible

export an excel file (*.xls), that contains the data related to the backlog registered in the EB tool, including the user stories, that will be mapped by US2StarTool. Then, the *.xls file is imported by US2StarTool.

In this work, we use the IstarTool as graphical editor to generate and export the i* model [7]. This tool is able to create i* diagrams, validating the syntax rules based on the official rules defined by [4]. To integrate the US2StarTool with Istar-Tool, it was generated a XMI file with the *.istar extension. The figures 1 and 2 presents the activities involved in the process using the US2StarTool.

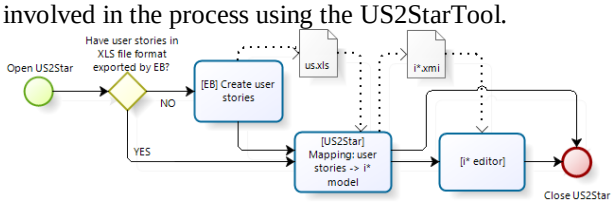


Fig. 1. External process to upload user stories to be mapped by US2StarTool.

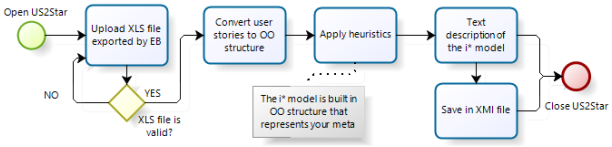
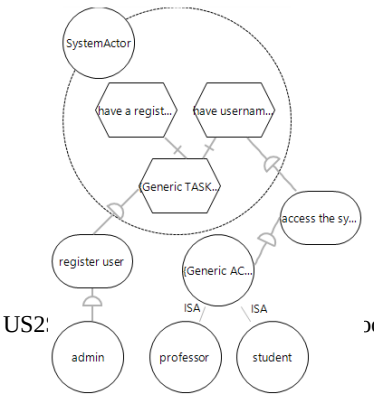


Fig. 2. Showing the process of use of this Tool.

To demonstrate the operation of the US2StarTool, we used four user stories of a login system presented in Table 2 as an example. The i* model generated by US2StarTool is shown in Figure 3.

Table 2. Part of User Stories of a login system.

US01	**As** professor **I want** have username and password **To** access the system	**US03**	**As** admin **I want** have a registration request **To** register a user
US02	**As** student **I want** have username and password **To** access the system	**US04**	**As** admin **I want** have username and password **To** register a user



US2 ool.

4 Scientific contributions

Visual models assist in understanding of how users will need to use the system and are effective for the stakeholders to understand the proposed solution and also to keep them interested and involved [1]. Even in an agile

environment it is necessary to develop some models before any implementation to ensure a shared understanding by the development team. Sharp et al. [8] concluded that the user stories are limited artifacts to provide understanding of the system as a whole and their dependency relationships are omitted.

The most important contribution of this work is the implementation of the tool named US2StarTool that maps a set of requirements described by user stories to a graphical model using the i* technique.

Therefore, the tool can assist the requirements engineering in the agile methodology context, contributing to a better view of actors and their dependencies in the system; better knowledge about actor roles inside the system-to-be; understand how actors can achieve their goals and actions; prioritize actors context and their dependencies in the system-to-be; define related functionalities and assign them to the same team.

5 Conclusions

The implementation of a tool to get i* models from user stories was successfully performed. The construction of US2StarTool enabled the study of requirements representation models in agile methods and i* technique. Were also studied the technology for creation of metamodels in Emfatic language and the use of EuGENia tool, both to generate the corresponding structure in Java to create metamodels, and for the creation of the graphic editor of the i* model. Heuristics mapping have been implemented and integrated with the Easybacklog tool to assist in creating the user stories, and finally, generate an input file for the graphics editor IstarTool. An example of application using real data (available at www.dimap.ufrn.br/~marciaj/US2StarTool) allows us to test the tool according to the analysis of the results suggesting its usefulness and contributions in agile development environment.

6 Ongoing and future work

A limitation of the tool is that is not possible to display the Resource element in the i* model generated, because as from a model of user stories is not possible to obtain this information to be mapped. One solution would be user interaction with US2StarTool to inform the dependencies between tasks and actors. The user can enter this information in the file .xmi exported by US2StarTool, or in the i* model editor, the IstarTool. Another limitation is the fact that it was not possible to generate a graphical representation of the i* model in itself US2StarTool. This occurred because the i* editor uses GMF technology, ie, is dependent on the Eclipse platform.

In order to continue the research for this work a few suggestions of further work can be cited. Improve the mapping process in order to display the resources in the i* model. User interaction with the US2StarTool after file import containing user stories is required. Thus, the user can tell whether action of the user stories depends on a specific feature of an actor. Integrate the tool with a graphical editor i* without the need to export a file and use another tool for this purpose. The i* model would be

shown in the own US2StarTool. This is not possible because the graphics generators used are dependent on the Eclipse platform, since they are based on GMF. View the process of action of heuristics step-by-step at the time of mapping, in order to facilitate the user who wants to learn about the mapping process. Show the result of mapping for SD and SR models separately. The tool displays the SR model of the i*, which is an evolution of the SD model with the expansion of Actor System. Developing the reverse mapping, or mapping model for user i* stories. Write a manual for using the tool.

References

1. Beatty, J. e Chen, A. Visual Models for Software Requirements. Washington, Microsoft Press: 2012.
2. Cohn, M.: User Stories Applied: For Agile Software Development (The Addison-Wesley Signature Series), March. Addison-Wesley Professional, Reading (2004).
3. Gomez, A., Rueda, G., & Alarcón, P. P. (2010). A Systematic and Lightweight Method to Identify Dependencies between User Stories. In A. Sillitti, A. Martin, X. Wang, & E. Whitworth (Eds.), Agile Processes in Software Engineering and Extreme Programming (Vol. 48, pp. 190-195): Springer Berlin Heidelberg.
4. http://istar.rwth-aachen.de/tiki-index.php?page=i*+Wiki+Home. Acessado em: Maio de 2015.
5. Jaqueira, Aline; Lucena, Márcia; Aranha, Eduardo; Alencar, Fernanda and Castro, Jaelson. Using i * Models to Enrich User Stories. Proceedings of the 6th International i* Workshop (iStar 2013), CEUR Vol-978. 2013.
6. Kolp, Manuel, Paolo Giorgini, and John Mylopoulos. "A goal-based organizational perspective on multi-agent architectures." Intelligent Agents VIII. Springer Berlin Heidelberg, 2002. 128-140.
7. Paes, J. AGILE: Uma Abordagem para Geração Automática de Linguagens i*. Dissertação de Mestrado em Ciência da Computação. Universidade Federal de Pernambuco, 2011.
8. Sharp, H., Robinson, H. Segal, J. and Furniss, D. (2006) 'The Role of Story Cards and the Wall in XP teams: a distributed cognition perspective', Proceedings of Agile 2006, IEEE Computer Society Press, pp 65-75.
9. Yu, E. "Modelling Strategic Relationships for Process Reengineering". PhD thesis.University of Toronto, Department of Computer Science. 1995.

Specifying guidelines to transform i* Model into User Stories: an overview

Celso Agra[1], Aeda Sousa[1], Josenildo Melo[1], Márcia Lucena[2], Fernanda Alencar[3]

[1]Programa de Pós-Graduação em Engenharia da Computação – UPE
{clasf, amcs, jasm}@ecomp.poli.br
[2]Departamento de Informática e Matemática Aplicada – UFRN
marcia.lucena@gmail.com
[3]Departamento de Eletrônica e Sistemas - UFPE
fernandaalenc@gmail.com

Abstract. Requirements engineering is a process of obtaining and detailing the goals of systems-to-be. Therefore, different approaches can improve the requirements elicitation phase, such as diagrams and documentations. Scenarios based on GORE approach, such as i* framework, model the early phase of requirements stage. The framework, however, can become a large and unreadable diagram. On the other hand, user stories are clear, concise, and can represent the late-phase of requirements. Therefore, this paper presents an overview of a proposal to transform i* models into user stories, as a way to facilitate the management of requirements in the agile projects.

Keywords. i* Framework, Goal-Oriented Requirements Engineering, Scenario transformation, User stories, Agile projects

1 Introduction

Requirements engineering is a process of obtaining and detailing the goals of systems-to-be [1]. A set of different solutions were created to improve the requirements elicitation phase, such as diagrams and documentations artifacts. They are scenario techniques based on Goal-Oriented Requirements Engineering (GORE) modeling which are being widely studied by academics and practitioners. These techniques define goals to stakeholders clearly and easily, and thus, they are becoming useful for Requirements Engineering [4].

According to [2], the main challenges found in Requirements Engineering are related to lack of communication and poor quality of artifacts. Communication problems during the requirements elicitation phase can compromise the schedule or even the budget of a project. The use of scenarios, such as GORE models and other artifacts requirements, may contribute with the elicitation and documentation requirements, and also with the building of better quality software.

Scenarios based on GORE approach, such as i* framework, model the early phase of the requirements stage, and consider organizational contexts, intentions and ration-

ales for stakeholders demands [5]. On the other hand, user stories are simple, clear and concise artifacts used to represent and detail requirements, and are, thus, valuable to users [6]. In addition, user stories describe functionalities in the late requirements phase, during the software lifecycle, considering "how" those requirements should be developed.

A set of proposals were made by combining i* model with other late-phase approaches and subsequent phases of software lifecycle [5]. However, none of them presented an approach to transform i* model into user stories, to represent an evolutionary approach for agile projects. Therefore, this paper presents an overview of a proposal to transform i* models into user stories, as an effective approach to improve the requirement engineering stages, in special, the management of requirements.

2 Objectives of the research

The main purpose of this research is to transform i* models into user stories, to facilitate the agile projects through the management of requirements. The management of requirements involves a set of activities such as documenting, modeling, prioritizing and tracking requirements, and also versioning and managing requirements changes [7]. Thus, managing requirements changes and specifying requirements, are a ways of improving and enriching the project documentation, adding, removing or editing software requirements, and relating them among the GORE approach and agile projects.

According to this proposal, requirements can be structured and prepared to be used by agile projects. Thus, the transformation proposal may assist in creating scenarios that are relevant to stakeholders and development teams. The use of an approach to assist in structuring requirements can be an alternative. However, this approach should result in consistent documents with a comprehensive overview of the proposed system, from a high level of requirements abstraction with i* to the specific functionalities of user stories.

3 Scientific contributions

The use of i* models in software projects: (i) contributes to the understanding of stakeholders intentions, and the "why" of a proposed requirements [8]; (ii) provides adequate representation of alternatives, and offers primitive modeling concepts such as softgoal and goal.[9]; (iii) is a complementary way from conventional requirements models that focuses on behaviors and interactions between systems and their environments [10].

On the other hand, user stories: (i) describe functionalities that will be valuable to either a user or purchaser of a system or software [11]; (ii) bring a clear, concise and objective description in natural language; (iii) are used to specify details and help to build tests.

Thus, the found benefits of both approaches can be useful to build scenarios during the elicitation process and requirements specification.

We argue the approach may be characterized as an evolutionary contribution where the business analysts can model the early phase of requirements stage. This result in a diagram that will become a user story to assist stakeholders in understanding their needs and in building or improving their own user stories based on results obtained with this approach.

3.1 The transformation proposal

The proposal is based in two approaches: (i) convert Strategic Dependency model (SD) in a temporary Strategic Rationale model (SR), to become understandable for the subsequent stage; (ii) isolate elements and linkages according to dependencies analysis and generate a set of user stories, represented as story cards.

We apply the semantic rules based on SR Model into SD Model to analyze the behavior of these linkages. This approach was based on Cross-Impact analysis [10], to understand how elements can be affected. Therefore, this analysis assists in understanding some semantic linkages from the i* model.

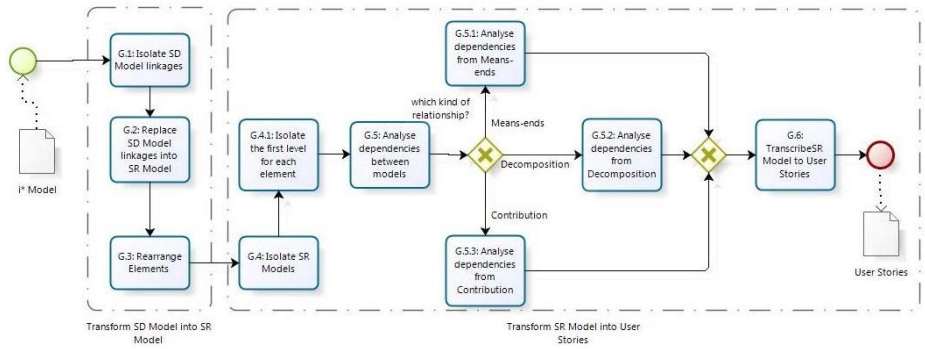


Fig. 1. Process flow using the BPMN notation.

Figure 1 describes the whole proposal process through the BPMN notation. In the first case, the transformation from SD model to SR model uses a set the guidelines applied in the i* model to perform the first phase of the process:

- G.1: Isolate the relations into small parts characterized as a requirement specification. In this case, it is necessary to analyze each dependency and their internal relations for each actor. This structure will be named as a requirement path.
- G.2: Replace the SD model linkages into SR model linkages as shown in figure 2.
- G.3: Rearrange the new SR Model vertically to become understandable when a case begins and how the case will affect the other elements.

After transforming SD model into SR model, we state that the user story should follow a set of guidelines as a way to identify dependencies:

- G.4: Isolate the SR models with their relations giving a level of relationship, i.e. only the first level of relationship among elements. The subsequent levels are not included in this user story.

- G.4.1: Consider as a one level relationship: Means-Ends, for "means" elements and their "ends"; Decomposition, for verified element and their sub-elements; Contribution, for the contributor and the contributed.
- G.5: Analyze dependencies between models, i.e., if an analyzed element is related to another isolated SR model.
- G.5.1: In a Means-Ends case, the element considered as “Ends” creates a dependency with the related “Means” element;
- G.5.2: For each Decomposition case, the element that is being decomposed creates a dependency with the element which is a part of its decomposition;
- G.5.3: For the Contribution case, the softgoal element which is contributed creates an dependency with its contributor;
- G.6: Each isolated model should be transcribed as a user story considering its dependencies among another user stories.

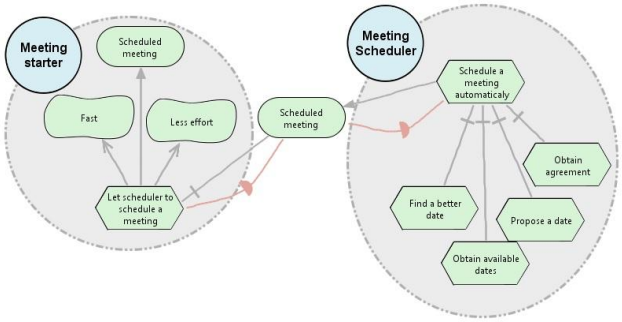


Fig. 2. Requirement path by example from [3], applying G.1 and G.2.

To illustrate the final result, table 1 shows both user stories generated after applying these guidelines.

Table 1. User Stories results

User Story 01	User Story 02
The “meeting scheduler” must “schedule a meeting automatically”, to obtain a “scheduled meeting”.	The “meeting starter” must “let scheduler to schedule a meeting” to obtain a “scheduled meeting”.
Acceptance Criteria: • Must “find a better date”; • Must “propose a date”; • Must “obtain agreement” • Must “obtain available dates”	Acceptance Criteria: • Must obtain “scheduled meeting” by “meeting scheduler” (dependency with User Story 01) • Must achieve “fast” (treatment) • Must achieve “less effort”

4 Related works

The GORE-based transformation scenarios techniques are being used for researchers and practitioners, and some related topics have been discussed in this paper. We con-

sidered papers that use GORE modeling to be transformed into different artifacts, such as BPMN, Use Case and MDD. For each work we analyzed the problem, the solution and the result.

[12] use an approach to obtain Business Process Modeling Notation (BPMN) models from i* models and vice versa. Modeling business processes by using a goal oriented approach, such as i* framework, helps in aligning the business process improvement towards the satisfaction of the organization's strategic goals, that was not covered by BPMN. Studies about GORE modeling and BPMN were found as follows: [13] used transforming rules for i* Goal Models into Business Process Models; [14] develop an approach based on KAOS goal models to create BPMN models; [15] create a relation between BPMN 2.0 and KAOS.

[9] propose a set of guidelines to provide a mapping and transformation of Use Cases from i* models. The authors states that there's a lack of proper understanding of the organization by the software developers, and the UML is ill equipped for organizational requirement modeling, that can be represented using i* framework. This same approach can be found as follows: [5], proposes a methodology to support a co-evolution from approaches focusing on i* model, and Formal Use Cases in UML notation, that can be used in a complementary manner with requirements engineering; [16] used a KAOS approach to create diagrams in UML, in special class Diagrams.

[17] proposes a set of guidelines to generate an OO-Method (Object-Oriented) conceptual model from i* model, aiming to improve the quality of the models used on the development of information systems, and consequently to obtain better products. According to these researches, the GORE modeling transformation techniques are not common in projects based on agile methods. This lack was explored by [6] who propose a set of heuristics to perform a mapping of the requirements, presented as user stories and transformed into i* models, as a way to reduce the limitation of software context and dependencies between user stories in agile methods.

5 Conclusion, Ongoing and future work

Combining Goal-Oriented Requirements Engineering (GORE) modeling with other late-phase artifacts is becoming widely studied. This approach will contribute to reduce the gap between early-phase and late-phase as well as facilitate the management of requirements in the agile projects, by covering activities, such as documentation and elicitation. In this paper we present an overview of how to transform i* Models into user stories as an understandable document for business analysts and stakeholders and also some of the benefits related to the transformation approach of scenarios. This will help the development team to gather more information during the Requirement Engineering phase.

As future work, we will perform a research in the semantic and syntax field to propose a consistent structure, and create a friendly representation of a software requirement specification for i* models. Also, some challenges can be solved with this approach and they'll be analyzed in the future. In addition a DSL is being created to be used in a computer-based solution, as a way to transcribe i* Models into a list of requirements specification. These artifacts are planned to be used with Agile Method-

ology documents, in special User Stories. The expected impact for those issues may reflect on the academic and professional environment, since the i* model may fit in agile methods that use elicitation process for the requirements.

References

1. Robert A. Elliott, Sr., and Edward B. Allen. "A methodology for creating an IEEE standard 830-1998 software requirements specification document". In Journal of Computing Sciences in Colleges, Pages 123-131, Volume 29 Issue 2, December 2013.
2. Wiegers, K. and Beatty, J., Software Requirements. Redmond. Microsoft. 3 Ed. 2013
3. Santander, V., Integrando Modelagem Organizacional com Modelagem Funcional [Thesis]. Recife (PE). Federal University of Pernambuco. 2002.
4. Sen, A.M. and Jain, S.K., An Agile Technique for Agent Based Goal Refinement to Elicit Soft Goals in Goal Oriented Requirements Engineering. International Conference on Advanced Computing and Communications, 2007. ADCOM 2007.
5. Bhuiyan, M.M.R.; Zahidul Islam, M.M.; Krishna, A. and Ghose, A., Co-evolution of Agent Oriented Conceptual Models and Use Case Diagrams. 6th International Conference on Quality Software, QSIC 2006. 2006.
6. Jaqueira, A., Lucena, M., Aranha, E., Alencar, F. and Castro, J., Using i * Models to Enrich User Stories. Proceedings of the 6th International i* Workshop (iStar 2013). 2013
7. Pohl, K. and Rupp, C., Requirements Engineering Fundamentals. Brasil. T&M Teste de Software LTDA. 2012.
8. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F. and Mylopoulos, J., Tropos: An Agent-Oriented Software Development Methodology. Autonomous Agents and Multi-Agent Systems, v. 8, n. 3, p. 203-236. 2004.
9. Santander, V. and Castro, J., Deriving use cases from organizational modeling. Proceedings of IEEE Joint International Conference on Requirements Engineering, 2002.
10. Yu, E., Modelling Strategic Relationships for Process Reengineering, PhD Thesis, University of Toronto, Toronto, Canada (1995)
11. Cohn, M. and Beck, K., User Stories Applied: For Agile Software Development. Addison-Wesley Professional. 2004
12. Alves, R., Silva, C. and Castro, J. A bi-directional mapping between i* and BPMN models in the context of business process management. In: Requirements Engineering @ Brazil
13. Decreus, K., Snoeck, M. and Poels, G. Practical Challenges for Methods Transforming i* Goal Models into Business Process Models. Requirements Engineering Conference, 2009. RE '09. 17th IEEE International
14. Horita, H., Honda, K., Sei, Y., Nakagawa, H., Tahara, Y. and Ohsuga, A., Transformation approach from KAOS goal models to BPMN models using refinement patterns. In Proceedings SAC '14 Proceedings of the 29th Annual ACM Symposium on Applied Computing pp. 1023-1024. 2014.
15. Cortes-Cornax, M., Matei, A., Letier, E., Dupuy-Chessa, S. and Rieu, D., Intentional Fragments: Bridging the Gap between Organizational and Intentional Levels in Business Processes. Proceedings of Confederated International Conferences: CoopIS, DOA-SVI, and ODBASE 2012, Rome, Italy, September 10-14, 2012.
16. Chanvilai, S., Honda, K., Nakagawa, H., Tahara, Y. and Ohsuga, A., Goal-oriented approach to creating class diagrams with OCL constraints., in Sascha Ossowski & Paola Lecca, ed., 'SAC' , ACM, , pp. 1051-1056. 2012.
17. Alencar, F., Marín, B., Giachetti, G., Pastor, O., Castro, J. and Pimentel, J. H., From i* Requirements Models to Conceptual Models of a Model Driven Development Process. Procedings in Second IFIP WG 8.1 Working Conference, PoEM 2009, Stockholm, Sweden, November 18-19, 2009.