# iStar 2011 – Proceedings of the 5th International *i** Workshop

**29-30th August, 2011**

**Trento, Italy**

**http://www.cin.ufpe.br/~istar11/**


Jaelson Castro
Xavier Franch
John Mylopoulos
Eric Yu (eds.)

**a CEUR Workshop Proceedings, ISSN 1613-0073**

# Index

**Preface**

**Keynote talk.** Governing Sociotechnical Systems. *Munindar P. Singh*, p. 1

## Scientific Papers

**Session 1. Framework Evaluation**

1. Eight Deadly Sins of GRL. *Gunter Mussbacher, Daniel Amyot, Patrick Heymans*, p. 2-7
2. iStarML: Principles and Implications. *Carlos Cares*, *Xavier Franch*, p. 8-13
3. Empirical Evaluation of Tropos4AS Modelling. *Mirko Morandini, Anna Perini, Alessandro Marchetto*, p. 14-19

**Session 2. Model Analysis and Evaluation**

4. Detecting Judgment Inconsistencies to Encourage Model Iteration in Interactive *i** Analysis. *Jennifer Horkoff, Eric Yu*, p. 20-25
5. Towards a Declarative, Constraint-Oriented Semantics with a Generic Evaluation Algorithm for GRL. *Hao Luo, Daniel Amyot*, p. 26-31
6. A Flexible Approach for Validating i* Models. *Ralf Laue, Arian Storch*, p. 32-36

**Session 3a. Ontologies and Foundations**

7. Ontological Analysis of Means-End Links. *Xavier Franch, Renata Guizzardi, Giancarlo Guizzardi, Lidia López*, p. 37-42
8. Supporting i* Model Integration through an Ontology-based Approach. *Karen Najera, Anna Perini, Alicia Martínez, Hugo Estrada*, p. 43-48
9. The Mysteries of Goal Decomposition. *Scott Munro, Sotirios Liaskos, Jorge Aranda*, p. 49-54

**Session 3b. Enterprise and Software Architectures**

10. Development of Agent-Driven Systems: from i* Architectural Models to Intentional Agents Code. *Maurício Serrano, Julio Cesar Sampaio do Prado Leite*, p. 55-60
11. Towards Adoption and Use of i* to Support Enterprise Architecture in Organizational Settings: a Position Paper. *Daniel Gross, Uwe Zdun, Eric Yu*, p. 61-65
12. Towards an i*-based Architecture Derivation Approach. *Diego Dermeval, Monique Soares, Fernanda Alencar, Emanuel Santos, João Pimentel, Jaelson Castro, Márcia Lucena, Carla Silva, Cleice Souza*, p. 66-71

**Session 4a. Application to Domains**

13. Nòmos: from Strategic Dependencies to Obligations. *Silvia Ingolfo, John Mylopoulos, Anna Perini, Alberto Siena, Angelo Susi*, p. 72-77
14. Technology Representation in i* Modules. *Eliel Morales, Xavier Franch, Alicia Martínez, Hugo Estrada, Oscar Pastor*, p. 78-83
15. An extension of i* to Model CSCW Requirements Applied to a Collaborative Conference Review System. *Miguel A. Teruel, Elena Navarro, Víctor López-Jaquero, Francisco Montero, Pascual González*, p. 84-89
16. Modeling Strategy Maps and Balanced Scorecards using i*. *Constantinos Giannoulis, Jelena Zdravkovic*, p. 90-95

**Session 4b. Variability, Product Lines, and SOA**

17. Capturing Contextual Variability in i* Models. *Alexei Lapouchnian, John Mylopoulos*, p. 96-101
18. Security Requirements Engineering for Service-Oriented Applications. *Fabiano Dalpiaz, Elda Paja, Paolo Giorgini*, p. 102-107
19. Goals and Scenarios for Requirements Engineering of Software Product Lines. *Gabriela Guedes, Carla Silva, Jaelson Castro*, p. 108-113

**Session 5a. User-Centered Design and Simulation**

20. Bridging User-Centered Design and Requirements Engineering with GRL and Persona Cases. *Shamal Faily*, p. 114-119
21. Issues and Challenges in Coupling Tropos with User-Centred Design. *Luca Sabatucci, Chiara Leonardi, Angelo Susi, Massimo Zancanaro*, p. 120-125
22. Causal vs. Effectual Behavior - Support for Entrepreneurs. *Jan Schlüter, Dominik Schmitz, Malte Brettel, Matthias Jarke, Ralf Klamma*, p. 126-131

**Session 5b. Sociality and Measurement**

23. A Social Interaction Based Pre-Traceability for i* Models. *Maurício Serrano, Julio Cesar Sampaio do Prado Leite*, p. 132-137
24. Requirements Engineering for Social Applications. *Amit K. Chopra, Paolo Giorgini*, p. 138-143
25. Intentional Models based on Measurement Theory. *Andre Rifaut*, p. 144-149

## Tool Fair Demos

**Preface**, p. 150

26. I*-Prefer: A Tool for Preference Model-Driven Decision Making. *Tianying Li, Tong Li, Haihua Xie, Lin Liu*, p. 151-153
27. OpenOME: An Open-source Goal and Agent-Oriented Model Drawing and Analysis Tool. *Jennifer Horkoff, Yijun Yu, Eric Yu*, p. 154-156
28. Implementing Structural Measures over i* Diagrams. *Daniel Colomer, Xavier Franch*, p. 157-159
29. GRL Modeling and Analysis with jUCMNav. *Daniel Amyot, Gunter Mussbacher, Sepideh Ghanavati, Jason Kealey*, p. 160-162
30. iStarTool: Modeling Requirements using the i* Framework. *Átlia Malta, Monique Soares, Emanuel Santos, Josias Paes, Fernanda Alencar, Jaelson Castro*, p. 163-165
31. Tool Interoperability using iStarML. *Carlos Cares, Xavier Franch, Daniel Colomer, Lidia López*, p. 166-168
32. Adding Functionality to openOME for Everyone. *Ralf Laue, Arian Storch*, p. 169-171
33. Tropos Modeling, Code Generation and Testing with the Taom4E Tool. *Mirko Morandini, Cu Duy Nguyen, Loris Penserini, Anna Perini, Angelo Susi*, p. 172-174
34. Analysing a Repository of Design Knowledge with the GO-DKL Browser. *Andrew Hilts, Eric Yu*, p. 175-177
35. i* Modules: a jUCMNav Implementation. *Daniel Colomer, Xavier Franch*, p. 178-180

# Preface

The iStar workshop series is dedicated to the discussion of concepts, methods, techniques, tools, and applications associated with *i** and related frameworks and approaches. Following successful workshops in Trento, Italy (2001), London, England (2005), Recife, Brazil (2008), and Hammamet, Tunisia (2010), the 5th International *i** Workshop is being held back again in Trento. As with previous editions, the objective of the workshop is to provide a unique opportunity for exchanging ideas, comparing notes, and forging new collaborations.

This year, the workshop is co-located with the 19th IEEE International Requirements Engineering Conference (RE 2011), benefiting from the common interests shared by the workshop and the conference. As with past editions, we have tried to keep the format small and informal so as to maximizing interaction. However, the increasing number of submissions and presented papers have forced for the first time in the workshop to organize some parallel sessions. We preferred to adopt this option, rather than to reduce the time allocated to papers or be more restrictive in the acceptance process. We have included in the wrap-up session a 5-minute summary of each parallel session given by the session chairs. Also in terms of format, the 30 minutes allocated to each paper have been split equally into presentation and discussion.

Concerning the review process, each of the 29 submitted papers went through a thorough review process with three reviews from a programme committee, providing useful feedback for authors. Revised versions of the 25 accepted papers are included in these proceedings. We thank authors and reviewers for their valuable contributions.

The program was completed with a keynote given by Prof. Munindar P. Singh entitled "Governing Sociotechnical Systems" and a tools fair in which 10 tools were presented in a plenary session - the first such event in the workshop series.

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

We look forward to lively conversations and debates with old and new friends at the workshop, in the wonderful surroundings of Trento, Italy!

Jaelson Castro, *Universidade Federal de Pernambuco, Brazil*
Xavier Franch, *Universitat Politècnica de Catalunya, Spain*
John Mylopoulos, *University of Trento, Italy*
Eric Yu, *University of Toronto, Canada*

# Programme Commitee

Fernanda Alencar, *Universidade Federal de Pernambuco, Brazil*.
Daniel Amyot, *University of Ottawa, Canada.*
Carlos Cares, *Universidad de la Frontera, Chile.*
Luiz Marcio Cysneiros, *York University, Canada*.
Fabiano Dalpiaz, *University of Trento, Italy*.
Hugo Estrada, *CENIDET, Mexico*.
Paolo Giorgini, *University of Trento, Italy*.
Daniel Gross, *University of Toronto, Canada*.
Renata S.S. Guizzardi, *Universidade Federal do Espírito Santo, Brazil*.
Jennifer Horkoff, *University of Toronto, Canada*.
Dimitris Karagiannis, *Universitaet Wien, Austria*.
Alexei Lapouchnian, *University of Toronto, Canada*.
Julio Cesar Sampaio do Prado Leite, *Pontificia Univ. Catolica do Rio de Janeiro, Brazil*.
Sotirios Liaskos, *York University, Canada*.
Lin Liu, *Tsinghua University, China.*
James Lockerbie, *City University London. UK*.
Lidia López, *Universitat Politècnica de Catalunya, Spain*.
Oscar Pastor, *Universidad Politécnica de Valencia, Spain*.
Michael Petit, *University of Namur, Belgium*.
André Rifaut, *Centre de Recherche Public Henri Tudor*, Luexembourg.
Pete Sawyer, *Lancaster University, UK*.
Dominik Schmitz, *RWTH Aachen University, Germany*.
Juan Carlos Trujillo, *Universidad de Alicante, Spain*.
Yves Wautelet, *Hogeschool-Universiteit Brussel, Belgium*.
Jelena Zdravkovic, *Stockholm University, Sweden*.

# Submissions Management

Carla Silva, *Universidade Federal da Paraiba, Brazil*.

# Tools Fair Organizer

Jennifer Horkoff, *University of Toronto, Canada*.

# Tools Fair Committee

Jennifer Horkoff, *University of Toronto, Canada*.
Xavier Franch, *Universitat Politècnica de Catalunya, Spain*.

# Webmaster

Diego Dermeval Medeiros da Cunha Matos, *Universidade Federal de Pernambuco, Brazil*.

# Proceedings Edition Support

Sergi Franch, *Col·legi Kostka, Spain*.

# Governing Sociotechnical Systems

Munindar P. Singh[1]

[1] North Carolina State University
http://www.csc.ncsu.edu/faculty/mpsingh/

**Abstract.** From social networks to business processes to collaborative science, a hallmark of modern applications of information technology is that they involve interactions between largely autonomous and heterogeneous participants. I define sociotechnical systems as those that involve both real-world "social" relationships between the participants and technological elements. Traditional computer science concentrates on the technological elements and has little to offer in the way of abstractions and techniques for sociotechnical systems, leaving practitioners with no option but to resort to ad hoc approaches.

I motivate governance as the proper notion of administration in such settings. Governance recognizes the inherent peer-to-peer nature of sociotechnical systems and their need to support flexible interactions among the participants. In this manner, it contrasts with the Twentieth Century notion of management of subordinates by superiors.

Governance offers a great opportunity for multiagent systems practice and research. I show how longstanding multiagent themes of organizations, institutions, and norms come together in governance, and also how we need to go beyond existing multiagent approaches in modeling and applying these themes.

I draw my examples from the Ocean Observatories Initiative, where this notion of governance is being applied.

# Eight Deadly Sins of GRL

Gunter Mussbacher[1], Daniel Amyot[2], and Patrick Heymans[3]

[1]Dept. of Systems and Computer Engineering, Carleton University, Canada
gunter@sce.carleton.ca
[2]School of Electrical Engineering and Computer Science, University of Ottawa, Canada
damyot@eecs.uottawa.ca
[3]PReCISE Research Centre, University of Namur, Belgium
phe@info.fundp.ac.be

**Abstract.** Goal modeling languages are now an integral part of requirements engineering. They allow for the systematic capture of rationales for stakeholder needs and enable the reasoning about potential system solutions. The goal-oriented Requirement Language (GRL) is both a typical goal modeling language and an international standard, yet it suffers from many shortcomings, akin to deadly sins, that relate to its concrete syntax, semantics, approach to modularity, analysis, and extensibility. Based on 10 years of experience using GRL, we discuss several shortcomings and point to relevant future work areas.

**Keywords.** Goal Modeling, GRL, Syntax, Semantics, Modularity, Analysis.

## 1 Introduction

Goal modeling forms an important part of requirements engineering. Over the last decade, through experience developing, standardizing, and using the Goal-oriented Requirement Language (GRL) [9], we have observed several important issues that we consider to be akin to "deadly sins". We also suspect that many of these sins could be generalized to other goal modeling languages such as i*, KAOS, and Tropos, hence their general interest to the goal modeling community. In the following section, we briefly set out our research objectives, and in Section 3, we discuss the identified issues. Section 4 provides our conclusion and reports in a few words on future work.

## 2 Research Objectives

We aim to improve the state of the art of goal modeling by first identifying shortcomings in one of the mainstream goal modeling languages (GRL), then analyzing these shortcomings systematically, investigating ideas on how to address them, and finally providing solutions for these shortcomings that can be incorporated into the GRL standard and perhaps into other goal modeling languages. Here, we report on the currently identified eight major shortcomings, analogous to deadly sins (the seven common ones plus Despair) of GRL language designers and tool builders (including us).

## 3 Scientific Contributions

The list of eight sins of goal modeling languages in this section starts with issues related to the syntax of goal modeling languages (Section 3.1), then continues with semantic issues (Section 3.2), and concludes with issues regarding modularity (Sections 3.3, 3.4, and 3.5), analysis (Sections 3.6 and 3.7), and extensibility (Section 3.8). Whenever applicable, dependencies between issues are stated.

### 3.1 Pride: Lack of Support for Multiple Concrete Syntaxes

Goal modeling languages are usually very proud of their visual representations. Goal models, however, tend to be complex with many relationships among many intentional elements and actors. In addition, it is difficult for one single concrete syntax to support the many types of goal language users in their various modeling and analysis tasks. For example, while GRL's graphical syntax helps with the understanding and analysis of goal models as an output representation, it is often inefficient as an input representation as it tends to slow down the creation and maintenance of complex models. Being largely based on i*, GRL's notation also likely suffers from many of the deficiencies identified for i* in terms of cognitive effectiveness [12]. The current syntax could be revised to fix these deficiencies, and new graphical syntaxes could even be added for specific types of users. Moreover, a more practical textual concrete syntax, like a programming language with a suitable Integrated Development Environment, could *also* be made available, this time mainly as an alternative input representation. A proposal was made by Liu and Yu in their original GRL proposal [11] but was never followed-up and it never made it to the standard. Similarly, for users unfamiliar with goal modeling, a tabular-based representation à la House of Quality [7] may be used. Fig. 1 shows an example of what Fig. 2 could look like in a tabular form, e.g., intentional elements are shown with their types, links (from rows to columns), and bindings to actors (shown as importance levels between 0 and 100).

| *Intentional elements* | *Links (A: AND, O: OR, D: Dependency, Others: contribution levels)* | | | | | | | | | | | | *Actors* | | | | *Type* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Reduce societal effects of dengue | Disease surveillance | Vector control | Vector surveillance | Perform dengue premise surveillance | Test for pesticide efficacy | Field agent-driven assessment | Home owner / tenant self assessment | Provide regular reports to SVCO | Keep public informed accurately a. t. | Do a good surveillance job | Ensure privacy | Dengue S&C Syetem | State Vector Control Official | Field Agent | Home Occupant | |
| Reduce societal effects of dengue | | A | A | A | | | | | | | | | 100 | | | | S |
| Disease surveillance | | | | | | | | | | | | | 0 | | | | T |
| Vector control | | | | | | | | | | | | | 0 | | | | T |
| Vector surveillance | | | | | A | A | | | | | | | 0 | | | | T |
| Perform dengue premise surveillance | | | | | | | O | O | + | | | | 0 | | | | T |
| Test for pesticide efficacy | | | | | | | | | | | | | 0 | | | | T |
| Field agent-driven assessment | | | | | | | | | | | D | − | 0 | | | | T |
| Home owner / tenant self assessment | | | | | | | | | | | | + | 0 | | | | T |
| Provide regular reports to SVCO | | | | | | | | | | | | | 80 | | | | S |
| Keep public informed accurately a. t. | | | | | | | | | D | | | | | 0 | | | S |
| Do a good surveillance job | | | | | | | | | | | | | | | 0 | | S |
| Ensure privacy | | | | | | | | | | | | | | | | 0 | S |

**Fig. 1.** Example of Potential Tabular Representation of a Goal Model

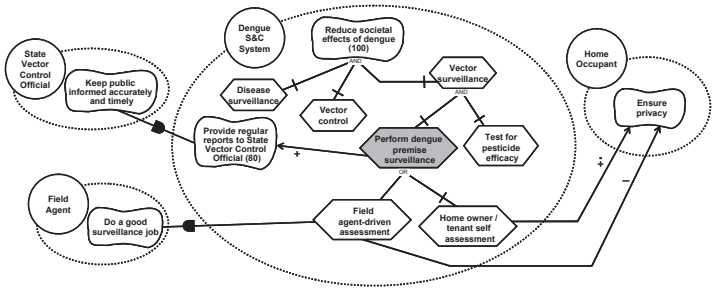



**Fig. 2.** Example for Modularity Issues

The various XML formats proposed over the years [4,9] are good as interchange formats, but not for human input. This concrete syntax issue is orthogonal to all other issues described in Section 3.

### 3.2 Envy: Lack of More Formal Semantics

Goal languages are often envious of the maturity of other modeling languages in terms of standardized semantics. GRL possesses a formal abstract syntax in the form of a standard metamodel. However, the meaning of a GRL model is largely defined by the evaluation mechanism chosen for that model (e.g., various categories of mechanisms are presented in [3], with other analysis techniques in [8]). The formal underpinning for evaluation/analysis mechanisms is rather ad hoc. Consequently, it is difficult to understand why one mechanism should be chosen over another. One option to improve formality is the definition and validation of a reference formal semantics for GRL rooted in mathematics (instead of typical operational semantics in the form of algorithms), for instance by following the guidelines proposed in [6]. This could be followed by the development of compliant interpretations in the form of constraint satisfaction problems, for which powerful analysis techniques and tools already exist.

### 3.3 Lust: Lack of Support for Interface Definitions

While it is somewhat possible to consider an actor as an encapsulated unit, this is quite difficult for a task (or any other intentional element) that is being decomposed into lower level elements. Even with actors, a clear interface definition is absent, leading to lustful situations between modeling elements. Dependencies help to a certain extent to separate goal trees of various actors, but relationships among elements of different actors may occur at any level of abstraction (see how the relationships between the four actors in Fig. 2 manifest themselves at different decomposition levels).

### 3.4 Sloth: Lack of Support for Relationship Abstractions

In GRL, there is no support (both from a language design point of view and from a tool support point of view) for describing the relationships of an intentional element

based on the relationships of its constituent elements. As an example, consider what the relationships of the highlighted task in Fig. 2 should be. Only one contribution is shown for this task, but several others exist when its lower-level elements are taken into account. Of course, not everything needs to be shown in a partial view such as a goal graph, but there should be some way to view *all* relationships of any given actor or intentional element. Tool builders are slothful in their support for dynamically deriving, from the complete goal model, a view for a particular element that takes the relationships of its children into account. Furthermore, it should be possible to explore a goal model with interactive queries such as "show me all intentional elements for this actor" and "expand this intentional element's relationships by two levels" rather than just having static views (queries are a common feature of UML modeling tools).

While dynamic views are mostly a tool support problem for an already finalized goal model, an arguably even worse situation occurs when the goal model is built in the first place. Let us assume that a contribution has been defined for a high-level intentional element to indicate this important relationship at this level of abstraction. When decomposing the element into lower-level elements, one may want to also refine the contribution shown for the element. This, however, is typically not possible without having to remove the higher-level contribution and consequently losing the high-level overview. Note that this issue depends significantly on the Envy, Wrath, and Lust sins; formal semantics with proper support for abstractions would allow interfaces to be defined more easily at various levels of abstraction and vice versa.

### 3.5 Gluttony: Lack of Support for Separation of Concerns

GRL suffers from gluttony in that all concerns often need to be consumed at once. With aspect-oriented modeling becoming more and more part of mainstream modeling techniques, goal models should support the identification and encapsulation of concerns regardless of whether they are crosscutting or not. This may involve simple support for tagging an element as being part of a concern to full-fledged support for aspect-oriented matching and composition [13]. Model transformations [1] and suggestions made for i* [5] may also help address this modularity issue. Gluttony is largely an orthogonal issue to all other issues in Section 3.

### 3.6 Wrath: Analysis Anomalies

There is a frustrating and even infuriating interaction between the desire to view relationships at different levels of abstraction and the deployed evaluation mechanisms. At the heart of this issue is that a relationship shown at more than one level of abstraction must be taken into account only once during analysis. While the left and middle GRL graphs in Fig. 3 are equivalent, the right goal graph leads to a different evaluation result because relationships with the softgoal are shown many times, at different levels of abstraction. Furthermore, the contribution value for the higher-level task in the middle goal graph needs to be set to 40 to reflect that the contributions of both sub-tasks in the left goal graph are collapsed into one higher-level contribution.

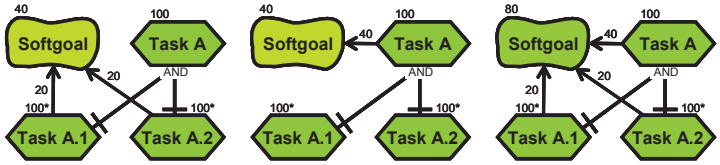



**Fig. 3.** Example for Analysis Anomaly

While GRL allows factors to be defined for intentional elements to differentiate their importance to a stakeholder, the importance of stakeholders themselves is typically not captured, even though all stakeholders do not have the same influence over the system under construction. Similarly, some intentional element may be non-negotiable from the point of view of one stakeholder but not from the point of view of another stakeholder, but this is also not captured.

### 3.7 Despair: Meaningless Numbers Problem

Modelers quickly lose hope in exploiting goal models in a useful way when the models do not properly reflect reality. One of the main difficulties when creating goal models is to specify appropriate contribution levels on contribution links. GRL should enable modelers to provide contribution ranges (to enable sensitivity analysis) or alternative levels as part of the goal model. A related issue is that it is often not possible to find a correspondence between the result of an evaluation and real life as it is difficult to translate real life into these seemingly arbitrary quantitative numbers and qualitative symbols, and vice versa. Greater reliance on key performance indicators (KPIs) derived from measurements of the real world may address the latter issue (as suggested in [14]). These issues are connected to Wrath as the ability to define sensible contribution levels is a prerequisite for a useful and sound evaluation of models.

### 3.8 Greed: Lack of Extensibility

Often, goal modeling languages do not allow language elements to be extended in a systematic way. This leads to different user communities that apply a particular goal modeling language in very greedy ways, and unfortunately, with incompatible dialects. An approach to address this issue is to provide the ability to a) define tags, stereotypes, or metadata for modeling elements, b) establish links between arbitrary modeling elements, c) group modeling elements, and d) define static constraints on the goal model (e.g., with OCL). This is partially supported in GRL [2], but there is much room for improving the packaging and modularization of such extensions. This issue has a dependency with Envy as the ability to define language extensions formally improves the semantic compatibility between variants of the modeling language. Better extensibility could also help tackle the lack of cognitive fitness discussed in Envy, but this may not always be the best way of producing a suitable dialect.

## 4 Conclusions and Future Work

We have presented a list of shortcomings of GRL with analogies to deadly sins. The most serious issues relate to vague definitions of goal model semantics (which essentially affects most sins), the lack of proper modularization, and the related difficulties when analyzing goal models at different levels of abstraction. The lack of modularized extensibility mechanisms is also a concern. We suspect that other popular goal modeling languages have committed many of these sins as well, to various extents.

In our future work on GRL, we plan to find absolution by a) solidifying the formal semantics with the help of mathematical and constraint-based techniques, b) investigating modules for stakeholders and intentional elements that provide precise interface definitions, c) providing textual and tabular syntaxes for goal models to improve model creation effectiveness, d) providing support for describing relationships of stakeholders and intentional elements based on their lower-level elements, and e) providing dynamic model exploration facilities (e.g., queries) in jUCMNav [10].

## References

1. Alencar, F., Lucena, M., Silva, C., Santos, E., Castro, J.: Improving the Modularity of i* Models. 4th International i* Workshop, CEUR vol. 586:3-8 (2010)
2. Amyot, D., Horkoff, J., Gross, D., Mussbacher, G.: A Lightweight GRL Profile for i* Modeling. RIGiM 2009, ER Workshops. Springer, LNCS vol. 5833:254-264 (2009)
3. 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(8):841-877 (August 2010)
4. Cares, C., Franch, X., Perini, A., Susi, A.: Towards interoperability of i* models using iStarML. Comput. Stand. Interfaces, 33(1):69-79 (January 2011)
5. Franch, X.: Incorporating Modules into the i* Framework. Advanced Information Systems Engineering, 22nd Int. Conf., CAiSE 2010. Springer, LNCS vol. 6051:439-454 (2010)
6. Harel, D., Rumpe, B.: Meaningful modeling: what's the semantics of "semantics"? IEEE Computer, 37(10):64-72 (2004)
7. Hauser, J.R., Clausing, D.: The house of quality. Harvard Business Review, May-June, 63-73 (1988)
8. Horkoff, J., Yu, E..: Analyzing goal models: different approaches and how to choose among them. 2011 ACM Symp. on Applied Computing (SAC '11). ACM, 675-682 (2011)
9. ITU-T: User Requirements Notation (URN) – Language definition, ITU-T Recommendation Z.151 (11/08). Geneva, Switzerland (2008); http://www.itu.int/rec/T-REC-Z.151/en
10. jUCMNav 4.3.0, http://softwareengineering.ca/jucmnav
11. Liu, L. and Yu, E.: GRL - Goal-oriented Requirement Language (2000); http://www.cs.toronto.edu/km/GRL/
12. Moody, D., Heymans, P., Matulevičius, R.: Visual syntax does matter: improving the cognitive effectiveness of the i* visual notation. Requirements Eng. J. 15(2):141-175 (2010)
13. Mussbacher, G.: Aspect-oriented User Requirements Notation. Ph.D. thesis, SITE, University of Ottawa, Canada (November 2010)
14. Pourshahid, A., Chen, P., Amyot, D., Forster, A.J., Ghanavati, S., Peyton, L., Weiss, M.: Business Process Management with the User Requirements Notation. Electronic Commerce Research, 9(4):269-316 (December 2009)

# iStarML: Principles and Implications[†]

Carlos Cares[1,2], Xavier Franch[2]

[1] Universidad de La Frontera, Av. Francisco Salazar 01145, 4811230, Temuco, Chile,
[2] Universitat Politècnica de Catalunya, c/ Jordi Girona 1-3, 08034, Barcelona, Spain,
{ccares,franch}@essi.upc.edu
http://www.essi.upc.edu/~gessi/

**Abstract.** iStarML is an XML-based format for enabling *i** interoperability. A relevant difference with any other interoperability proposal is that iStarML is founded under the assumption that there is not a common ontology guiding this communication proposal. The different *i** variants and even particular applications proposing new language constructors forced to confront a theoretical approach for supporting an interoperability approach in an evolving and variable semantic scenario. In this paper we focused on the theories behind the iStarML proposal, which include sociological, cybernetics and linguistics approaches. Finally, we apply what these theories predict to the case of the *i** framework and its research community.

**Keywords:** *i** Framework, iStar, variants, interoperability.

## 1 Introduction

The *i** (iStar) framework has become a recognized and widespread framework in the Information Systems Engineering community. Given that the *i** framework incorporates goal- and agent-oriented modelling and reasoning tools, it has been a milestone for providing the basis, developing and spreading goal-orientation as a relevant paradigm in Requirements Engineering and agent orientation in Software Engineering. As a result of different interpretations and adoptions, several *i** variants have emerged, which have evolved at different maturity levels. Some of them have reached the category of industrial standard (e.g., GRL in ITU-T) whilst others are in some initial stages of development or were conceived for supporting a localized (in scope and time) problem. Besides, there is a set of proposals that cannot be considered *i** variants because they simply include new or modified language constructions. In [1] we have summarized a literature review reporting this fact.

As an effect of the past and even current proliferation of different *i** variants, a derived set of software tools and prototypes have been generated. However, interoperability has been an elusive target. Although there is a clear core common to virtually all *i** variants, tool interoperability is founded on an agreement that implies the existence of a shared ontology, which it has not been the case of *i** variants.

[†] This work has been partially supported by the Spanish project TIN2010-19130-C02-01.

We have tackled the problem of proposing an interoperability framework for *i** variants. The aim is model interchange to "work together", which is the deep meaning of interoperability. We have called iStarML to the proposal of a XML-based format language [2]. It incorporates structures which allows the representation of a wide set of *i** variants. As part of the theoretical approach that supports iStarML we have presented the proposal from [3] as a key theory because it introduces the concept of supermetamodel, which has been a key concept for demonstrating the reduction of complexity of the translation among *n i** variants. Besides, this framework introduce degrees of semantic preservation which we have used to qualify the translation from an *i** variant to another [1]. The iStarML applications and usages that we have promoted [4] illustrate the feasibility not only of interaction between different *i** variants but also the feasibility of using iStarML as a valid textual representation over which other applications can be built.

## 2 Objectives of the Research

However, we have not explained the theoretical principles which have supported the idea of enabling interoperability without having a shared and common ontology, i.e. why and how interoperability can be sustained in a human poly-semantic scenario. The goal of tackling this social perspective was to consider the Requirements Engineering's (and also Scientifics') principle of proposing a better approach if there a model (or theory) for *understanding why*.

In this paper we briefly present different theories that support the idea of having communication without a shared ontology, some of them from linguistics, others from cybernetics, and also from sociology of science. The social nature of these theories gives us some information of what could be expected as research community and the feasible social and technical scenarios that we could expect.

## 3 Scientific Contributions

Basic foundations in Computer Science affirm that interoperability requires a stable semantic scenario to reach high levels of interoperability. However, this is not the situation in the *i** community because different tools have been inspired on different *i** variants. They do not only have different metamodels (syntax) but also they have a different mapping to objects of reality (semantic) of their language constructs. We present principles from Sociology of Science, Cybernetics and Linguistics which support the idea of enabling interoperability in spite of the absence of a shared ontology.

### 3.1 Sociology of Science Approaches

Sociology of Science embraces those studies that explain the foundations of science (i.e., Philosophy of Science) from a sociological perspective. One of the relevant contemporary philosophers has been Thomas Kuhn, who has modelled science

evolution through discontinuous jumps [5]. Kuhn called these jumps "scientific revolutions", which are leaded by a reduced set of pioneers who change the basic conceptualizations and build a new way of doing science. This new way of perceiving and researching is called "scientific paradigm". Each paradigm has an initial and underlying ontology, which is "known" (in the beginning) only by the pioneers, who try to communicate it and teach the new way of doing research using this ontology. The impossibility of explaining the new ontology based on the previous ontology is called by Kuhn "the incommensurability problem". In spite of that, along time, the ontology is spread, and used, not always as it was proposed but according to the particular interpretations of the followers. This phase is called "the revolutionary stage". However, at some moment, the conceptualization converges to a stable and shared ontology (the key concepts of the paradigms) and epistemology (what the community accept as valid methods for knowledge production). When it happens, the following phase, the normal-science stage, starts. The cycle continues when theoretical anomalies appear (unsatisfactory explanations) and a new pre-revolutionary stage germinates. Although the original Kuhn's idea was to explain big scientific movements, lately he recognizes that there are a lot of small and even micro-revolutions which present the same behaviour. Figure 1 illustrates the stages.

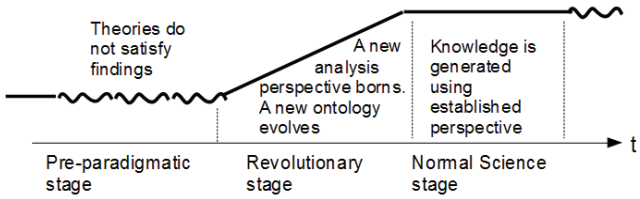



**Fig. 1**. Kuhn's model of a paradigmatic shift

Bourdieu's theory of scientific fields [6] is the second approach from sociology of science that we reviewed. In this theory the key concept is the symbolic capital. Scientific behaviour is associated to fields that have, in its centre, the highest concentration of symbolic capital; normally the leaders/pioneers of the fields occupy these positions. They dominate the concepts and try to spread their ideas. Scientifics try to maximize their symbolic capital by two ways: moving to the centre of the field, which means to exactly follow the pioneers' ideas and collaborate with them, or generating new scientific fields by the intersection with other fields that allow them occupying the centre of a new scientific field. Either in the zones of lower symbolic capital or in the new fields, the concepts are not used as in the centre of the reference scientific field. In Fig. 2, left, the idea of scientific field is illustrated.

In both theories it is recognized the fact that research activity without a fully shared ontology, as it happens in the *i** community, is feasible. Also, from these two theories, we cannot expect the paradigms (i.e., the *i** framework) be extended over time or the *i** field be kept the same way is in the centre and in the very far periphery. A point of difference among these two approaches is the position about community agreement on a shared ontology. From Kuhn it is predicted that having a shared ontology stops the revolutionary stage and from Bourdieu it is predicted that we will

always have uncontrolled interpretations and uses. However, this apparent contraction is not real if we suppose that having static scientific fields is part of "normal science".

### 3.2 Cybernetics Approaches

Cybernetics has been conceptualized as the General Theory of Control [7]. We can rescue from Cybernetics both, classical and new conceptual frameworks. A classic contributor has been Ross Ashby, who proposed a definition of intelligence from the control perspective. Thus, intelligence is understood as a repertoire of behaviours; therefore having more intelligence or variability means having a broader repertoire of behaviours. In this theory it is said that humans are able to create intelligence amplifiers in order to enlarge their control capabilities, which means to increase variability. One of Ashby's examples is the difference between *the ships being loaded quickly and easily by movements of a control handle, or slowly and laboriously by hand* [8]. Other intelligence amplifiers can be a dictionary, a sunglasses, a calculator and of course, a modelling tool, because they improve the repertoire of behaviours.

In addition, contemporary cybernetics takes concepts as *autopoiesis* [9] to explain that biological-based systems continuously regenerate the processes that produce them (*autopoiesis*). This should be understood from both, as an internal point of view (transformations) and as an external one (interactions). It is said that biological systems are operationally closed systems which are self-produced and self-referred. It means their actions are the effect of their interpretations (meaning-making process). It also implies that operational distinctions (ontology) that use a system in order to guide its interactions and transformations are not observable ones since they are internal to the system. Therefore a biologically-based communication process emerges without an explicit (non-external and non-shared) ontology. As an extension of that, and due to intelligence amplifiers (e.g. modelling tools) are part of the interactions of the system with its environment, then interoperability will take place if the meaning-making process produces some interpretation (e.g. for an arriving model) which improve variability.

### 3.3 Semiotics Approaches

Semiotics is about the interpretation of signs, syntax, semantics and pragmatics as well. The main focuses are models that explain human communication from both individual and collective perspectives. A relevant semiotic concept is language expressions and their meanings, which changes depending on the community. What is interesting for us are collective perspectives because they may model communication in scientific communities. Semiotics considers that some language expressions can reach stable meanings in the natural dynamic of meaning systems which implies that these expressions can be used as *intepretants*, i.e. using in sentences that explain other meanings. This is why an explanation might be completely understood by some people meanwhile some others will have a different conception about it or partially get what it means.

Yuri Lotman [10] introduced the concept of semiosphere to explain communication inside medieval human cultures. Firstly he expresses that mono-semantic systems do not exist in isolation. These related systems are part of a continuous sphere of meaning called semiosphere. Lotman explains that the boundary is the area of accelerated semiotic process (interpretations). This theoretical approach affirms that in peripherical areas, where structures are "slippery", less organized and more flexible, the dynamic process meets with less opposition and, consequently, develops more quickly. Then, one may say that the new semiosphere grows leaving in the centre the dominant semiotic system constituted by a wide set of stable concepts. This theory seems be very applicable for *i** community as part of a more general software engineering community and also for a particular *i** variant community. From this perspective the sentence from Lotman saying that the creation of meta-structural self-descriptors (grammar) appears to be a factor which dramatically increases the rigidity of the semiosphere's structure and slows down its development, it can be easily understood, e.g. by producing some social-based standard. In Fig. 2, right, the general idea of semiosphere is illustrated.

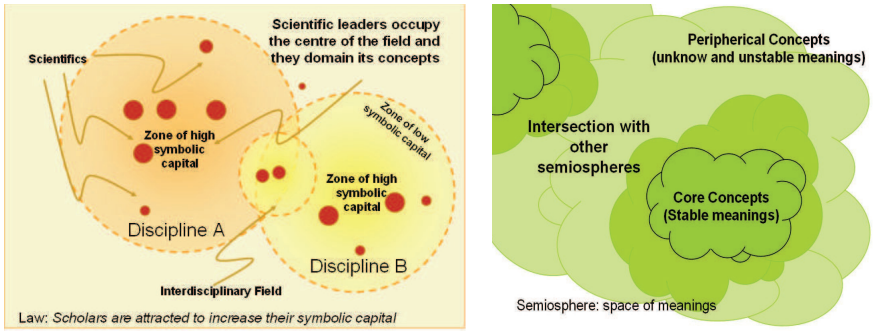



**Fig. 2.** Concepts on Bourdieu's Scientific Fields and Lotman's Semiospheres

### 3.4 Implications for iStarML and the *i** Research Community

After the theoretical analysis which only in part we have summarized here, some conclusions emerge. Firstly, a complete shared ontology is neither a human requisite for interaction nor for action, even when this communication is intermediated by intelligence amplifiers as software tools. Therefore, an interoperability proposal appears theoretically feasible. Moreover, we conclude that the structure of the interoperability language should correspond to the way of a Bourdieu's scientific field or, similarly, to a Lotman's semiosphere, i.e. core or stable concepts at the centre, and the increasing of semantic variability and additional constructs towards the periphery. If the inteoperability language may reproduce the semiosphere structure, then intermediate structures should be in terms of core structures. Therefore, we proposed iStarML using a concentric ring structure: (1) the core set of stable and common abstract concepts (*actor, intentional element*, *intentional link, dependency*), (2) a core set of common concepts established by a predefined set of main tag attributes (e.g., *type="softgoal"*), (3) a space for specifying variations on existing but not so common

attributes (e.g., *value="xor"*) and (4) new elements (e.g. type="norm" prior="low". Therefore an iStarML message should be part of a meaning-making process, thus, it, could be interpreted depending of the target *i** variant community.

In addition, the application of these theoretical frameworks allows to derive some conjectures about *i** community: (1) If *i** represents a materialization of a scientific revolution then is not expected that the *i** ontology keeps being an underlying ontology. It will be explicitly expressed at some moment. (2) The externalization of an *i** ontology will not stop the scientific activity; it only will change its state. Proliferation of interpretations will be stopped but "normal" (in the sense of Kuhn) applications will be massively reported afterwards. (3) Following Lotman's and Bordieu's theories, externalizing an ontology may be a very difficult job if central scholars of the field are not involved. (4) Expressing an ontology will not avoid intersections to other scientific fields, therefore, proliferation of applications with and without *i** modifications will take place.

## 4 Conclusions

The theoretical foundation for proposing iStarML, an interoperability proposal for *i** variants, has been reviewed. It included sociology of science, cybernetics and semiotics. Following these theories about human communication and scientific behaviour we have explained some reasons behind central features of iStarML.

However, the social nature of this theoretical framework allowed obtaining conjectures (i.e. theory predictions) about the future of the *i** community. Mainly they point out to the expression and formalization of the *i** ontology which stops the revolutionary stage, fixes the grammar and establishes a new research period about applications under a stable semantic scenario. In this hypothetical scenario iStarML should be revised according to the inclusion of a set of language constructs from this explicit formalization.

## References

1. Cares, C., Franch, X.: A Metamodelling Approach for *i** Model Translations. CAiSE 2011.
2. Cares, C., Franch, X., Perini, A., Susi, A.: Towards Interoperability of *i** Models Using iStarML. *Computer Standards & Interfaces*, 33(1) (2011).
3. Wachsmuth, G.: Metamodel Adaptation and Model Co-adaptation. ECOOP 2007.
4. Colomer, D., López, L., Cares, C., Franch, X.: Model Interchange and Tool Interoperability in the *i** Framework: A Proof of Concept. WER 2011.
5. Kuhn, T.: *The Structure of Scientific Revolutions*. The University of Chicago Press (1996).
6. Bourdieu, P.: The Specificity of the Scientific Field and the Social Conditions of the Progress of Reason. *Social Science Information* 14(6), 1975.
7. Beer, S.: *Cybernetics and Management*. The English Universities Press (1967).
8. Ashby, W.R.: *An Introduction to Cybernetics*. Chapman & Hall (1957).
9. Maturana, H.R., Varela, F.J.: *De Máquinas y Seres Vivos: Autopoiesis la Organización de lo Vivo*. Santiago de Chile, Editorial Universitaria (1998).
10. Lotman, J.: *On the Semiosphere*. *Sign Systems Studies*, 33(1) (2003).

*CEUR Proceedings of the 5th International i* Workshop (iStar 2011)*

# Empirical Evaluation of Tropos4AS Modelling

Mirko Morandini, Anna Perini, and Alessandro Marchetto

FBK-CIT, Trento, Italy,
{morandini,perini,marchetto}@fbk.eu,

**Abstract.** Our work addresses the challenges arising in the development of self-adaptive software, which has to work autonomously in an unpredictable environment, fulfilling the objectives of its stakeholders, while avoiding failure. In this context we developed the Tropos4AS framework, which extends the AOSE methodology Tropos to capture and detail at design time the specific decision criteria needed for a system to guide self-adaptation at run-time, and to preserve the concepts of agent and goal model explicitly along the whole development process until run-time.
In this paper, we present the design of an empirical study for the evaluation of Tropos4AS, with the aim of assessing the modeling effort, expressiveness and comprehensibility of Tropos4AS models. This experiment design can be reused for the evaluation of other modeling languages extensions.

**Key words:** Agent-oriented software engineering, Empirical studies, Self-adaptive systems.

## 1 Introduction

Today's software is expected to be able to work autonomously in an open, dynamic and distributed environment. *Self-adaptive software systems* were proposed as a solution to cope with the uncertainty and partial knowledge in such environments. The development of such software, which should automatically take the correct actions based on knowledge of what is happening, guided by the *objectives* assigned by the stakeholders, gives rise to various challenges: the software needs multiple ways of accomplishing its purpose, enough knowledge of its construction and the capability to make effective changes at runtime, to be able to autonomously adapt its behaviour to satisfy the requirements, shifting decisions which traditionally have been made at design-time, to run-time.

In our recent work we try to address these challenges, proposing the Tropos4AS (Tropos for Adaptive Systems) methodology [1]. Tropos4AS aids the software engineer in capturing and detailing at design time the specific knowledge and decision criteria that will guide self-adaptation at run-time. Moreover, it brings the high-level requirements, in form of goal-models, to run-time, to enable the system to monitor their satisfaction, to reflect upon them and to guide its behaviour according to them.

Like Tropos4AS, various extensions of the Tropos modelling language and methodology were proposed in the last years, specific for different purposes and various domains. However, only few attempts were made to assess such extensions by means of empirical studies. We present the design of two experiments with subjects, which have the scope to assess the novel extensions of Tropos4AS by comparison with the general methodology Tropos. Applying proper statistical tests, we are able to collect evidence on the effectiveness (modelling effort, model correctness and model comprehensibility) of Tropos4AS models, evaluating the results of modelling tasks, comprehension tasks and supporting questionnaires. The design is general and thus reusable for the evaluation of other specific extensions to general modeling languages.

## 2 Background

The Tropos4AS methodology extends Tropos to provide a process and modelling language that captures at design time the knowledge necessary for a system to deliberate on its goals in a dynamic environment, thus enabling a basic feature of self-adaptation. It integrates the goals of the system with the environment, and gives a development process for the engineering of such systems, that takes into account the modelling of the environment and an explicit modelling of failures. Tropos goal modelling is extended along different lines:

*i*) Capturing the influence of artifacts in the surroundings of an actor in the system to the actor's goals and their achievement process. This is achieved by modelling an actor's environment and defining *conditions* on the environment artifacts, to guide or guard state transitions in the goal satisfaction process, e.g. achievement conditions, goal creation conditions or failure conditions.

*ii*) The definition of goal types (maintain, achieve,...) and additional inter-goal relationships (inhibition, sequence), to detail the goal achievement and alternatives selection dynamics.

*iii*) Modelling possible failures, errors and proper recovery activities, to elicit missing functionalities to make the system more robust, to separate the exceptional from the nominal behaviour of the system, and to create an interface for domain-specific diagnosis techniques.

For the aim of providing an explicit representation of high-level requirements at run-time and lowering the conceptual gaps between the software development phases, we perserve the concepts of *agent* and *goal model* explicitly along the whole development process. The detailed Tropos4AS goal models represent the *"knowledge level"*, that is, the rationale behind the execution of specific tasks (i.e. plans). Adopting an implementation architecture which supports goal models, the software can navigate and reason on them and exploit available alternatives satisfy its requirements. A complete translation of requirements concepts to traditional software level notions, such as classes and methods of object-oriented programming, is avoided. This contributes to a smoother transition between the development phases, reducing loss and conceptual mismatch and simplifying the

tracing of decisions made in requirements analysis and design to the implementation and vice-versa.

A direct mapping from goal models to implementation concepts is defined, relying on agents with a BDI (Belief-Desire-Intention) architecture and a native support for the concepts of agent and goal. With a supporting middleware, an explicit, navigable and monitorable representation of goal models at run-time is realised. Tropos4AS (with the graphical modelling language presented in Figure 1) is detailed in [1], and [2]. Details for the operational semantics attributed to condition evaluation and to the satisfaction of goals in goal models, can be found in [3]. Note that the optimisation of a system's behaviour, by the use of run-time goal model reasoning, learning or knowledge acquisition strategies, is not part of, but would be complementary to Tropos4AS.

*Tool support.* The *Taom4E* Tropos modelling tool (selab.fbk.eu/taom) supports modelling of extended Tropos4AS models and includes a plug-in for an automated code generation (***t2x*** tool), base on the *Taom4E* Tropos modelling tool which uses the *Jadex* agent framework as implementation platform.

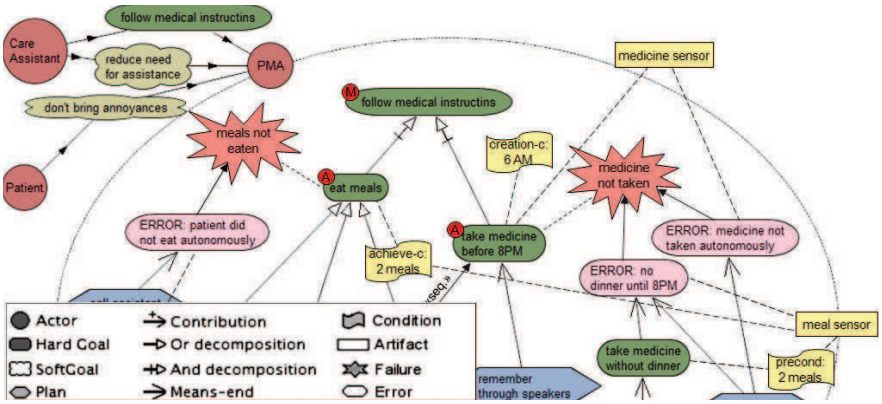



**Fig. 1.** Fragment of the Tropos4AS model for a patient monitoring system, one of the objects used in the the comprehension experiment.

## 3 Empirical Evaluation

The evaluation of novel extensions to a development methodology poses various challenges, since they are usually not yet extensively used in practice. Moreover, it is challenging to set up a fair and meaningful comparison for the evaluation of the introduced extensions: An empirical study which consists of a comparison of Tropos4AS with a methodology with a similar scope but with different roots, would inevitably also assess the performance of the whole Tropos language. Therefore, an evaluation limited to the novel extensions, which is our

scope, would be impossible. Conversely, an empirical study which involves implementation, would require participants that are experienced in the use of the implementation language, and demand a very high time effort. Also, a comparison of the whole *modelling process* would not be feasible within the given time constraints. Thus, to assess the novelties introduced with Tropos4AS, we propose to perform a controlled experiment with subjects, comparing the Tropos4AS modelling language to the underlying Tropos modelling language[1], which showed its effectiveness in various studies, e.g. [5].

The evaluation of a modelling language can be characterised by three main aspects: (1) the effort for modelling, (2) the effectiveness for capturing the requirements and (3) the comprehensibility of the obtained models. The study we propose consists of modelling and comprehension tasks performed by a group of subjects, and is divided into two experiments:

**Modelling:** we evaluate if Tropos4AS is effective in modelling self-adaptive systems, with an acceptable modelling effort, in comparison to Tropos.

**Comprehension:** we evaluate if the Tropos4AS modelling extensions increase the comprehensibility of the requirements of a system.

The design of the experiments follows the guidelines by Wohlin et al. [6] and allows to have a high degree of control over the study, to achieve results with statistical significance. Tropos and Tropos4AS represent the control *treatment* and the treatment to evaluate. The quality focus of the experiment concerns the capability of the treatments in supporting the analysts in requirements modelling and comprehension. The target *subjects* are researchers and Ph.D. students, while the *objects* of study are requirements specifications (textual and graphical) of two software systems with adaptivity features. It is however important that these systems can be modelled in a satisfactory way with both the general and the domain-specific methodology. We define three research questions (together with the relative null- and alternative hypotheses):

*RQ*1**:** Is the effort of modelling requirements with Tropos4AS significantly higher than the effort of modelling them with Tropos?

*RQ*2**:** Is the effectiveness of Tropos4AS models significantly higher than the effectiveness of Tropos models, for representing requirements of an adaptive system?

*RQ*3**:** Do Tropos4AS models significantly improve the comprehension of the requirements of a self-adaptive system, in comparison to Tropos models?

To investigate on these questions (with the aim of showing if the relative null-hypothesis can be rejected or not), we run a modelling and a comprehension experiment, which both adopt a paired, counterbalanced experiment design based on two laboratory sessions, such that the subject perform the experimental task twice, once with each object and treatment, exploiting all possible combinations. This lets us evaluate the performance of the subjects with both treatments, avoiding learning effects.

[1] We refer to the Tropos modelling language, as defined in [4]. In particular, we focus on Tropos *goal diagrams*, which are mainly affected by the novel extensions.

*Design of the modelling experiment.* The modelling experiment covers the research questions $RQ1$ and $RQ2$ and consists of:

1. a presentation and training session for the subjects, to introduce or refresh notions related to both treatments, and to explain the experiment tasks;
2. a pre-questionnaire to capture information about the experience of the subjects and about the clearness of the notations and of the experimental task;
3. two supervised laboratory sessions concerning a modelling task to be performed in an open time frame, asking the subjects to **model with as much details as possible the textual requirements specification**s handed out, with the assigned modelling language;
4. post-questionnaires asking about the perceived effort for each treatment and about personal opinions, comparing both treatments.

The research questions include the abstract terms *effectiveness* and *effort*, which have to be detailed in order to characterise these two terms for the scope of the study and to associate them to variables which can be evaluated. $RQ1$ is decomposed to aspects considering time, perceived effort, and the difficulties encountered while modelling, while the aspects for $RQ2$ consider the expressiveness of the modelling language as perceived by the subjects and the correctness of the models built. These aspects are evaluated collecting the questionnaire results (on an ordinal $1 \ldots 5$ *Likert* scale, from *strongly agree* to *strongly disagree*) and measuring the time spent. Moreover, model correctness is evaluated against an expert-made *gold standard* model, evaluating the coverage of three predefined software execution scenarios.

*Design of the comprehension experiment.* The comprehension experiment, covering $RQ3$ and conducted with the same subjects, consists of:

1. two laboratory sessions concerning a comprehension task to be performed in a fixed time, asking the subjects to **answer to five comprehension questions on the object assigned, by looking at the Tropos or Tropos4AS models handed out** (built by modelling experts) and the respective textual requirements specifications;
2. a final questionnaire with questions on the subjective perception of subjects with respect to the experiment and the treatments.

The main dependent variable is the correctness of the subjects' answers to the comprehension questions, measured by comparing the answers given to gold standard answers, in terms of precision and recall.

*Statistical evaluation.* To determine if the null-hypotheses can be rejected and thus an affirmative answer can be given to the research questions, considering the nature of the variables and the experiment design, we apply a non-parametric, paired *Wilcoxon* test, adopting a 5% significance level for the obtained *p-values* (refer to [6] for details). Medians, averages and *Cohen.d* effect size are applied to analyze trends and to estimate the magnitude of the obtained results. Similar tests are used to evaluate the adequateness of the experimental settings by an analysis of the pre-questionnaires.

## 4 Conclusion

We described the design of an empirical study consisting of two controlled experiments, which aims to evaluate the extensions introduced by the Tropos4AS framework to the Tropos modelling language. The structured experimental set-up would be suitable in general to contribute to the evaluation of modeling language extensions.

We run the experiment with 12 researchers and PhD-students, with two small systems as objects and proper questionnaires. The analysis of the obtained data with the abovementioned statistical tests gave positive, mostly statistically significant results for both the expressiveness and the comprehensibility of Tropos4AS [7], while the modelling effort (except for looking up in the language specifications) seems not to be significantly higher than for Tropos. Analyzing possible threats to validity, a statistical evaluation (ANOVA test) of various co-factors (object, subject experience, subject position, laboratory) has shown that there was no significant impact on the obtained results. Conversely, some subjects reported difficulties in traditional Tropos modelling because of missing concepts in the language. A complete analysis of the results and the replication packages are available in [2].

We plan to complete the assessment, repeating the study with a higher number of subjects and evaluating the complete modelling process, e.g. by an off-line (observational) case study on the development of a system in a dynamic domain.

## References

1. Morandini, M., Penserini, L., Perini, A.: Towards goal-oriented development of self-adaptive systems. In: SEAMS '08: Workshop on Software engineering for adaptive and self-managing systems, ACM (2008) 9–16
2. Morandini, M.: Goal-Oriented Development of Self-Adaptive Systems. PhD thesis, DISI, Università di Trento, Italy (March 2011) Available at http://eprints-phd.biblio.unitn.it/511.
3. Morandini, M., Penserini, L., Perini, A.: Operational Semantics of Goal Models in Adaptive Agents. In: 8th Int. Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS'09), IFAAMAS (May 2009)
4. Penserini, L., Perini, A., Susi, A., Mylopoulos, J.: High variability design for software agents: Extending Tropos. ACM Transactions on Autonomous and Adaptive Systems (TAAS) **2**(4) (2007)
5. Hadar, I., Kuflik, T., Perini, A., Reinhartz-Berger, I., Ricca, F., Susi, A.: An empirical study of requirements model understanding: Use Case vs. Tropos models. In: SAC. (2010) 2324–2329
6. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in software engineering: an introduction. Kluwer Academic Publishers, Norwell, MA, USA (2000)
7. Morandini, M., Marchetto, A., Perini, A.: Requirements Comprehension: A Controlled Experiment on Conceptual Modeling Methods. In: Proceedings of the first Workshop on Empirical Requirements Engineering (EmpiRE11). (August 2011)

# Detecting Judgment Inconsistencies to Encourage Model Iteration in Interactive i* Analysis

Jennifer Horkoff[1], Eric Yu[2]

[1]Department of Computer Science, [2]Faculty of Information, University of Toronto
jenhork@cs.utoronto.ca, yu@ischool.utoronto.ca

**Abstract.** Model analysis procedures which prompt stakeholder interaction and continuous model improvement are especially useful in Early RE elicitation. Previous work has introduced qualitative, interactive forward and backward analysis procedures for i* models. Studies with experienced modelers in complex domains have shown that this type of analysis prompts beneficial iterative revisions on the models. However, studies of novice modelers applying this type of analysis do not show a difference between semi-automatic analysis and ad-hoc analysis (not following any systematic procedure). In this work, we encode knowledge of the modeling syntax (modeling expertise) in the analysis procedure by performing consistency checks using the interactive judgments provided by users. We believe such checks will encourage beneficial model iteration as part of interactive analysis for both experienced and novice i* modelers.

**Keywords:** Goal-and Agent-Oriented Models, Early Requirements Engineering, Model Analysis, Interactive Analysis, Judgment Consistency.

## 1 Introduction and Motivation

Modeling and analysis can be challenging in Early Requirements Engineering (RE), where high-level system requirements are discovered. In this stage, hard-to-measure non-functional requirements are critical, and understanding the interactions between systems and stakeholders is a key to system success. Because of the high-level, social nature of Early RE models, it is important to provide procedures which prompt stakeholder involvement (interaction) and model improvement (iteration). To this end, our previous work has introduced interactive, qualitative analysis procedures over agent-goal models (specifically, i* models) which aim to promote model iteration and convergent understanding [1-5]. These procedures are interactive in that, where partial or conflicting analysis labels appear in the model, users are asked to provide a human input as resolution before the procedure proceeds further.

Experiences with skilled i* modelers in complex case studies have provided evidence that interactive analysis prompts further elicitation and beneficial model iteration [1,3]. However, case studies comparing ad-hoc to semi-automated interactive analysis using novice participants showed that model iteration was not necessarily a consequence of systematic interactive analysis, but of careful

examination of the model prompted by analysis in general [6]. We concluded that the positive iterative effects of interactive analysis found in previous case studies were dependent upon modeling expertise (the ability to notice when analysis results were inconsistent with the model), domain expertise (the ability to notice when results differed from the modeler's understanding of the world), and interest in the domain being modeled (caring enough about the modeling process to improve the model).

One consequence of these results would be to recommend that interactive analysis be performed by, or in the presence of, someone with significant knowledge of i*. However, this is often not a reasonable expectation, as many i* modelers may be new to the notation and modeling technique, and will want to be guided by evaluation procedures in analyzing the model. As a result, we aim to embed some modeling expertise into the analysis procedure and corresponding tool support by detecting inconsistencies using the results of interactive judgments.

Case study experiences show that making judgments over the model can lead the modeler to revise the model when the decision made using domain knowledge differs from what is suggested by the model. For instance, in the simple example model for Implement Password System in Fig. 1, if the application Asks for Secret Question but does not Restrict Structure of Password, model analysis would suggest that Usability would be at least partially satisfied. If instead, the modeler thinks that Usability should be partially denied, this means the model is inaccurate or insufficient in some way. Perhaps, for example, Usability also requires hints about permitted password structure.

However, in our student study we found several occasions where novice modelers made judgments that were inconsistent with the structure of the model, and did not use these opportunities to make changes or additions to the model. To place this situation in the context of our previous example, if the Application Asks for Secret Question but does not Restrict Password the student may have decided that Usability was still partially denied, continuing the evaluation without modifying the model to be consistent with their judgment.

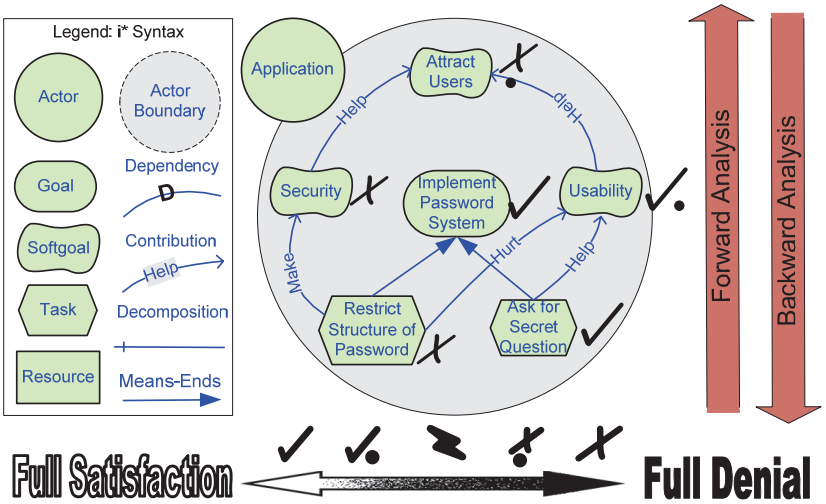



**Fig. 1.** Simple Example of an i* Model for a Password System from [2,11]

Similarly, our studies and experiences showed that it is easy to forget previous judgments over an intention element and to make new judgments which are inconsistent with previous judgments. For example, a user may decide that if Security is partially denied and Usability is partially satisfied, Attract Users is partially denied. In another round of analysis, if they are presented with an identical situation, they may now decide that Attract Users has a conflict.

We use these observations to guide us in embedding modeling expertise into interactive i* analysis by detecting inconsistencies using judgments. We distinguish and check for two types of inconsistencies: inconsistencies with the structure of the model and inconsistencies with judgment history. In this work, we take the initial steps of describing these checks formally and through examples. Future work will test the practical effectiveness of these checks in encouraging beneficial i* model iteration.

## 2 Background

We assume the reader is familiar with the i* Framework. The evaluation procedures and their extensions described in this work use the syntax defined in [7]. More information can also be found on the i* Wiki Guidelines [8].

In order to more precisely define the consistency checks introduced in this work, we summarize the formalization of the i* framework presented in [2]. The definitions use the → notation to represent relationships between elements, so if $(i_1,i_2) \in R$ we write this as $R{:}i_1 \rightarrow i_2$.

***Definition: i*model.*** *An i* model is a tuple <I, R, A>, where I is a set of intentions, R is a set of relations between intentions, and A is a set of actors. Each intention maps to one type in {Softgoal, Goal, Task, Resource}. Each relation maps to one type in* $\{R^{me}, R^{dec}, R^{dep}, R^{c}\}$*, means-ends, decomposition, dependency, and contribution links, respectively.*

Analysis labels are used in i* to represent the degree of satisfaction or denial of an intention. We use the formal definition of analysis predicates from [2], adapted from [9]:

***Definition: analysis predicates.*** *We express agent-goal model analysis labels using a set of predicates, V, over* $i \in I$*. Each* $v(i) \in V$ *maps to one of {S(i), PS(i), C(i), U(i), PD(i), D(i)} where S(i)/PS(i) represents full/partial satisfaction, C(i) represents conflict, U(i) represents unknown, and D(i)/PD(i) represents full/partial denial.*

In addition, we have defined a conceptually useful total order where $v_1 > v_2$ implies that $v_1$ is more desirable (or "higher") than $v_2$. This order is as follows:

$$S(i) > PS(i) > U(i) > C(i) > PD(i) > D(i) \quad (1)$$

The framework for interactive goal model analysis summarized in [10] currently provides two types of analysis procedures: forward (from alternative solutions to goals) [1,3,4] and backward (from goals to solutions) [2,5]. Generally, the procedures start from initial labels expressing the analysis questions, e.g. what if the Application Restricts Structure of Password and Asks for Secret Question? (forward) or is it possible for Attract Users to be at least partially satisfied? (backward). Propagation is automatic following rules defined in our previous work. Propagation can be described via the forward and backward propagation axioms described in [2]. Generally, for an intention $i \in I$, where $i$ the destination of one to many relationships, $r \in R : i_1 \text{ x } \ldots \text{ x } i_n \rightarrow i$, these predicates take on the form:

**Forward Propagation:**
*(Some combination of* $v(i_1) \wedge \ldots \wedge v(i_n), v \in V) \rightarrow v(i)$
**Backward Propagation:**
$v(i) \rightarrow$ *(Some combination of* $v(i_1) \wedge \ldots \wedge v(i_n), v \in V)$

The interactive nature of the procedures comes when human judgment is needed to resolve incoming partial or conflicting labels (forward) or to provide feasible combinations of incoming labels to produce a target label (backward). New judgments are added to the model formalization by replacing the axioms defined above for an intention with new axioms of the same form, describing the judgment. For example, given S(Restrict Structure of Password) and S(Ask for Secret Question) (both alternatives are satisfied),we decide that Usability has a conflict, C(Usability), we would remove all axioms having Usability as a target or source and add:

**Forward:** S(Restrict Structure of Password) ∧ S(Ask for Secret Question) → C(Usability)
**Backward:** C(Usability) → S(Restrict Structure of Password) ∧ S(Ask for Secret Question)

For simplicity, in this work we will refer to the left side of the forward propagation axioms as a *combination of labels, CL*, and the right side as the *individual label, IL*. Forward judgments then consist of $CL \rightarrow IL$ and backward judgments consist of $IL \rightarrow CL$.

## 3 Detecting Inconsistencies in Interactive Judgments

In this section we define two types of inconsistencies using human judgments.

### 3.1 Inconsistencies with Model

When considering inconsistencies between a judgment and the model, we compare the contents of the combinations of labels (*CL*) to the individual label (*IL*), looking for inconsistencies. For example, if the combination of labels has no positive labels (S, PS) and the *IL* is positive, we classify this as inconsistent (Case 3). We enumerate the following cases which we define as inconsistent, summarizing each case in after the "//" symbols:

*For a judgment CL → IL or IL → CL over i ∈ I:*
*//there are no unknown labels in the CL, but the IL is unknown*
Case 1: *for all $v_j(i_j)$ in CL, $v_j \neq U$ and IL = U(i)*
*//there are no negative labels in the CL, but the IL is negative*
Case 2: *for all $v_j(i_j)$ in CL, ,$v_j \neq$ PD or D and IL = PD(i) or D(i)*
*//there are no positive labels in the CL, but the IL is positive*
Case 3: *for all $v_j(i_j)$ in CL, , $v_j \neq$ PS or S and IL = PS(i) or S(i)*
*//the CL is all positive or all negative, but the IL is a conflict*
Case 4: *for all $v_j(i_j)$ in CL, ($v_j$ = PS or S) or ($v_j$ = PD or D) and IL = C(i)*

In the forward case, the combination of labels can be said to represent evidence from the model, while the individual label is the user judgment. In the backward case, the individual label is the required evidence in the model, while a permissible combination of labels is the user judgment applied to the model structure.

### 3.2 Inconsistencies with Judgment History

When considering inconsistencies with between old and new judgments over the same intentions, we compare the combination of labels (*CL*) in the new and previous judgments, looking for cases when the combination of labels is the same, is clearly more positive, or more negative, using the ordering of labels from (1). We use this comparison to decide whether the new individual label (*IL*) is consistent with the old individual label. An example of the case is described in Section 1, when the combination of labels is equal, but the individual label is not. In another example, the user decides that with incoming labels of *PS(Security)* and *PD(Usability)*, Attract Users is *C(Attract Users)*. In the next round of evaluation, incoming labels may be *PS(Security)* and *C(Usability)*. The new combination of labels is more positive than the previous, as *C > PD*, so the individual label should not be less than the previous individual label, C, i.e. not *U*, *PD*, or *D*.

To aid in our definition of these cases we will refer to *ILnew* and *Cnew*, the most recent judgment for *i ∈ I*, and *ILprev* and *Cprev* , the previous judgments for *i*. We define psuedocode to check for these types of inconsistencies as follows:

*For a judgment CLnew → ILnew (backward: ILnew → CLnew) over i ∈ I:*
*For each previous judgment CLprev → ILprev over i ∈ I:*
*//compare labels in previous CLs to labels in new CL*
*For each $v_j(i_j)$ ∈ CLprev,*
*For $v_k(i_j)$ ∈ CLnew, compare $v_j(i_j)$ to $v_k(i_j)$*
*Classify as: >, =, or <*
*CLnew → ILnew is inconsistent with CLprev → ILprev if:*
*//The new CL is more positive, but the IL is more negative*
*All classifications are > or =, and ILnew < ILprev*
*//The new CL is more negative, but the IL is more positive*
*All classifications are < or =, and ILnew > ILprev*
*//The new and old CLs are identical, but the IL has changed*
*All classification are = (CLprev = CLnew), and ILnew ≠ ILprev*

## 4 Discussion, Conclusions and Future Work

This work reinforces the semantics of i* by embedding rules into the iterative analysis procedures which check for consistency amongst and between user judgments in the model. We have been very flexible and permissive in defining our judgments, only defining cases which are clearly inconsistent. For example, we could include rules to measure when a CL is mostly negative (many more negative labels than positive), and check that the IL is at least partially negative.

Although we have defined inconsistent judgment situations, we have not specified what actions to take when inconsistencies are found. In order to provide flexibility, we do not recommend preventing users from making inconsistent judgments, but instead suggest warning users, either when the judgment is made, or after the fact using a judgment consistency checker. This feature would work similarly to a built-in model syntax checker. Both the judgment consistency and model syntax checks are currently being implemented in the OpenOME tool [12]. The GMF meta-model of the tool has been expanded to include judgment and evaluation alternatives.

As we are aiming for model iteration, future work should adapt these checks to take frequent model changes into account. . Studies involving experienced and new i* users are needed to test the effectiveness of these checks in encouraging model iteration through interactive analysis.

## References

[1] J. Horkoff and E. Yu, "Interactive Analysis of Agent-Goal Models in Enterprise Modeling," International Journal of Information System Modeling and Design (IJISMD), vol. 1, 2010, pp. 1-23.

[2] J. Horkoff and E. Yu, "Finding Solutions in Goal Models: An Interactive Backward Reasoning Approach," 29th International Conference on Conceptual Modeling, Springer-Verlag New York Inc, 2010, p. 59.

[3] J. Horkoff and E. Yu, "Evaluating Goal Achievement in Enterprise Modeling – An Interactive Procedure and Experiences," The Practice of Enterprise Modeling, Springer, 2009, pp. 145-160.

[4] J. Horkoff and E. Yu, "A Qualitative, Interactive Evaluation Procedure for Goal- and Agent-Oriented Models," CAiSE□09 Forum, Vol-453, CEUR-WS.org, 2009, pp. 19-24.

[5] J. Horkoff and E. Yu, "Qualitative, Interactive, Backward Analysis of i* Models," 3rd International i* Workshop, CEUR-WS.org, 2008, pp. 4-46.

[6] J. Horkoff and E. Yu, "Interactive Goal Model Analysis Applied - Systematic Procedures versus Ad hoc Analysis," The Practice of Enterprise Modeling, 3rd IFIP WG8.1 (PoEM'10), 2010.

[7] E. Yu, "Towards modelling and reasoning support for early-phase requirements engineering," Proceedings of ISRE 97 3rd IEEE International Symposium on Requirements Engineering, vol. 97, 1997, pp. 226-235.

[8] i* Wiki," http://istarwiki.org", 2010.

[9] P. Giorgini, J. Mylopoulos, and R. Sebastiani, "Goal-oriented requirements analysis and reasoning in the Tropos methodology," Engineering Applications of Artificial Intelligence, vol. 18, 2005, pp. 159-171.

[10] J. Horkoff and E. Yu, "A Framework for Iterative, Interactive Analysis of Agent-Goal Models in Early Requirements Engineering," 4th International i* Workshop, submitted, 2010.

[11] OpenOME, an open-source requirements engineering tool, http://www.cs.toronto.edu/km/openome/", 2010.

# Towards a Declarative, Constraint-Oriented Semantics with a Generic Evaluation Algorithm for GRL

Hao Luo and Daniel Amyot

School of Computer Science and Electrical Engineering, University of Ottawa, Ottawa, Canada
haoluo.cn@gmail.com, damyot@eecs.uottawa.ca

**Abstract.** Goal models described with the Goal-oriented Requirement Language (GRL) are amenable to various kinds of analyses, including quantitative and qualitative propagations of satisfaction values. However, most approaches use bottom-up evaluations involving operational semantics that can only answer "what if" questions. This paper introduces a new declarative semantics for GRL based on a constraint-oriented interpretation of goal models. This semantics enables constraint solvers to evaluate and optimize goal models in a way that is more generic than bottom-up and top-down propagation techniques, hence enabling other questions to be answered. A prototype that combines the jUCMNav modeling tool and the JaCoP constraint solver to support quantitative evaluations is used to demonstrate the feasibility and potential of this new approach.

**Keywords.** Constraint-oriented goal evaluation, Goal-oriented Requirement Language, jUCMNav, semantics, User Requirements Notation.

## 1 Introduction

The User Requirements Notation (URN) is a language that combines scenario modeling (with Use Case Maps – UCM) and goal modeling (with the Goal-oriented Requirement Language – GRL) to support requirements engineering activities, especially for reactive systems and business processes. This standard language is defined with a metamodel, a concrete graphical syntax, and an XML-based interchange format [5].

In GRL, a model is composed of *intentional elements* (i.e., goals, softgoals, tasks, resources, and beliefs) connected by *links* (decomposition, contribution, and dependencies) and potentially allocated to *actors*. The URN standard includes a set of guidelines and requirements describing how to analyze GRL models based on a set of initial satisfaction values given to some of the intentional elements (i.e., an *evaluation strategy*), which are then propagated to the other intentional elements through the links connecting them. *Satisfactions* in GRL can be qualitative (satisfied, denied, etc.) or quantitative (integer in [–100..100]). Similarly, contributions can have a qualitative *weight* (make, break, etc.) or a quantitative one ([–100..100]). One particularity of GRL (which does not exist in *i**) is that intentional elements in an actor can have an *importance* factor that, when combined to satisfaction levels, helps measure the overall satisfaction of an actor (again, these measures can be qualitative or quantitative).

Although the standard does not impose specific propagation algorithms to evaluate a goal model against a given evaluation strategy, several algorithms (fully quantitative, fully qualitative, and hybrid) are suggested and are further explored in [1]. These algorithms are all automated, i.e., no interactivity is required to solve conflicts, and they are defined based on an operational semantics for GRL. However, these algorithms only support bottom-up propagation, which means that they can only answer "what-if" types of analysis questions. An evaluation strategy hence typically initializes some of the leaves in the goal graph, and the values are then propagated to the intentional elements higher in that graph.

Bottom-up analysis in limitative in practice; it is akin to testing in terms of analytical power, i.e., one often uses many strategies to find a good global trade-off or find an optimal satisfaction value for a given goal or actor (usually without any guarantee). There is a need to support top-down and inside-out analysis techniques for GRL in order to find solutions to more interesting questions such as "is there a way to reach this satisfaction level for this top-level goal?", or more generically "what is the maximum satisfaction of this goal given these constraints on other goals?".

This paper presents a new semantics and an algorithm for GRL model analysis that will enable modelers to answer all these types of questions. Tool support is also provided to demonstrate the feasibility of the approach.

## 2 Objectives of the Research

This research aims to support a generic and automatic propagation algorithm for GRL models that can support bottom-up, top-down, and inside-out analysis, as well as optimizations in the presence of constraints (i.e., intentional elements initialized anywhere in the model). To enable such an algorithm, we need to move away from the operational semantics often associated with GRL to a declarative semantics amenable to solving in the presence of constraints. Our approach is as follows: i) provide a declarative semantics for GRL (in this paper, we limit ourselves to quantitative evaluations only), ii) define a transformation of this semantics in terms of a constraint-oriented language for which automated solvers exist, and iii) prototype the approach.

The prototype algorithm is integrated to the jUCMNav tool as one of its GRL evaluation algorithms. jUCMNav is a free Eclipse-based URN modeling and analysis tool that already supports qualitative, quantitative, and hybrid evaluation mechanisms for GRL, but that only use bottom-up propagation [7]. jUCMNav offers an open architecture that makes it easy to add new GRL evaluation algorithms. The constraint-oriented technology we selected is JaCoP [6], a mature library that provides many modular constraint-solving and search mechanisms. JaCoP was selected because it is a Java open source project (like jUCMNav) and because it complies to the JSR-331 API for constraint programming, but others (e.g., Choco) could be used as well.

## 3 Scientific Contributions

This section briefly presents the three main contributions of our work.

### 3.1 Declarative Semantics for GRL

The proposed declarative semantics for GRL is expressed in mathematical terms. Intuitively, the syntactic domain is composed of intentional elements (with their types and importance factors), decomposition links (and their types: OR or AND), contribution links (and their weights), dependency links, actors, and a strategy, with the semantics and well-formedness rules expressed in the URN standard (e.g., an intentional element has only one type of decomposition). The semantic domain is one where each syntactically correct GRL diagram is interpreted as an evaluated set of intentional elements and actors, where evaluations are integers between –100 (denied) and 100 (satisfied), inclusively, as required by the URN standard [5]. The semantic function maps the syntactic constructs to evaluations in the semantic domains. $v(S)$ is used to denote the evaluation of intentional element $S$.

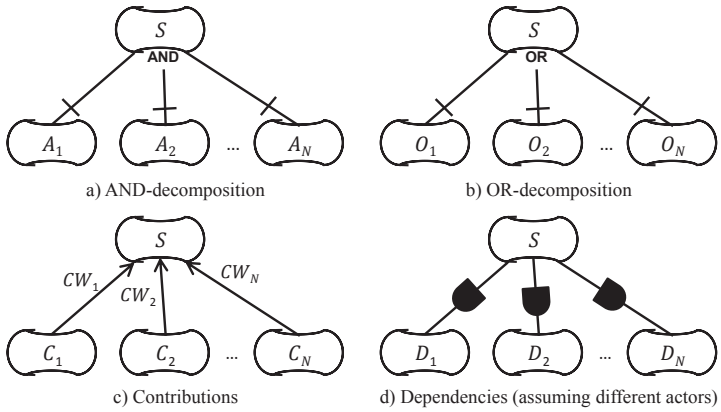



**Fig. 1.** GRL link types

Fig. 1 illustrates the four types of links we consider in this work. $S$, $A_x$, $O_x$, $C_x$, and $D_x$ are all intentional elements whereas $CW_x$ are contribution weights. In GRL, the relationships that exist between the sources and targets of these links are as follows.

- For the AND-decomposition (Fig. 1a), the satisfaction value of the parent $S$ is the minimum satisfaction value of its children: $v(S) = \min_{1 \le x \le N} v(A_x)$.
- For the OR-decomposition (Fig. 1b), the satisfaction value of the parent $S$ is the maximum satisfaction value of its children: $v(S) = \max_{1 \le x \le N} v(O_x)$.
- For contributions (Fig. 1c), the satisfaction value of the target $S$ is the weighted sum of the satisfaction values of its sources, bounded to [–100..100] (we abstract from the notion of tolerance factor discussed in [1]): $v(S) = \max(-100, \min(100, \sum_{x=1}^{N} v(C_x) \times CW_x \,/\, 100))$.
- For dependencies (Fig. 1d), the depender $S$ cannot be more satisfied than its dependees: $\bigwedge_{x=1}^{N} v(S) \le v(D_x)$.

When the same intentional element has multiple types of links, the semantics we provide is aligned with that of [1], i.e., decomposition values are considered as a basis to which contributions are added, and everything is constrained by dependencies. Hence, for AND-decompositions, we get the following general relationship:

$$v(S) = \max(-100, \min(100, \min_{1 \le x \le N} v(A_x) + \sum_{x=1}^{N} v(C_x) \times CW_x/100))$$

$$\wedge$$

$$\bigwedge_{x=1}^{N} v(S) \le v(D_x)$$

The relationship for the OR-decomposition is similar, but with max instead of min.

### 3.2 Transformation to a Constraint-Oriented Language

The JaCoP library comes with a set of functions and search mechanisms that can support an implementation of the semantics introduced in the previous section. Each intentional element in the GRL model becomes a unique variable in JaCoP, with [–100..100] as a domain. The equations and inequalities describing each of the intentional elements in terms of its links are converted to JaCoP constraints. For example, *B≤A* becomes `XlteqY(A,B)`, *C=min(A,B)* becomes `Min({A,B},C)`, and so on (some complex relationships need to be split into intermediate constraints along the way). Finally, the strategy definition provides constant values to the variables corresponding to the intentional elements initialized by that strategy.

We also experimented with the notion that GRL tasks that are leaf nodes in a model should only have 100 or 0 as possible satisfaction values, denoting the fact that these tasks are either performed/selected completely or not at all. It is trivial to detect such situations and to add corresponding JaCoP constraints.

JaCoP allows for different types of searches through the solution space to come up with a solution, when one exists. For example, one can maximize, minimize, or find median values for the variables. Heuristics are also available to accelerate the search. Multiple solutions can also be reported when more than one exists. However, refining the model while navigating through multiple solutions is left for future work.

### 3.3 Tool Prototype

Our proof-of-concept implementation integrates the JaCoP library to jUCMNav [9]. When the tool user selects a strategy to evaluate against the GRL model, the model and the strategy are converted on the fly to JaCoP constraints, and a search for a solution is started. We use by default a depth-first minimization search of the solution tree, and the first solution found by JaCoP is sent back to jUCMNav for display, with color coding. Fig. 2 illustrates interesting features of the algorithm and prototype tool. The intentional elements with a (*) and a dashed outline are initialized by the strategy.

Fig. 2a illustrates the traditional situation where we have the equivalent of a bottom-up propagation (leaves are initialized). Our algorithm can actually do everything that the quantitative algorithm in [1] can do (assuming a tolerance factor equal to 0). More interestingly, the tool will provide a solution to a situation where a root node is initialized, as shown in Fig. 2b. In this top-down-like situation, a minimal solution for B is found and returned by the JaCoP engine. In Fig. 2c, an additional constraint on C is added, and hence another minimal solution for B is returned. Note that in this generic case, any node can be initialized, not just roots or leaves (actually, the GRL model does not even need to be a tree; acyclic graphs are supported as well).

The absence of a solution can also be reported by JaCoP, and this is visualized in jUCMNav with a special value (–101, *conflict*) and blue color coding. For instance, in Fig. 2d, if C has a satisfaction level of –50, there is no solution for B such that A's satisfaction equals to 100. The impact of using tasks as leaf nodes (i.e., they can only have 0 or 100 as satisfaction values) is shown in Fig. 2e and Fig. 2f.

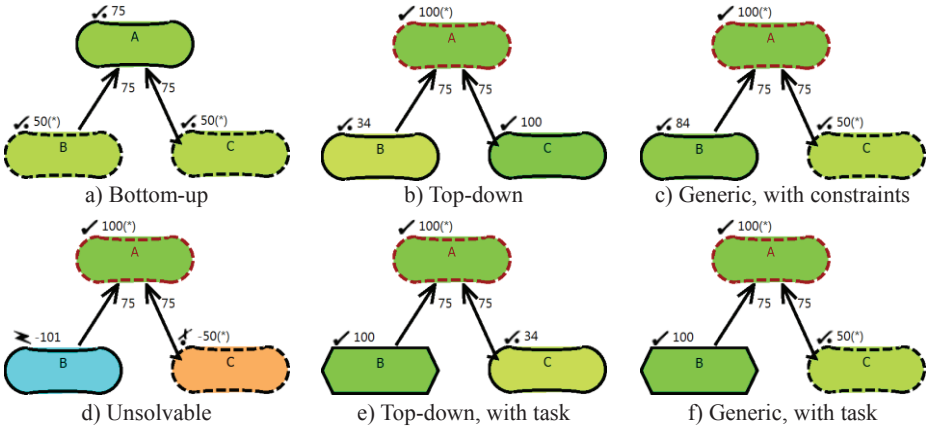



**Fig. 2.** Examples of GRL models solved with the generic algorithm

## 4 Conclusions and Related Work

This paper proposes a simple and yet powerful interpretation of GRL models through a declarative semantics amenable to transformations to constraint-oriented languages. The prototype implementation, which combines the jUCMNav modeling tool and the JaCoP constraint solver to support quantitative evaluations, demonstrates the feasibility and potential of the semantics and overall analysis approach.

Not only are the proposed semantics and quantitative propagation algorithm more generic than the existing ones for GRL [1,5], they also bring some benefits that could complement existing work in other goal languages. For example, Giorgini *et al.* [3] provide support for an automated top-down qualitative analysis for TROPOS models, with two values per intentional element (positive and negative evidence) but without conflict resolution nor quantitative values. Horkoff *et al.* [4] propose an interactive and iterative approach to find solutions in *i** models based on single evaluation values and a SAT solver. Their axiomatized qualitative propagation rules are useful for early requirements, conflict resolution, and the handling of multiple sources of weak evidence. However, a fully automated and quantitative approach like ours becomes quite beneficial when more precise and quantitative knowledge about the domain is available. Having the possibility to quickly set and evaluate constraints on intermediate nodes of a goal model is also useful during the exploration of the model (in a sense, this is another level of interactive analysis). In addition, running existing strategies when the model evolves (regression analysis) is more efficient in our context. Letier *et al.* [8] have extended KAOS goal models with probabilities for reasoning about

partial quantitative satisfaction in an automated way. However, we argue that the use of probabilities is another level of complexity and precision that few goal modelers can actually exploit in a pragmatic way, so our approach is likely more accessible.

## 5 Ongoing and Future Work

Extensions to this work include the support of remaining GRL concepts (e.g., XOR-decomposition, tolerance, and satisfaction at the actor level), but also GRL extensions like key performance indicators. We could also consider a global satisfaction for the entire model computed from actor satisfaction levels. Better formalization of the semantics, including an assessment of its soundness and completeness, are also required. There should also be a way to specify in a GRL strategy whether some goals should be maximized or minimized, and this could be taken into consideration by constraint solvers for an enhanced analysis experience. We also need to explore the usefulness of adapting this work to qualitative and hybrid evaluations. Although early performance results of our prototype are encouraging (most average-sized models we have played with can be solved within several milliseconds, hence enabling modelers to explore different values for a strategy on the fly), we often faced situations where a certain combination of model/strategy requires hours to solve. These situations need to be investigated, with a way to stop long evaluations. Finally, we believe that such declarative semantics will enable researchers to explore the usability of various semantics of goal modeling languages (in terms of complexity and customization) for different categories of users, application domains, and development phases.

**Acknowledgements**. We thank NSERC for financial support, and P. Heymans and F. Roucoux for useful discussions on the topics of GRL formalization and analysis.

## References

1. 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 (August 2010)
2. Amyot, D., Mussbacher, G.: User Requirements Notation: The First Ten Years, The Next Ten Years. Invited paper, Journal of Software (JSW), Vol. 6, No. 5, 747-768 (May 2011)
3. Giorgini, P., Mylopoulos, J., Sebastiani, R.: Goal-oriented requirements analysis and reasoning in the Tropos methodology. Eng. Appl. of AI, 18(2):159-171 (2005)
4. Horkoff, J., Yu, E.: Finding Solutions in Goal Models: An Interactive Backward Reasoning Approach. Conceptual Modeling – ER 2010. Springer, LNCS 6412:59-75 (2010)
5. ITU-T: User Requirements Notation (URN) – Language definition, ITU-T Recommendation Z.151 (11/08). Geneva, Switzerland (2008); http://www.itu.int/rec/T-REC-Z.151/en
6. JaCoP - Java Constraint Programming Solver, ver. 3.1 (2011); http://www.osolpro.com/
7. jUCMNav 4.4.0 (2011); http://softwareengineering.ca/jucmnav
8. Letier, E., van Lamsweerde, A.: Reasoning about partial goal satisfaction for requirements and design engineering. 12th Foundations of Software Eng. (FSE-12): 53-62, ACM (2004)
9. Luo, H.: Generic Propagation Algorithm for Goal Models. M.Sc. project, SITE, University of Ottawa, Canada (2011); http://www.UseCaseMaps.org/pub/

# A Flexible Approach for Validating *i** Models

Ralf Laue, Arian Storch

Chair of Applied Telematics / e-Business, University of Leipzig, Germany
laue@ebus.informatik.uni-leipzig.de

**Abstract.** In this article, we present a flexible approach to verify the syntactical correctness of $i^*$ models. We translate the information that is included in an $i^*$ model into a set of Prolog facts. Logical reasoning is applied for finding problems in a model.
Our validation method has been implemented in the *openOME* modelling tool. By using our validation add-on, modellers get feedback about problems and possible improvements of the model.

## 1 Introduction

Based on experiences from teaching the $i^*$ framework to students it has been reported that beginners often misunderstand the concept of some elements of the $i^*$ modelling language [1]. They would profit from a modelling tool that can not only locate problems resulting from such misunderstandings, but also gives feedback on correcting the model.

Furthermore, the $i^*$ framework is open to extensions and adaptions. For this reason, a modelling tool should support various modelling styles. It should be able to check whether a model conforms to a given style or to modelling conventions that exist in an organisation.

## 2 Objectives of the Research

The aim of our research is to provide a mechanism for validating $i^*$ models that has the following properties:

- The modeller gets an immediate feedback about possible problems.
- If a problem is detected in the model, the modeller has the possibility to learn about the reasons for the problem and about possibilities to model the intended meaning of the model in a correct way.
- Own validation rules can easily be added, for example based on company-wide style guidelines.
- The user is able to select the rules that should be applied in the validation.
- The validation procedure is flexible enough to allow an analysis of the textual model element labels.

## 3 Scientific Contributions

A modelling tool should assist its users when they have to create models that adhere to the restrictions of the modelling language. First and foremost, the model has to conform to the metamodel of the language. For the language $i^*$, several publications have suggested such a metamodel [2]. By enforcing metamodel conformance, an $i^*$ modelling tool can prevent syntactical modelling errors like an actor being modelled inside another actor. However, metamodel conformance does not yet mean that a model is correct with respect to the language definition. There are additional semantic constraints such as "There should not be a cycle made from actor association links".

For checking this type of constraints, the use of the Object Constraint Language (OCL) has been suggested in [3]. When we integrated validation support into the $i^*$ modelling tool *OpenOME*, we followed another approach that is based on logic programming: First, we translate the information that is contained in a model into a set of Prolog facts. Afterwards, logical reasoning is used for locating problems. For translating an $i^*$ model into the corresponding Prolog facts, we use a simple XSLT transformation. An $i^*$ model that needs to be validated is translated automatically into a set of Prolog facts by this XSLT transformation.

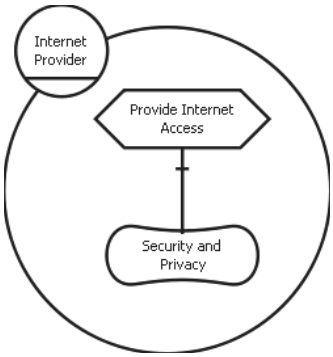



**Fig. 1.** the $i^*$ model...

```
role('_nKlm').
elementname('_nKlm', 'Internet Provider').
task('_LX3l').
elementname('_LX3l', 'Provide Internet Access').
contains('_nKlm', '_LX3l0').
softgoal('_xdXW').
elementname('_vgGY', 'Security and Privacy').
contains('_nKlm', '_vgGY').
anddecomposition('_LX3l0', '_xdXW').
shape('_nKlm', 2005, 822, 1096, 924).
shape('_LX3l', 190, 423, NaN, NaN).
shape('_xdXW', 1730, 1356, NaN, NaN).
```

**Fig. 2.** ...and the corresponding Prolog facts

Fig. 1 and 2 show an $i^*$ model fragment and the corresponding Prolog facts. For the sake of readability, we have shortened the unique identifiers that have been associated to each model element by Eclipse's XMI serialisation.

Once the information from the $i^*$ model is available as Prolog facts, it is very simple to locate problems. For example, the code for searching a goal that is wrongly connected to another goal using a means-ends link can be found by the query `goal(G1),me(G1,G2),goal(G2).` In general, means-ends links used wrongly (i.e. not linking a task to a goal) can be identified by the query `me(E,Partner),(not(task(E));not(goal(Partner)))`. It is also no problem

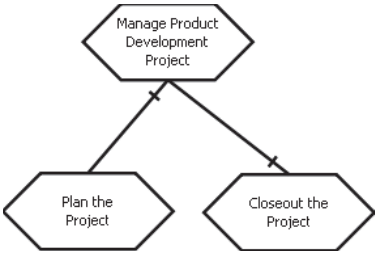



**Fig. 3.** Syntactical correct, but the layout of the tasks is misleading (source: $i^*$ wiki modelling guidelines)

to detect issues like the mentioned problem of cycles within actor dependency links.

Note that the size of position of the modelling elements are available as Prolog facts as well, allowing reasoning about the layout of the model. This allows to search for problems like the one depicted in Fig. 3. While the model is syntactically correct, the direction of one of the task decomposition links should be changed such that a task that is decomposed into a sub-task is always located above the sub-task. The possibility to deal with this kind of layout problems is an advantage of our approach compared to the OCL-based method described in [3]. Another advantage is that by exporting the model as Prolog facts, we have access to a variety of methods that can analyse the textual labels of model elements. This can be used for finding violations of naming conventions. In our current implementation, we use a heuristic method that analyses the labels and tries to detect two common types of errors: goals that have wrongly been modelled as tasks and softgoals that have been wrongly modelled as goals.

We have included all rules about semantic constraints that can be found in the $i^*$ wiki (http://istar.rwth-aachen.de) as well as some other validation rules described in the literature [3–5] or identified by our own analysis. It is possible to configure the subset of rules to be applied, which allows to use different styles for $i^*$ models. For all $i^*$ models validated so far (including rather large models with more than 100 elements), our validation mechanism delivered a result within one second. In order to prevent performance problems due to running the validation in background, we start the validation mechanism by executing it on demand by selecting a menu item. It should, however, be possible to use the validation mechanism as a background process without affecting the performance (as reported in [6] for a similar case).

Fig. 4 shows how detected problems are reported by the modelling tool: Each problem is shown as error or warning in the "Problems View" of the modelling tool. Also, a visual marker is added to the model element for which the problem has been found. If the modeller needs more information about the detected problem, it is possible to look up additional information about each problem class on a web site. We have already established links to the various guidelines available on the $i^*$ wiki. A novice user can quickly learn to draw correct $i^*$ models

by following the cycle "Model – Validate – Learn about the reported errors – Correct".

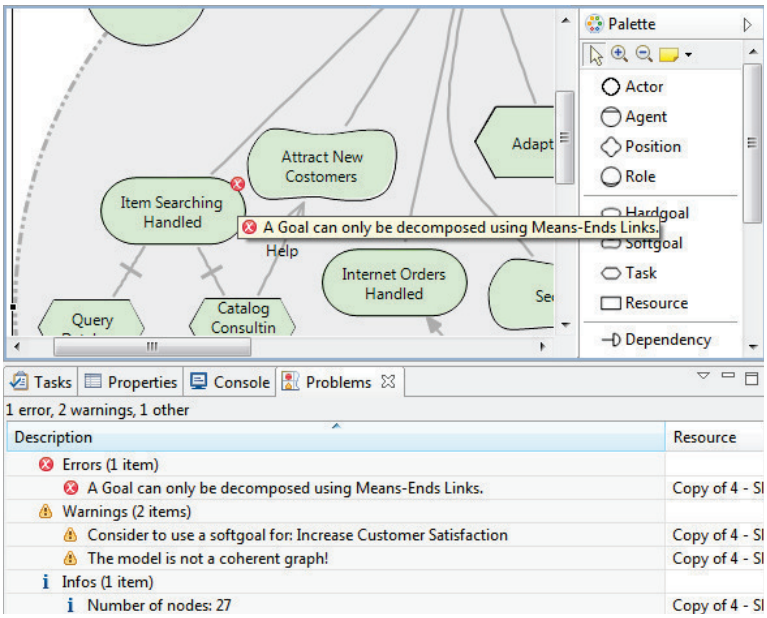



**Fig. 4.** Errors, Warnings and Information shown in *openOME*

## 4 Conclusions

The approach described in this paper has been implemented as a plugin into the Eclipse-based tool *openOME*. It is flexible enough to be used with every model editor that is based on Eclipse EMF/GMF. Previously, we have implemented the same validation mechanism into the business process model editor *bflow* Toolbox*. In a controlled experiment [7], we made the observation that providing an immediate feedback about modelling errors had a significant influence on model quality: The group provided with validation support made 6 errors in 7 models. For the control group which had to solve the same modelling task using the same tool without validation support we counted 24 errors in 6 models [8].

## 5 Ongoing and Future Work

We are interested in extending the approach described in this paper by analysing the texts used in the models more deeply. This would make it possible to check

for style rules such as "The infinitive of a verb together with an object has to be used for describing a task." Other interesting challenges would be to integrate the ideas of the Goal Clarification Method described in [9] and to develop recommendations for correcting a erroneous model.

Developers who are interested in using or improving our plug-in are invited to do so. More information about the plugins can be found at the Eclipse Modeling Toolbox project site `http://sourceforge.net/projects/eclipsemodeling`.

## References


1. Horkoff, J., Elahi, G., Abdulhadi, S., Yu, E.: Reflective analysis of the syntax and semantics of the i* framework. In: Advances in Conceptual Modeling, Challenges and Opportunities. Volume 5232 of Lecture Notes in Computer Science. Springer (2008) 249–260
2. Cares, C., Franch, X., Lopez, L., Marco, J.: Definition and uses of the i* metamodel. In: Proceedings of the 4th International i* Workshop, Hammamet, Tunisia. Volume 586 of CEUR Workshop Proceedings., CEUR-WS.org (2010) 20–25
3. Amyot, D., Yan, J.B.: Flexible verification of user-defined semantic constraints in modelling tools. In: Proceedings of the 2008 Conference of the Center for Advanced Studies on Collaborative Eesearch: Meeting of Minds. (2008) 7:81–7:95
4. Amyot, D., Horkoff, J., Gross, D., Mussbacher, G.: A lightweight GRL profile for i* modeling. In: Advances in Conceptual Modeling - Challenging Perspectives, ER 2009 Workshops. Volume 5833 of Lecture Notes in Computer Science., Springer (2009) 254–264
5. de Pádua Albuquerque Oliveira, A., do Prado Leite, J.C.S., Cysneiros, L.M.: Using i* meta modeling for verifying i* models. In: Proceedings of the 4th International i* Workshop, Hammamet, Tunisia. Volume 586 of CEUR Workshop Proceedings., CEUR-WS.org (2010) 76–80
6. Blanc, X., Mounier, I., Mougenot, A., Mens, T.: Detecting model inconsistency through operation-based model construction. In: ICSE '08: Proceedings of the 30th International Conference on Software engineering, New York, USA, ACM (2008) 511–520
7. Laue, R., Kühne, S., Gadatsch, A.: Evaluating the Effect of Feedback on Syntactic Errors for Novice Modellers. In: EPK 2009, Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten. CEUR Workshop Proceedings (2009)
8. Gruhn, V., Laue, R.: A heuristic method for detecting problems in business process models. Business Process Management Journal **16** (2010) 806–821
9. Jureta, I., Faulkner, S.: Clarifying goal models. In: Challenges in Conceptual Modelling. Tutorials, posters, panels and industrial contributions at the 26th International Conference on Conceptual Modeling - ER 2007. Volume 83 of CRPIT., Australian Computer Society (2007) 139–144

# Ontological Analysis of Means-End Links

Xavier Franch[1], Renata Guizzardi[2], Giancarlo Guizzardi[2], Lidia López[1]

[1]Software Engineering for Information Systems Research Group (GESSI)
Universitat Politècnica de Catalunya (UPC)
c/Jordi Girona, 1-3, 08034 Barcelona, Spain
{franch, llopez}@essi.upc.edu
[2]Ontology and Conceptual Modeling Research Group (NEMO)
Federal University of Espírito Santo
Av. Fernando Ferrari, S/N, 29060-970, Vitória/ES, Brazil
{rguizzardi, gguizzardi}@inf.ufes.br

**Abstract.** The *i** community has raised several main dialects and dozens of variations in the definition of the *i** language. Differences may be found related not just to the representation of new concepts but to the very core of the *i** language. In previous work we have tackled this issue mainly from a syntactic point of view, using metamodels and syntactic-based model interoperability frameworks. In this paper, we go one step beyond and consider the use of foundational ontologies in general, and UFO in particular, as a way to clarify the meaning of core *i** constructs and as the basis to propose a normative definition. We focus here on one of the most characteristics *i** constructs, namely means-end links.

**Keywords:** *i**, iStar, means-end links, fundational ontologies, UFO.

## 1 Introduction

Throughout the years, different research groups have proposed variations to the modelling language proposed in the *i** framework (from now on "the *i** language") [1][2]. Some come from paradigm shifts, others propose some particular type of new construct, and still others issue slight modifications related to the core constructs of the *i** language. This third type of variations mainly appear because the definition of the *i** language is neither fully detailed nor formal, and researchers may have interpreted the same constructs in different ways. The absence of a universally agreed metamodel has accentuated this effect [3].

There are two possible positions. One may argue that due to the social intention behind *i** modelling, a certain degree of freedom is convenient and then these slight changes should be acceptable. On the contrary, a more strict position is to consider necessary the existence of a shared body of knowledge on *i** with a well-defined meaning. In this paper, we align with the second option: our position is that those concepts that form the core of the *i** language shall be well-defined and agreed by the community. This is important to allow a uniform and consistent use of the language,

in a way that the community members are able to understand and communicate well through their models. Moreover, if it is a community objective that *i** gains industrial acceptance, then it is necessary to provide one interpretation of its core concepts.

There may be some discussion about where the boundaries of the *i** language core are, but some constructs like actors, goals and means-end links, to name a few, seem out of discussion. In this paper, we are interested in the analysis of means-end links. Therefore, we search for a proposal that may serve as basis for a community agreement about what a means-end link exactly is. Agreement shall be first at the syntactic level by referencing to some *i** metamodel (e.g., which type of intentional element may be involved in a particular context). But syntax is not enough, still different modelers may interpret syntactic-equivalent models in different ways. We need a deeper understanding of the meaning of means-end link. In this paper, we propose the use of the UFO foundational ontology [4].

With respect to the analysis and (re)design of conceptual modeling languages (i.e. the focus of this particular paper), we must understand ontology as in conceptual modeling, i.e. as a theoretical body of knowledge or foundation (that is why we call it foundational ontology). We propose the use of a foundational ontology as a reference model to which the concepts of the metamodel of an existing modeling language must be mapped into. This approach enables the evaluation of the modeling language, which can further lead to re-design. In other words, UFO is employed here as a well-founded basis for (1) making explicit the ontological commitments of each modeling language; (2) defining (ontological) real-world semantics for their underlying concepts; and (3) providing guidelines for the correct use of these concepts. The adequacy of this ontology for our purposes lies on the fact that UFO includes the concepts that are relevant (i.e. may serve as interpretation) to analyse the *i** framework. In fact, previous works have already analysed some of the concepts of Tropos with the purpose of mapping its metamodel into the metamodel of other modeling languages, namely AORML [5] and ARIS [6]. More about how UFO has been developed may be found in [4][7][8].

## 2 Objectives of the Research

The general objective of our long-term joint research is to provide an ontological foundation to the *i** language core. In this paper, as a proof-of-concept, we focus on one particular *i** construct that lies in the heart of the language: the means-end link (ME-link). Even thought it may seem that it is not a much controversial construct, it already poses some questions that have not a clear response. Thus, it is a good case as proof of concept of our work. The general objective of understanding the **meaning of a ME-link** can be refined into a series of more concrete research questions:

- Which *i** intentional elements may appear as means and as end of an ME-link?
- Which ontological properties must these elements fulfil?
- Which conditions must be verified to establish an ME-link among two elements?
- Which relationships do exist with other types of links?
- Which are the implications of an ME-link established in one actor A, over another actor B, such that an actor link from B to A is established?

# 3 Scientific Contributions

## 3.1 State of the Art on the use of Means-end Links

Different *i** variants/approaches/dialects [9], whilst respecting the main idea of the ME-link as "a means to attain an end", state their own (if any) restrictions (see Table 1). In his seminal proposal, Yu stated about ME-links: "the end can be a goal, task, resource or softgoal, whereas the means is *usually* a task". The term "usually" was dropped in the evolution of this variant, the *i** wiki version, where not just the means were completely restricted to tasks (i.e., not *usually* but *always*) but also the end was restricted to goals. The guidelines in the wiki state that other types of links are available for different combinations of intentional element's types. Whilst being clear in terms of what types are permitted, this definition is very restrictive.

Another issue to remark is the relationship to the concept of "OR-decomposition". The concept of ME-link is close to that of OR-decomposition, in the sense that the source elements of the link (either "means" or "sub-elements") are interpreted in a kind of logical OR relationship with respect to the target. In fact, several proposals seem to not distinguish these two concepts, e.g. GRL has eliminated means-end links and just OR-decomposition (in addition to AND- and XOR-decomposition) is offered. On the contrary, some Tropos definitions are using both constructs in its language, like in the following quotations: "in an OR decomposition the sub-goals represent alternative ways to achieve the root goal", and "the means/end relationship specifies a means (in terms of a goal, a plan or a resource) to satisfy the goal"; whilst others Tropos' papers are closer to the classical Yu's proposal or even whilst providing both constructs, it is not clear which is the real difference.

It is also worth to remark that existing works do not seem to address much the relationship between ME-links and actor links. In our ongoing work analysing the meaning of the is-a actor link (follow-up of our previous work in [10]), we have mainly investigated if new means can be added to an end that is inherited in a subactor. To do so, a question need to be answered: is a means-end relation always complete, is it always incomplete, or if none of the former, is it possible to distinguish an incomplete relation from a complete one?

Last, a final issue is whether the means for an end are to be considered exclusive or not (XOR vs. OR). For instance, Yu's thesis does not explicitly state this interpretation, but from the examples the means seem to be always exclusive. Just one analysed proposal (GRL) allows explicitly declaring the type of logical operator, but we remind that GRL is not distinguishing among ME-links and OR-/XOR-decompositions.

| Approach | Link | Intentional types (means->end) | OR/XOR/not stated |
|---|---|---|---|
| Yu's thesis | Means-end link | (usually) T -> G \| SG \| T \| R | Not stated |
| *i** wiki | Means-end link | T -> G | Not stated |
| Tropos | Means-end link | G \| T \| R -> G | Not stated |
| | OR-decomposition | G -> G | Not stated |
| GRL | OR-decomposition | G \| SG \| T \| R -> G \| SG \| T \| R | Explicitly declared |

Figure 1. Summary of the state of the art.

### 3.2 Means-end Links from an Ontological Point of View

Before analysing *i** ME-links, it is important to provide an ontological view of the language intentional elements. Reading this section requires some basic knowledge about UFO [4][7].

We interpret the *i* Agent* and *Role* as the concepts of Agent and Social Role in UFO (respectively). *Position* is also interpreted as a complex Social Role, since this *i** concept is defined solely with the purpose of aggregating different roles. The abstract *Actor* concept only captures general relations between *Agent*, *Roles*, *Positions* and other modeling elements and, thus, it has no specific interpretation in itself.

We interpret *i* Goals* as Goals in UFO. Goals in UFO are sets of intended states of affairs of an agent. The relationship between an *Agent* in *i** and a *Goal* is interpreted indirectly by making use of the concept of Intention (or Internal Commitment) in UFO, which is a Mental Moment of an Agent. As previously discussed, UFO contemplates a relation between Situations and Goals such that a Situation (or possibly a number of Situations) may satisfy a Goal. In other words, since a Goal is a proposition (the propositional content of an Intention), we have that a particular state of affairs can be the truthmaker of that proposition.

The concept of *Softgoal* does not have a uniform treatment in the *i** community. Sometimes, softgoals are taken to represent non-functional requirements. In other places, a softgoal is considered as a fuzzy proposition, i.e., one which can be partially satisfied (or satisfied to a certain degree, or yet, *satisficed*) by Situations [5]. We here take a different stance, namely, that a *Softgoal* is one "subjective to interpretation" and "context-specific". Hence, for softgoals, it seems to be impossible to eliminate a judging agent (collective or individual) from the loop. Thus, instead of considering in the ontology a new satisfices relation between Situation and Goal which perhaps should contemplate a fuzzy threshold of satisfaction, we take a different approach. We consider the relation of satisfaction as a ternary relation that can hold between an Agent, a Goal and Situation. An instance of this relation is derived from the belief of an agent that a particular situation satisfies the goal at hand. Now, in this view, different agents can have different beliefs about which sets of situations satisfy a given goal. In fact, it is exactly this criterion which seems to capture the aforementioned notion of softgoals and its differentiae w.r.t. hardgoals: (i) a goal G is said to be a hardgoal iff the set of situations that satisfy that goal is necessarily shared by all rational agents; (ii) a goal G is said to be a softgoal iff it is possible that two rational agents X and Y differ in their beliefs to which situations satisfy that goal.

The mapping of the *Task* concept from *i** to some UFO concept is established in a direct manner. *Task* in *i** is a specific way of doing something to satisfy some *Goal* (or satisfacing some *Softgoal*). From the UFO ontology, we have that an Action (instance of an Action Universal) is an intentional event performed by agents with the purpose of achieving goals. Consequently, the *i* Task* construct can be interpreted as an Action Universal.

The concept of *Resource* has been interpreted as a Resource in UFO, i.e., as a Non-agentive Substantial (or Object) which participates in a Complex Action. A Complex Action is a composition of at least two basic Actions or Participations. Participations can themselves be intentional (i.e., Actions) or non-intentional Events.

### 3.2.1. Ontological Analysis of the *i** Means-end Relation

We first analyse *ME-links* between a *Task* and a *Goal*. In *i**, the *ME-link* is a ternary relation indexed to an *Agent*'s (subjective) point of view. However, as stated above, we consider in UFO that all agents have a consensual opinion regarding the satisfaction of a hardgoal. In this sense, we can exclude the agent's point of view from the definition of this *ME-link*, simply interpreting it as: a Task T is a means to a Goal G (G being the end) iff one or more executions of T (i.e., action instances of type T) produce a post-situation which satisfies G.

Similarly, the *ME-link* can also be defined between a *Resource* type and a *Task*, or between a *Resource* type and a *Goal*. The former mode of this relation can be interpreted as follows: a Resource type R is a means to a Goal G (G being end) iff every Action which satisfies that Goal has as part a Participation of a Resource of that type. In contrast, the *ME-link* between a *Resource* and a *Task* should be interpreted as: a Resource type is a means to a Task (end) iff every Action instance of that Task has as part a Participation of a Resource of that type.

In the case of a *Softgoal*, we should in fact consider the perspective (i.e. belief) of a particular agent, since its satisfaction set is not a consensus. Thus, we define the ME-link between a *Task* and a *Softgoal* as follows: a Task T is a means to a Softgoal S (S being the end) in the point of view of Agent A iff one or more executions of T produce a post-situation which A believes to satisfy S.

### 3.2.2. Resolving the Dispute Regarding Means-end and OR-decomposition

As already mentioned in section 3.1, the *i** dialects do not agree on the use of the ME-link and the OR-decomposition. Having two distinct relations that model the same phenomenon in the world is usually not a good practice, because, in general, the modeler will attempt to ascribe different meanings to each of them, or else randomly chose one or another when modeling such phenomenon. Thus, we believe it is in the interest of the *i** community to reach an agreement, having one uniform view in this regard. In this paper, instead of taking a stand, we prefer to present two possibilities to be debated with other *i** community members.

A first option (Fig. 2, left) is having only one relation, namely the ME-link, excluding the OR-decomposition from the *i** core. This option favors a cleaner language, which may lead to a simpler use of the framework. In this case, the ontological definitions in section 3.2.2 are applied, along with the following: i) Goal G2 is a means to a Goal G1 iff satisfying G2 produces a post-situation that satisfies G1, and ii) Task T2 is a means to a Task T1 iff executing an Action instance of T2 causes the execution of an Action instance of T1.

A second option (Fig. 2, right) is having both ME-link and OR-decomposition relations as part of the *i** core. Although leading to a more complex language, one can argue that this may add expressivity. In this case, the ontological definitions in section 3.2.1 are applied if we consider means and ends of **distinct types**. Among intentional elements of the **same type** we consider OR-decomposition only, e.g. an OR-decomposition of goal $G_0$ into subgoals $G_1 \ldots G_n$ should be interpreted as: $(G_0 \leftrightarrow (G_1 \vee G_2 \vee \ldots \vee G_n))$. Thus, these relations reflect logical relations between propositions.

| | | End | | | |
|---|---|---|---|---|---|
| Means | | T | R | G | SG |
| | T | ME | --- | ME | --- |
| | R | ME | --- | ME | --- |
| | G | --- | --- | ME | --- |
| | SG | ME | ME | --- | --- |

| | End | | | |
|---|---|---|---|---|
| | T | R | G | SG |
| T | Or-D | --- | ME | --- |
| R | ME | --- | ME | --- |
| G | --- | --- | Or-D | --- |
| SG | --- | --- | --- | Or-D |

Figure 2. Two tables summaring the two options considered in this section.

## 4 Conclusions

In this paper, the ontological underpinnings of the means-end link have been analysed and two different positions for further discussion, identified. We think that this is a first step of a long-term goal about providing an ontological foundation to the *i** language core with the aim of obtaining a shared definition of that core fostering standardization and thus eventually boosting industrial adoption.

## Acknowledgements

This work was partially supported by the Spanish project TIN2010-19130-C02-01. We are also grateful to the support provided by FAPES (Grant #45444080/09) and CNPq (Grant numbers 481906/2009-6 and 309382/2008).

## References

1. Franch, X., Maté, A., Trujillo, J.C., Cares, C.: On the joint use of *i** with other Modelling Frameworks: a Vision Paper. Accepted for publication in IEEE RE 2011.
2. Cares, C., Franch, X.: A Metamodelling Approach for *i** Model Translations. CAiSE 2011.
3. Franch X.: Fostering the Adoption of *i** by Practitioners: Some Challenges and Research Directions. In *Intentional Perspectives on Information Systems Engineering*, 2010.
4. Guizzardi, G.: *Ontological Foundations for Structural Conceptual Models*. Ph.D. Thesis, CTIT PhD-thesis, University of Twente, The Netherlands, 2005.
5. Guizzardi, R.S.S., Guizzardi, G.: Ontology-based Transformation Framework from Tropos to AORML. In *Social Modeling for Requirements Engineering*, The MIT Press, 2011.
6. Cardoso, E.C.S., Santos Jr., P., Almeida, J.P.A., Guizzardi, R.S.S., Guizzardi, G.: Semantic Integration of Goal and Business Process Modeling. CONFENIS 2010.
7. Guizzardi, G., Falbo, R. A., Guizzardi, R. S. S.: Grounding Software Domain Ontologies in the Unified Foundational Ontology (UFO): The case of the ODE Software Process Ontology. IDEAS 2008.
8. Guizzardi, G., Wagner, G.: On A Unified Foundational Ontology and some Applications of it in Business Modeling. EMOI-INTEROP 2004.
9. López, L., Franch, X., Marco, J.: Making Explicit some Implicit Language Decisions in the *i** Framework. Accepted for publication in ER 2011.
10. López, L., Franch, X., Marco, J.: Defining Inheritance in *i** at the Level of SR Intentional Elements. iStar 2008.

# Supporting i* model integration through an ontology-based approach

Karen Najera[1,2], Anna Perini[2], Alicia Martinez[1], and Hugo Estrada[1]

[1] National Center of Research and Technological Development - CENIDET
Interior Internado Palmira S/N, 62490 Cuernavaca, Morelos, Mexico
{karen.najera, amartinez, hestrada}@cenidet.edu.mx
[2] Fondazione Bruno Kessler - IRST, Center for Information Technology (CIT)
Via Sommarive, 18, 38050 Trento, Italy
{knhernandez, perini}@fbk.eu

**Abstract.** The i* framework is widely used for organizational modeling. It has been applied in different application domains, hence many i* variants have been proposed. Sharing information and integration of models expressed in i* variants imply interoperability problems. The interoperability has been approached at different levels, e.g. through unified metamodels, or with an interchange format for representing i* models. As a preliminary step toward addressing the interoperability problem, our aim in this paper is to investigate the role of an ontology-based metamodel to realize integration of models expressed in i* variants, bringing the advantages of ontologies to the organizational modeling domain. We describe the ontology-based metamodel of the i* framework and the process followed to build it, exploiting Model Driven Engineering ideas. Moreover, we describe a first application of our approach to define a tool-supported process aiming at generating ontologies corresponding to models expressed in the i* language.

**Keywords:** i*, conceptual modeling, ontologies, model-driven engineering, model transformations.

## 1 Introduction

The i* framework [10] is a well known organizational modeling technique, that inspired several studies and extensions. It uses strategic relationships to model the social and intentional context of an organization. Nowadays, many research projects use the i* framework in different application domains, hence many i* variants have been proposed, such as Tropos [6], Service-oriented i* [4] and so on.

Since models are created with particular variants, sharing information and integrating models expressed in different i* variants imply interoperability problems. The interoperability has been approached at different levels, e.g. through the definition of a unified metamodel(e.g. [2]) and [8]), or with the introduction of interchange format for representing i* models such as iStarML [3]. As a preliminary step toward addressing the interoperability problem, our aim in this

paper is to investigate the role of an ontology-based metamodel to realize integration of models expressed in i* variants, bringing the advantages of ontologies to the organizational modeling domain.

Recent literature [9], [7] put in relationship ontologies and the layered architecture used in the Model Driven Engineering (MDE) approach (where models, metamodels and metametamodels correspond to the M1, M2 and M3 layers, respectively) with the purpose of bridging models and metamodels with ontologies. The authors specify the advantages of using ontologies, namely: ontology linking service, where models and metamodels are transformed in terms of ontologies to improve interoperability; querying, automated reasoning and others. In our work, in addition to model integration, we aim at providing a solution, which permits bringing ontologies advantages to the organizational modeling domain. In this research work we have developed the metaontology of the i* framework. It has been built using the standard semantic web language OWL [1] exploiting MDE ideas. As a first application of our approach, we described a tool-supported process aiming at generating ontologies corresponding to models expressed in the i* language.

## 2 Objectives

The main objective of this research work is to propitiate the integrability of models represented in the i* modeling language or represented in any of its variants through the use of ontologies. For the accomplishment of this main objective we have identified three specific objectives: the first corresponds to the development of a metaontology (which we have called OntoiStar) for representing the i* metamodel; the second corresponds to the development of a methodology for guiding the process of integrating additional concepts of i* variants into OntoiStar for improving the interoperability; and the last one is related to the use of OntoiStar as the underlying baseline for the automatic transformation of a model represented in any i* variant into ontologies derived from the concepts of OntoiStar. At present we have addressed the first objective and partially the third one which will be totally covered after we achieve the second objective (since now we have only implemented the automatic transformation from i* models into OntoiStar).

## 3 Scientific contributions

We aim to show that the use of ontologies is an appropriate way for supporting integrability of models expressed in i* variants and a promising approach towards tackling the i* variants interoperability problem. For that reason, our scientific contributions addressed so far are related with a semi-automated approach to generate organizational ontologies from an i* model (both the strategic dependency and the strategic rationale model). We first developed the metaontology OntoiStar which corresponds to the ontological representation of the i* metamodel; and then we developed a tool to automatically transform an i* model to

instances of OntoiStar. OntoiStar has been built using the OWL language [1], as it is the standard semantic web language, the organizational knowledge can be shared to be understandable not only for humans but also for software system to automatically discover the meaning of business resources defined in the models. OWL allows to define axioms in OntoiStar for defining the semantic of each i* variant and the definition of syntactic constraints. Therefore it is possible to analyze the syntactic correctness of i* models. Moreover, OWL supports inference rules which we will apply for avoiding the loss of information caused by differences in the integrated i* variants.

The constructs included in the i* metamodel were taken from two i* metamodel proposals [2] [8], which consist of mainly the common constructs of the i* variants. We developed OntoiStar based on those common constructs and selected characteristics described below of each proposal. We have applied a transformation language bridge approach [9] based on MDE. MDE is a software development methodology that recognizes models as a key role for describing the system to be developed. It is based on layered architectures, where models, metamodels and metametamodels correspond to the M1, M2 and M3 layers, respectively. In Fig. 1 we present the two layered architecture of this approach. On the left side, the i* modeling language architecture, where the i* metamodel is located in the M2 layer, and it is described by its metametamodel (represented in the Unified Modeling Language) in the M3 layer, and on the right side, our proposed ontology architecture, where the resultant OntoiStar has been located in the M2 layer and it is described by the OWL metamodel. The transformation bridge then is defined in the M3 layer. It contains the mapping rules between concepts from the i* metametamodel, such classes and associations and concepts from the OWL metamodel, such classes and properties. The transformation bridge is applied in the layer M2, transforming the i* metamodel into the metaontology: OntoiStar. For transforming an i* model to instances of OntoiStar (in the layer M1), we propose an automatic transformation tool.

Applying this approach we generate a logical knowledge base, where the terminological part is provided by the metaontology OntoiStar and the assertional part corresponds to a specific organization description represented in an i* model, which is mapped as instances of ontoiStar.

The transformation bridge is defined as follows:

**(1) Identifying constructs from the i* metamodel and from the OWL language**.

1. We adopted from the metamodels presented in [2] and [8] the common characteristics, including concepts, relations and attributes. The specific characteristics adopted for each one are described in Table 1.
2. The main constructs of the OWL language are: Class, Object property and Data property and the axioms: ObjectPropertyDomain, ObjectPropertyRange and DataPropertyDomain.

**(2) Defining the relationships between constructs from the i* metamodel and the OWL language**. Based on this definition, we proposed the following transformation rules:

**Table 1.** Specific characteristics adopted from i* metamodels

| Reference model for i* [2] | Unified metamodel for i* [8] |
|---|---|
| Dependum class. | All concept relationships representation as classes and associations. |
| Attributes: Label and Type, from Node class and IntentionalElement class respectively. | The high abstraction level class iStarRelationship. |
| ContributionType enumeration types: '+' and '-'. | The iStarRelationship subclasses: ActorRelationship, DependencyRelationship and InternalElementRelationship. |
| Class properties (except the disjoint property between DependableNode and IntentionalElement). | |

1. Each concept, concept relationship and enumeration class included in the i* metamodel is represented as a class in OWL.
2. Each association included in the i* metamodel is represented as an object property in OWL. Where its domain corresponds to the association source and its range corresponds to the association target.
3. Each class property included in the i* metamodel is represented with axioms in OWL.
4. Each enumeration element included in the i* metamodel is represented as a class instance of the owner enumeration class in OWL.
5. Attributes. There are two types of attributes:
   (a) Each enumeration type attribute included in the i* metamodel is represented as an object property in OWL. Where its domain corresponds to the owner class and its range corresponds to the enumeration class.

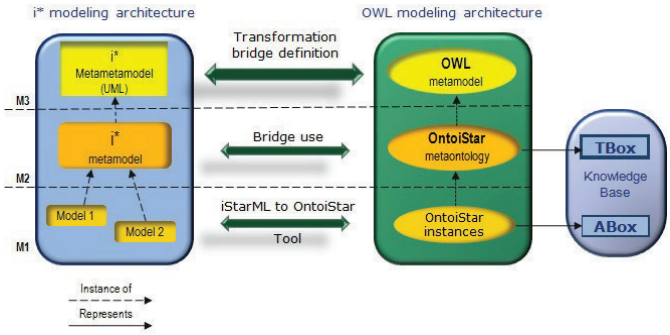



**Fig. 1.** Transformation bridge


**Fig. 2.** Fragment of the OntoiStar taxonomy

(b) Each primitive data type attribute included in the i* metamodel is represented as a data property in OWL. Where its domain corresponds to the owner class and its range corresponds to the primitive data type.

We implemented OntoiStar using the ontology editor and knowledge-base framework Protègè [5]. A Fragment of the OntoiStar taxonomy is presented in Fig. 2.

### 3.1 Generating ontologies corresponding to i* models

A software tool has been developed where given the i* model represented in iStarML format and the metaontology OntoiStar, the tool populates the metaontology with instances of i* elements belonging to a specific organizational model. For doing that, mapping rules between the iStarML elements (tags and attributes) and the concepts in OntoiStar (classes and properties) have been defined. The tool overview is presented in Fig. 3. The first step is the development of the i* model which can be realized with i* modelers or editors that enables producing a model specified in the iStarML format. The automatic transformation process consists of parsing the iStarML file and, according to the defined mapping rules implemented in the tool *iStarMLtoOntoiStar*, instantiating the corresponding classes and properties in the metaontology OntoiStar. The output of the tool contains the metaontology OntoiStar with instances representing the knowledge content in the i* model. It could be edited with an ontology editor or it could be the input of an ontology based application.

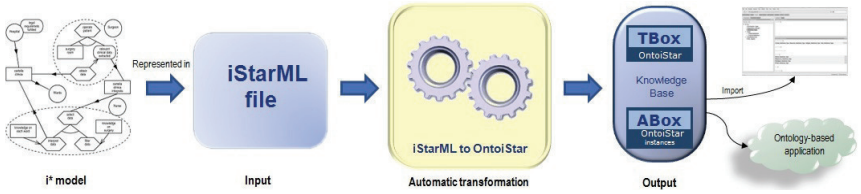



**Fig. 3.** Tool overview

## 4 Conclusion

In this paper we presented a semi-automated approach to generate organizational ontologies from an i* model. Specifically we described how we developed the metaontology of i* modeling language (OntoiStar) and the transformation process that we have applied to derive from i* models their corresponding OWL ontologies. As the basis of our future work, OntoiStar contains the common and relevant characteristics of i* variants obtained from two i* metamodel proposals.

## 5 Ongoing and future work

Currently, we are addressing the integration of i* variants to OntoiStar. The methodology is applied to the i* variants: Tropos and Service oriented i*. Moreover, we are also extending the tool for covering the automatic transformation for models of those variants. In the future the methodology and the automatic transformation will be applied to more i* variants. Towards addressing the interoperability problem in a more systematic way, we will investigate the transformation of the knowledge contained in the metaontology OntoiStar to any of the i* variants already integrated in OntoiStar. We plan the use of inference rules for avoiding the loss of information caused by differences in the i* variants. Empirical evaluation of the effectiveness of ontologies to solve the interoperability between i* variants will follow.

## References

1. S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. OWL Web Ontology Language Reference. Technical report, W3C, http://www.w3.org/TR/owl-ref/, February 2004.
2. C. Cares, X. Franch, E. Mayol, and C. Quer. A Reference Model for i*. In *Social Modeling for Requirements Engineering*, pages 573–606. MIT Press, 2010.
3. C. Cares, X. Franch, A. Perini, and A. Susi. Towards interoperability of i* models using istarml. *Computer Standards & Interfaces*, 33(1):69–79, 2011.
4. H. Estrada. *A service-oriented approach for the i* framework*. PhD thesis, Valencia University of Technology, Valencia, Spain, 2008.
5. J. H. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubézy, H. Eriksson, and N. F. Noy. The evolution of Protégé: an environment for knowledge-based systems development. *Int. J. Hum.-Comput. Stud.*, 58(1):89–123, 2003.
6. P. Giorgini, J. Mylopoulos, A. Perini, and A. Susi. The Tropos Methodology and Software Development Environment. In *Social Modeling for Requirements Engineering*, pages 405–423. MIT Press, 2010.
7. B. Henderson-Sellers. Bridging metamodels and ontologies in software engineering. *Journal of Systems and Software*, 84(2):301–313, 2011.
8. M. Lucena, E. Santos, C. T. L. L. Silva, F. M. R. Alencar, M. J. Silva, and J. Castro. Towards a unified metamodel for i*. In *Research Challenges in Information Science*, pages 237–246, 2008.
9. S. Staab, T. Walter, G. Gröner, and F. S. Parreiras. Model driven engineering with ontology technologies. In *Reasoning Web*, pages 62–98, 2010.
10. E. S.-K. Yu. *Modelling strategic relationships for process reengineering*. PhD thesis, University of Toronto, Toronto, Ont., Canada, 1996.

# The mysteries of goal decomposition

Scott Munro[1] Sotirios Liaskos[2] Jorge Aranda[3]

[1] Department of Philosophy, York University
Toronto, Canada
scmunro@yorku.ca
[2] School of Information Technology, York University
Toronto, Canada
liaskos@yorku.ca
[3] Department of Computer Science, University of Victoria
Victoria, Canada
jaranda@uvic.ca

**Abstract.** Goal decomposition structures lie at the heart of goal modeling languages such as *i**. High-level goals of stakeholders are recursively decomposed into lower level ones and eventually into leaf level tasks to be performed by agents. The decomposition structure can also develop through a bottom up approach whereby higher-level goals are introduced as justifications for existing low-level ones. The very concept of decomposition, however, both as process and as artefact is rarely questioned in requirements engineering. In this paper, we argue that it may be of value to give a closer look into goal decomposition and clarify what we actually know about it and what is yet to be understood. We report on an on-going effort to identify empirical work on decomposition coming from various research fields, hoping to find such evidence. We then pose some research questions that we believe need to be pursued in order to improve our understanding of goal decomposition.

**Key words:** requirements engineering, goal modeling, i-star

## 1 Introduction

Goal decomposition is central in goal modeling. High-level goals of stakeholders are recursively decomposed into lower level ones and eventually into leaf level actions, a subset of which is potentially to be performed by a machine agent. The reverse bottom-up process of discovery of high-level goals as explanations of existing tasks complements the top-down one. The result, the goal decomposition model, is a central piece of a goal model such as an *i** strategic rationale diagram. The decomposition structure serves the purpose of connecting stakeholder desires with system functions and has been found to serve many benefits [14].

But where do decompositions come from and how exactly are they used? While the literature offers a wealth of case studies illustrating the benefits of using goal models, proposing general processes for developing models, or discussing meta-models and ontological treatments, little seems to be known about

goal decomposition as a cognitive process. If we knew more about the nature of decompositions we would be able to allow for more natural and systematic ways of identifying and using such, increasing the quality and usefulness of goal models such as *i** models.

In this paper, inspired from some early experimental results on the matter of goal decomposition, we describe some highlights from our on-going work to understand what the literature in a variety of fields has to say about decomposition. We then describe the particular questions we are hoping an empirical research program on decomposition could attempt to answer.

The presentation is organized as follows. We provide an overview of our research goals and questions (Section 2), present some highlights from the literature we have been studying (Section 3) and then describe what we have learned and how we intend to proceed (Sections 4 and 5).

## 2 Objectives of Research

Central to the emergence of the goal-oriented paradigm is the fact that it offers a clean approach to connect the problem domain with the solution domain through recursive decompositions of expressions of the former (goals) into expressions of the later (actions). The literature proposes several approaches and techniques for developing such decompositions. For example, KAOS offers AND-decomposition patterns based on temporal semantics of the goals to be decomposed [3], while Rolland et al. [12] propose scenario-goal coupling to guide decomposition, an idea also used in Liaskos et al. later [8]. More general model development and enrichment methods, by e.g. informing development from other sorts of models have been proposed, e.g. [6].

However, although in abstract terms we can think of a decomposition task as a complete and sufficient activity to manage goals in order to make them more easy to tackle, in practical terms, we have people doing the decomposing and people that need to judge whether the decomposed goals fit the higher-level goal exactly - i.e. when a decomposition is "good". Thus, the result of goal decomposition activities is inextricably linked to cognitive and psychological considerations and is influenced by the mental process that is followed (or lack thereof). Our experience in developing goal models, indeed suggests that not only different approaches and different people produce different decomposition models, but that understanding what the ideal model for a particular situation is can be very puzzling.

Our early results from a pilot experimental study were illuminating as to how different modelers, when given the same exactly input produce different decompositions. In this study we invited a number of senior students of Information Technology to develop goal models for solving a particular problem. The students had similar backgrounds and no working knowledge of goal modeling. They were all shown the same short video presentation on goal decomposition – as a top-down process. Then they were given a short textual description of a problem (a version of the classic Meeting Scheduling problem) and about 40 minutes to

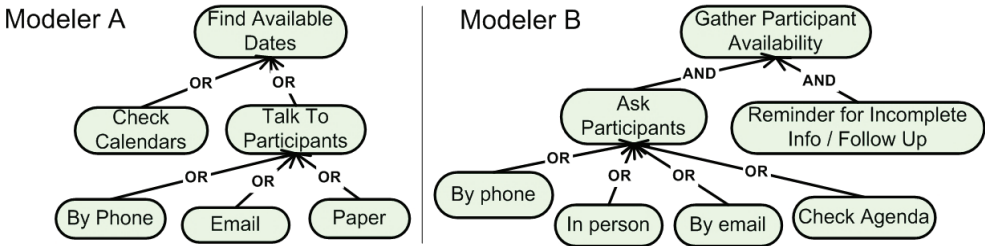



**Fig. 1.** Different Modelers Produce Different Models

develop goal models identifying alternative solutions for the set problem. The problem statement was illuminating as to what concerns and alternatives were to be addressed (e.g. different communication media). The participants produced different models. The sizes ranged from 18 to 58 nodes and from 7 to 22 decompositions, often very different in structure and approach. Figure 1 depicts parts of two such models. These particular models exhibit a great degree of similarity (relative to other possible pairs). They are not the same however: they arrive to a slightly different set of leaves (relative to the other pairs in our sample) and they have some structural and semantic (wrt. the goal descriptions) dissimilarities. Their differences are instructive. Modeler A lists three mechanisms to obtain information from participants; modeler B includes four, and these are slightly different (both include phone and email, A includes paper, B includes 'in person' requests, and B also includes checking the agenda of the participant—a task that A believes is disconnected from the process of talking to participants). Notice that, for B, "talking to participants" is expressed as "asking for availability". Also, modeler A does not consider a follow-up as part of their goal model. Finally, the top goals are different. B is only concerned with gathering participant availability, while A's top goal is to find available dates (despite the fact that the two means by which A plans to fulfill that goal would not necessarily lead to its satisfaction).

We find these slight differences quite curious: why do they occur and what is the impact of each choice? Which of the produced decomposition models is better? An answer to this would assume a clear criterion of "betterness" of one decomposition over the other. It appears that one needs to quantify into measurable criteria the supposed reasons for using goal models, such as pertinence and sufficient completeness of the resulting requirements, comprehensibility of the decomposition per se, its appropriateness in explaining requirements to users or in allowing better readability of complex requirements documents [14].

But even if we are able to perform "betterness" comparisons does this give us a deep understanding of the *nature* of decomposition? What we find central in understanding goal decomposition models is a study of the very low-level human activities that relate to creating and using decompositions. How do humans produce decompositions? How do they read and understand them? What aspects influence decomposition development and understanding and how? And are those differences across modelers truly problematic, or should they be seen

as an opportunity for them to learn more about the subtleties of their domain and about the perceptions of their peers [5]? In our endeavour to shed some light on these questions and inform further experimentation, we began by looking for relevant empirical evidence in a variety of research areas outside requirements engineering – we present some examples below.

## 3 Understanding Decomposition

Our preliminary literature search was performed in an attempt to isolate any empirical work done concerning decomposition. Much of the literature we found is on the benefit of pre-constructed decompositions. Armstrong et al., for example [1], are frequently cited as empirical evidence for the usefulness of decomposition as a tool for assisting human judgment. In this study, a number of students were split into two equal groups and assigned the task of solving a problem. The problem was either presented as a single non-decomposed question, e.g. *"how many packs (rolls) of Polaroid color films do you think were used in the United States in 1970?"* or presented as a decomposed set of questions: *"a) how many people do you think were living in the U.S. in 1970?" "b) in 1970 what do you think was the size of the average family living in the U.S?" "c) in 1970 what percentage of the families do you think owned cameras?"* etc. It was found that more accurate answers were produced by the students who received the decomposed problem. A similar study was performed by Dennis et al. [4] in which multipart questions were presented as a whole, e.g., *"what do you feel are the most important outputs from, inputs to, and data elements in the proposed computer system for Ace Video Rental?"* Or presented one at a time as individual questions, e.g., *"what do you feel are the most important inputs to the proposed [...] ?"*. It was found that more ideas were generated by the groups that received the individual decomposed questions. Neither study, however, speaks about the generation of the decomposition and what the impact to finding the right solution is when the decompositions are improper (incomplete, misleading etc.).

Lyness & Cornelius [9] performed a complex experiment in which students were asked to judge the quality of hypothetical college professors using either decomposed or non-decomposed methods. The non-decomposed method was to give a single rating out of 7 on a Likert scale. The decomposed method had the students rating for many (3, 6, or 9) dimensions given to them, e.g., knowledge of subject, grading philosophy, and testing procedures. These were then placed in weighted combinations. It was found that the subjects using the weighted decomposed method offered more reliable results. In terms of goal decomposition, that could potentially imply that breaking down and aggregating satisfaction criteria could lead to more accurate satisfaction assessment. Note, however, that presence of pre-existing decomposition is assumed.

Decomposition has been shown to be coarsely effective in this same sense in other areas as well. Hertz [7] shows it in capital investments, Polya [10] as a basis for mathematic problem solving, and Raiffa [11] in management science. These works however focus mostly on the application of decomposition to these areas

rather than offering strong empirical evidence. Our understanding so far is that research on the matter has surprisingly been abandoned in the recent years.

## 4 Conclusions

The empirical results that decomposition appears to help correct problem solving and assessment can be argued to support the utility of goal models in requirements engineering, as well. Thus top-level decomposition may guide correct identification of lower-level ones and eventually of leaf level tasks. A fundamental difference however is that in goal modeling it is assumed that the same agent produces the decomposition and continues with solving the problem (i.e. introduce more decompositions or operationalizations) while in the experimental studies an authoritative high-level decomposition is considered to be a given. In that regard, assuming that assistance in solving a problem also implies assistance in understanding an existing solution, the experimental results may also suggest that decompositions appropriately prepared by analysts are effective in communicating their solutions to e.g. clients and other stakeholders. Of course, the nature of the problem at hand (e.g. to find an answer to a unique question vs. to find an optimal set of requirements) seems to require consideration when utilizing those results.

Our sense is that many questions regarding decomposition are yet to be addressed. For example, why does decomposition of a problem aid in solving it? Is some particular sort of decomposition a natural problem solving method, while other sorts less enabling or even obstructing of the problem-solving process? Are there other representations that are not hierarchical and aid effectiveness even more? What is "effectiveness" after all, especially in less precise problems, which are typically the requirements engineering ones?

## 5 On-going and Future Work

We intend to continue our effort to better define the empirical research agenda on decomposition. Our understanding so far suggests three fronts on which such study needs to be performed. Firstly, define more rigorously the various uses of decomposition, which may be competing with respect to a "goodness" measure. For instance, the use of decomposition by one person as an aid to solve a problem (e.g. the way pen and paper are an aid to performing complex multiplications) may result in a model that is not optimal for communicating a solution to somebody else. Secondly, understand the role of the process and the agent in the decomposition: what is the difference between a structured approach [12, 3, 8] and a free-form approach, with respect to identified betterness criteria? How do agent characteristics affect the result and/or understanding thereof? Thirdly, how does our acquired understanding of decompositions relate to $i^*$ or other goal modeling meta-models? Is $i^*$'s means-ends and e.g. KAOS's OR-decomposition ontologically the same and if not what is the impact of their differences in developing them? Is OR-decomposition truly a decomposition (as AND-decomposition is)

or a mere way to express alternative AND-decompositions – which effectively necessitates investigating the two as separate kinds of concepts? Efforts to clarify those aspects are well under way (e.g. [2, 13]) further motivating the question of what empirical research program can inform and be informed by these works.

## References


1. J. S. Armstrong, W. B. Denniston, and M. M. Gordon. The use of the decomposition principle in making judgments. *Organizational Behavior and Human Performance*, 14(12):257–263, 1975.
2. Carlos Cares and Xavier Franch. A metamodelling approach for *i** model translations. In Haralambos Mouratidis and Colette Rolland, editors, *Advanced Information Systems Engineering*, volume 6741 of *Lecture Notes in Computer Science*, pages 337–351. Springer Berlin / Heidelberg, 2011.
3. Robert Darimont and Axel van Lamsweerde. Formal refinement patterns for goal-driven requirements elaboration. In *Proceedings of the 4th ACM SIGSOFT symposium on Foundations of Software Engineering (SIGSOFT '96)*, pages 179–190, New York, NY, USA, 1996. ACM Press.
4. A. R. Dennis, J. S. Valacich, T. Connolly, and B. E. Wynne. Process structuring in electronic brainstorming. *Information Systems Research*, 14(12):268–277, 1996.
5. Steve Easterbrook, Eric Yu, Jorge Aranda, Yuntian Fan, Jennifer Horkoff, Marcel Leica, and Rifat Abdul Qadir. Do viewpoints lead to better conceptual models? an exploratory case study. In *In Proceedings of the 13th IEEE International Requirements Engineering Conference (RE'05)*, pages 199–208, 2005.
6. Xavier Franch, Gemma Grau, Enric Mayol, Carme Quer, Claudia P. Ayala, Carlos Cares, Fredy Navarrete, Mariela Haya, and Pere Botella. Systematic construction of i* strategic dependency models for socio-technical systems. *International Journal of Software Engineering and Knowledge Engineering*, 17(1):79–106, 2007.
7. D. B. Hertz. Risk analysis in capital investment. *Harvard Business Review*, pages 86–106, 1964.
8. Sotirios Liaskos, Alexei Lapouchnian, Yijun Yu, Eric Yu, and John Mylopoulos. On goal-based variability acquisition and analysis. In *Proceedings of the 14th IEEE International Requirements Engineering Conference (RE'06)*, Minneapolis, Minnesota, September 2006. IEEE Computer Society.
9. K. S. Lyness and Cornelius E. T. A comparison of holistic and decomposed judgment strategies in a performance rating simulation. *Organizational Behavior and Human Performance*, 29(1):21–38, 1982.
10. G. Polya. *How to Solve It.* Princeton: Princeton University Press, 1945.
11. H. Raiffa. *Decision analysis: Introductory Lectures on Choices Under Uncertainty.* Addison-Wesley, 1968.
12. Colette Rolland, Carine Souveyet, and Camille Ben Achour. Guiding goal modeling using scenarios. *IEEE Transactions on Software Engineering*, 24(12):1055–1071, 1998.
13. Angelo Susi, Anna Perini, John Mylopoulos, and Paolo Giorgini. The Tropos metamodel and its use. *Informatica*, 29:401–408, 2005.
14. Axel Van Lamsweerde. Goal-oriented requirements engineering: A guided tour. In *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering (RE'01)*, page 249, 2001.

# Development of Agent-Driven Systems: from i* Architectural Models to Intentional Agents' Code

Maurício Serrano and Julio Cesar Sampaio do Prado Leite,

Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro,
Rua Marquês de São Vicente 225, Ed. Padre Leonel Franca 13o. andar,
Rio de Janeiro, Brasil
mauserr@les.inf.puc-rio.br, http://www-di.inf.puc-rio.br/~julio/

**Abstract.** The intentionality concept can improve the cognitive capacity of software agents, especially if the proposed intentional reasoning engine deals with softgoals at runtime. In this scenario, the use of an intentionality-based technological set to develop agent-driven systems from i* models to code is adequate. In this paper, we propose heuristics to improve the development of agent-driven systems from i* models to Belief-Desire-Intention-based code. Moreover, we apply a fuzzy-logic-based mechanism to deal with softgoals "on the fly", improving the reasoning engine of intentional agents. We compare our efforts with related work and illustrate our contributions with a case study.

**Keywords:** i* models, model-driven development, intentional agents, transformational heuristics, fuzzy-logic-based reasoning engine.

## 1 Introduction

According to [1], a multi-agent system (MAS) is a system composed of many intelligent agents that work in a collaborative manner in order to achieve their goals. In other words, these intelligent agents tend to automatically find the best solution for the delegated goals – i.e. "without intervention". Therefore, the agents can assume different responsibilities – normally represented as different roles or capabilities. Moreover, they can perform several tasks by also adapting themselves to better achieve their goals.

A multi-agent system can be driven by behavioral agents [2] or intentional agents [3]. On one hand, the former orientation is more appropriate to deal with systems based on agents that can be modeled and implemented by using specific behaviors, such as: eating, sleeping, running, reading, and so on. Therefore, the behavior-based orientation seems proper to represent, for example, self-organization in social colonies (e.g. ant colonies). On the other hand, the latter orientation can improve the reasoning and learning capacity of intelligent agents by using the intentionality abstraction. According to [3], an intentional multi-agent system represents an adequate way to deal with human practical reasoning and to improve the goals formation and human-mental states interpretation. The BDI (Belief-Desire-Intention) model is an intentional model, intensely investigated in the Artificial Intelligence

field. More recently, this model has been applied to intelligent agents by allowing them to deal with (i) **beliefs** – i.e. the agent's knowledge about, for example, the real world or a specific context; (ii) **desires** – i.e. the goals to be achieved by the agents; and (iii) **intentions** – i.e. a sequence of tasks that must be performed by the agents to achieve the delegated goals (in the BDI model, intentions have the semantics of actions with the purpose to achieve the desires).

In order to guide the development of multi-agent system, TROPOS [4] offers an agent-oriented methodology. This method is centered on the i* Framework [5], which abstractions (e.g. goals, softgoals, tasks, resources, beliefs and others) are used to model the requirements and design details of the MAS under development. There are different approaches [6] [7] centered on the TROPOS methodology that propose solutions to respectively: (i) develop MAS by applying heuristics to go from i* models to BDI-based code; and (ii) perform the development of MAS from organizational architectures in i* to architectures based on agents by also applying some Agent UML diagrams to capture the agents' intentionality. Section 3 highlights how our work differs from previous work.

In this paper, we propose specific design and implementation heuristics to drive the development of MAS from the requirements to code centered on intentional agents. Therefore, we propose: (i) using i* for both the requirements model as well as the design model; and (ii) the implementation of the modeled system based on the BDI model of the JADEX [8], an add-on to the JADE MAS platform. JADEX is also centered on the intentionality abstraction. Moreover, we offer an approach centered on fuzzy-logic to analyze the impacts of tasks on softgoals implemented as a reasoning engine for the intentional agents.

This paper is organized in Sections: Section 2 discusses the main objectives of our research; Section 3 presents some scientific contributions; Section 4 summarizes the proposal by presenting the final considerations; and finally, in Section 5 we consider the ongoing and future work.

## 2 Objectives of the research

In order to facilitate the presentation of our research, we use a case study – the Lattes-Scholar [9]. It uses two specific services: (i) the Lattes service [10], which consists of a base of scientific and technological curriculums maintained by the Brazilian Government; and (ii) the well-known Google Scholar service. The objective is to list the publications from an author's curriculum sorted by the number of citations.

We modeled the requirements and the detailed design of Lattes-Scholar by using i*, specifically the Strategic Dependency (SD) and Strategic Rationale (SR) models. Fig. 1 shows the architecture of Lattes-Scholar modeled in i*.

There are some differences between the i* modeling and the specification in BDI notation, such as: (i) the i* modeling represents a network of social actors, while the BDI-based specification represents the internal and mental architecture of a unique software agent; and (ii) the i* modeling represents sofgoals as well as how tasks, goals and softgoals impact on these softgoals by using positive and negative contribution links.

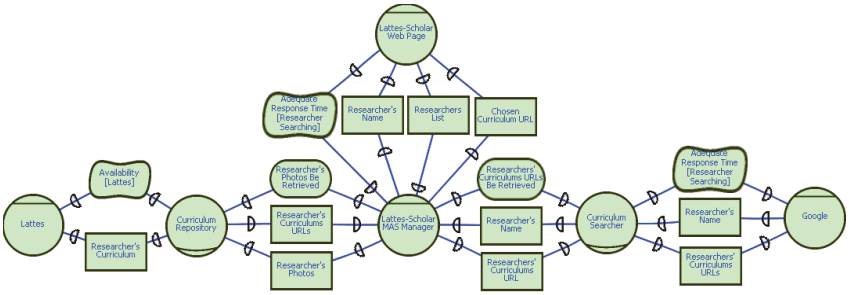



**Fig. 1.** Lattes-Scholar's architecture modeling in i*

Although there are semantic differences between the models i* and BDI, these models share common concepts, such as: actor/agent, goal, task/intention, belief, resource/belief, and others. Therefore, it is possible to associate the abstractions of these models, as presented in Fig. 2. Table 1 describes some of these (design) transformational heuristics to produce a BDI specification from SD and SR i* models.

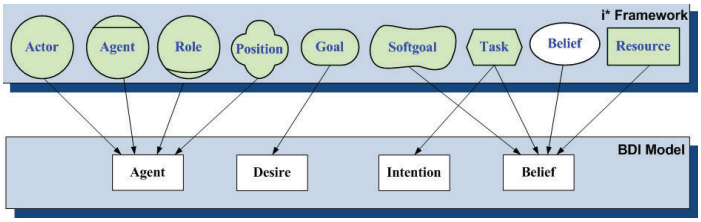



**Fig. 2.** Association between i* abstractions and BDI abstractions

It is important to consider that – after applying the design heuristics – there are some gaps in the BDI model of the agents, such as *types*. The software engineer must manually fill in these gaps.

Now, our interest is in a lower abstraction level, on which intelligent agents centered on BDI specifications are implemented as intentional agents in JADEX. The JADEX Framework implements a BDI architecture for agents of the JADE platform. The intentions defined in BDI specifications are implemented as Java object classes – i.e. Plan – in JADEX. Furthermore, JADEX is an add-on for the JADE platform, which implements the FIPA agent communication protocol and messages exchange. Fig. 3 illustrates the associations between the abstractions of the BDI specification and the MAS code in JADEX. We omitted the implementation heuristics (from BDI models to JADEX agents' code) due to lack of space.

Basically, the executable agents are implemented as JADEX agents. Non-executable agents (originated from roles and positions) are implemented as capabilities, which are acquired at runtime by software agents of the MAS platform. Desires are translated as goals and maintained or performed according to the tag "type" from the BDI specification. The intentions are translated as plans of agents – i.e. Java classes that extend the "Plan" class of JADEX. Beliefs with 0..1 or 1..1

cardinalities are translated as beliefs of agents, while beliefs with 0..n or 1..n cardinalities are translated as a set of beliefs.

**Table 1.** Transformational Heuristics: from i* Framework to BDI specification

| # | When it is applied | Action | Exit (XML Fragment) |
|---|---|---|---|
| 01 | For each actor (or agent, or role, or position) of the system. | Create an agent in the BDI specification. Actors and agents have a tag runnable="yes". Roles and positions have a tag runnable="no". | <agent<br>name="[actor_name]"<br>runnable="[yes\|no]"><br><beliefs/><br><desires/><br><intentions/><br></agent> |
| 03 | For each softgoal of an actor. | Create a belief with the type "Softgoal". | <belief<br>name="[softgoal_name]"<br>type="Softgoal" /> |
| 06 | For each task that is performed to achieve a goal of an actor. | Create an intention and associate it to a desire (goal). | <intention name="[task_name]"><br><desire>[goal_name]</desire><br><script lang="" /><br></intention> |
| 07 | For each task that is performed to achieve a goal of an actor. | Create a belief with the type "Task". | <belief<br>Name="[task_name]"<br>type="Task" /> |
| 14 | For each dependency *per* goal. | Create a desire in the specifications of both involved agents. | <desire name="[goal_name]"<br>type=""/> |

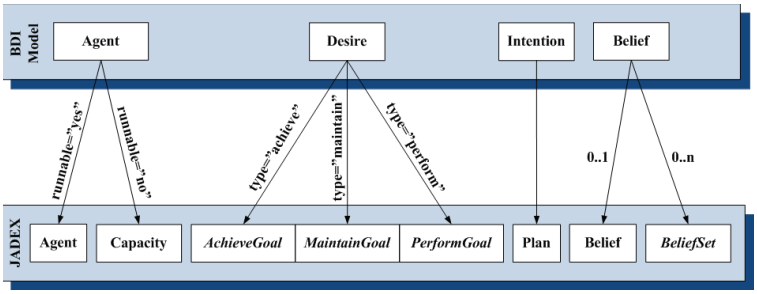



**Fig. 3.** Association between BDI abstractions and JADEX abstractions

We developed a qualitative reasoning engine [11] based on fuzzy logic to allow agents to perform their tasks by considering different quality criteria – i.e. modeled as softgoals in the i* models – and negative and positive contributions to them. The softgoals and contributions are available to the reasoning engine as agents' beliefs. The fuzzy logic usage allows agents to reason – "on the fly" – about softgoals by simulating the algorithms proposed in the i* Framework for models analysis. Other proposals [12] [13] are based on quantitative reasoning engines. According to our investigation by developing different intentional MAS, the use of a quantitative

reasoning to deal with softgoals can be viewed as inadequate, especially if we consider their subjective and imprecise nature.

## 3 Scientific contributions

In [6], the authors suggest some heuristics that do not cover all abstractions of the i* Framework. For example, it does not deal with the role and position abstractions. They also do not address how to translate dependencies to interaction protocols. Moreover, they propose a reasoning engine based on softgoals priorities. In our proposal [11], we offer heuristics for all i* abstractions as well as our reasoning engine is enriched by a fuzzy-logic-based set to improve the agents' cognitive capacity on dealing with the intrinsic uncertainness of the modeled softgoals.

In [7], the authors use an extension of the UML to model the agents and their intentions. We argue that i* models can capture these intentions without the introduction of other notations or diagrams. However, some UML diagrams can be appropriate to deal with situations/aspects i* does not cover, such as temporality.

Summarizing the scientific contributions of our research, we can mention: (i) the usage of the i* as both requirements model and design model instead of using it to only model the requirements; (ii) the heuristics from i* abstractions to BDI abstractions – i.e. from i* models to BDI specification; (iii) the heuristics from BDI abstractions to JADEX BDI abstractions – i.e. from BDI specification to MAS code in JADEX; (iv) an approach centered on fuzzy-logic to analyze the impacts of tasks on softgoals; and (v) a reasoning engine for agents centered on this approach.

## 4 Conclusions

In this paper we present an overview of our work in order to develop MAS centered on the intentionality concept. The proposed support offers heuristics to conduct this development from i* models to the BDI-based code in JADEX. In addition, the proposed reasoning engine covers all i* abstractions, including, for example, belief, role and position abstractions.

One contribution of our efforts consists of avoiding the introduction of different notations and diagrams to capture the actors and agents intentions. In this case, we propose the use of the i* models (SR and SD models) to graphically represent the requirements and design details.

An intentional MAS reasoning engine combined with a fuzzy-logic mechanism [11] improves the cognitive capacity of the agents by allowing, for example, to deal with softgoals at runtime. The main idea is to analyze – "on the fly" – the impacts of the tasks, specified in the i* models, on the sofgoals, also specified in these models.

Finally, we can argue that, based on the systematic and incremental application of our approach to the Lattes-Scholar case study, we seem to be in the right direction in terms of: (i) reducing the necessary models to specify the requirements and the design details; (ii) improving the cognitive capacity of software agents centered on the intentionality through the use of the BDI model as well as the fuzzy-logic-based

support set; and (iii) conducting the development of an intentional MAS from the requirements to code.

## 5 Ongoing and future work

Since the beginning of 2010, we have been applying our proposal to the Lattes-Scholar case study. Among other contributions, this incremental and systematic development has allowed us to evolve both the Lattes-Scholar system and our proposal – i.e. the proposed heuristics, our fuzzy-logic-based approach and our agents' reasoning engine.

As future work, we intend to develop a tool support to help the transformation process from the i* models to the BDI-based code in JADEX. The main idea is to semi-automate this process using the proposed heuristics and intentionality.

## References

1. Wooldridge, M.: An Introduction to MultiAgent Systems. John Wiley & Sons Ltd, ISBN 0-471-49691-X, 366 pages (2002)
2. Bellifemine, F., Caire, G., Grenwood, D.: Developing Multi-Agent Systems with JADE. John Wiley & Sons, ISBN-13: 978-0-470-05747-6, 286 pages (2007)
3. Bratman, M. E.: Intention, Plans, and Practical Reason. University of Chicago, ISBN: 1575861925, 208 pages (1999)
4. Castro, J., Kolp, M., Mylopoulos, J.: Towards Requirements Driven Information Systems Engineering: The Tropos Project. Information Systems, vol.27, n. 6, pp. 365-389 (2002)
5. Yu, E.: Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. 3rd IEEE Int. Symp. on Requirements Engineering (RE'97), pp. 226-235, Washington D.C., USA (1997)
6. Perini, A. and Susi, A.: Automating Model Transformations in Agent-Oriented Modeling. In Agent-Oriented Software Engineering VI, Springer LNCS, vol. 3950, pp. 167-178, Berlin/Heidelberg (2006)
7. Silva, C., Dias, P., Araújo, J., Moreira, A.: De Arquitecturas Organizacionais em i* a Arquitecturas Baseadas em Agentes: Uma abordagem orientada a modelos. XIV Workshop on Requirements Engineering, page 48, April (2011)
8. Braubach, L., Lamersdorf, W., Pokahr, A.: Jadex: Implementing a BDI Infrastructure for JADE Agents. Distributed Systems and Information Systems, vol. 3, n. 3, pp.76-85, September (2003)
9. Lattes-Scholar: Requirements Engineering Group at PUC-Rio. Available at: http://www.er.les.inf.puc-rio.br/~wiki/index.php/Lattesscholar (May 2011)
10. Lattes Platform. Available at: http://lattes.cnpq.br/ (May 2011)
11. Serrano, M., Serrano, M. and Leite, J.C.S.P.: "Dealing with Softgoals at Runtime: A Fuzzy Logic Approach", to appear at the 2nd International Workshop on Requirements@Run.Time (2011)
12. Penserini, L., Perini, A., Susi, A., Morandini, M., Mylopoulos, J.: A Design Framework for Generating BDIAgents from Goal Models. 6th Int. joint Conference on Autonomous Agents and Multiagent Systems, DOI:10.1145/1329125.1329307, number 149 (2007)
13. Weber, C., Current, J., Benton, W.: Vendor Selection Criteria and Methods. European Journal of Operational Research, vol. 50, no. 1, pp. 2–18 (1991)

# Towards Adoption and Use of i* to Support Enterprise Architecture in Organizational Settings: a Position Paper

Daniel Gross[1], Uwe Zdun[2], Eric Yu[1]

[1] Faculty of Information, University of Toronto, 140 St. George Street, Toronto, Canada;
[2]Faculty of Computer Science, University of Vienna, Berggasse 11, 1090 Vienna, Austria

{daniel.gross@utoronto.ca, eric@cs.toronto.edu, uwe.zdun@univie.ac.at}

**Abstract**. The i* modeling framework has over the last decade received much attention amongst research groups worldwide. However, despite the proliferation of i* research, there has been little work to-date to exploring the adoption and use of i* in enterprise organizational settings. This paper discusses some key challenges for using i* in an enterprise to support the modeling and analysis of distributed decision-making during the design and evolution of an enterprise architecture in an organizational setting, and proposes a research agenda to address such challenges in order to facilitate the adoption and use of i* in enterprise organizations.

**Keywords:** Enterprise Architecture, Decision-orientation, Agent-orientation, Goal-orientation, distributed decision-making

## 1. Introduction

The i* modeling framework has over the last decade received much attention amongst research groups worldwide. However, despite the branching out of i* research into various domains, there has been little work to-date on exploring the adoption and use of i* to support stakeholder and designers involved in distributed decision-making in general and during the design and evolution of an enterprise architecture in an organizational setting in particular. Yet, the i* modeling framework appears particularly well suited to support enterprise architectural decision-making in organizations, which requires dealing with the interests and needs and decision-making of various business and technical stakeholders and designers holding different responsibilities in organizational setting. For example, using intentional actors and goal concepts supports representing and analyzing the different organizational stakeholders and designers, their interests and decision-making, and their organizational interdependence, while they aim to achieve respective goals.

However, supporting distributed decision-making during enterprise architectural design and evolution in an organizational setting raises various modeling and methodological challenges. For example, it is common for large organizations to operate hundreds if not thousands of business applications to support their business processes and strategy, with different teams maintaining and evolving different

enterprise applications in the organization [1]. Various decisions that individual teams more or less independently make could potentially affect whether the enterprise organization as a whole is hindered or helped in achieving short term and/or longer term goals [1, 2]. At the same time various management stakeholders and enterprise-wide architects aim to promote enterprise wide business and technical goals and choices across the organization, negotiating decisions, the adoption of decisions as well as dealing within decision compliance in the organization [3, 4]. Decision making must thus be coordinated across teams and across managerial levels [5].

Furthermore, in large organizations decision-making authority is by necessity distributed [6, 7]. Different organizational participants, each contributing distinct knowledge and skills, are given authority and autonomy to make decisions. A key challenge is to allow for autonomous decision-making in various organizational localities, while ensuring the overall consistency of decision-making within the enterprise organization [8].

Challenges also emerge from the various different knowledge domains organizational participants work in and the different levels and scope of details associated with participants' individual vantage points. For example, strategic management stakeholders are often interested in the enterprise as a whole and the broader market place, while business architects' main interests are business structures and processes. Enterprise wide architects' main focus is on the enterprise application landscape, while individual enterprise application architects look at individual capabilities and services provided to the enterprise. While each such organizational participants have distinct responsibilities and purposes in mind, alignment across such diverse domains and vantage points during decision-making is essential to the success of the enterprise.

Finally, there are also challenges related to the adoption of novel methods and related tool support. Introducing new methods and tools to relevant stakeholders and designers in enterprise organizations is usually a difficult task. Organizational participants often have limited time and budget available for learning and exploring novel modeling and analysis techniques.

Furthermore, usually there are many different enterprise tools in use in an enterprise organization to support the development and deployment of business strategies, processes and applications. Tools that support the various organizational participants need to be integrated with an i* decision modeling and management tool to support the systematic linking of decision-making to artifact creation, change and evolution. Tool support also needs to be tailored to the needs of various different tool users, each with a different vantage point on decision-making. For example, a management stakeholder would likely have a different stake in decision-making than enterprise architects, each needing different kinds of features and views on decision-making in the organization.

## 2. Objectives of the research

The overall objective of this research is to study the adoption and use of the i* modeling framework in an enterprise organizational setting to support dealing with

distributed decision-making during enterprise architectural design and evolution. To this end this research aims to explore:

a) in what way the i* decision modeling and analysis approach could be adapted to meet the needs of different types of organizational participants in an enterprise organizational setting while individually, collectively and/or collaboratively performing decision tasks;
b) what additional modeling and analysis features could be identified to support distributed decision-making tasks in an enterprise organizational setting; including support for dealing with decision-making at different organizational levels of abstraction, such as the enterprise business level, the business processes level, the enterprise architecture level, the enterprise application level, and enterprise component level, as well as the IT infrastructure level [9];
c) what individual and collaborative tools to offer to organizational participants to support i* modeling and analysis in an enterprise setting, including how different stakeholders and designers with different technical skills could be supported, as well as the integration of tool support into the existing modeling and analysis tool set used by stakeholders in the enterprise organization;
d) how to deal with in an integrated manner with qualitative and quantitative goals and goal reasoning often employed in enterprise organizational settings;
e) how to ease the adoption and use of the notation, method and tool; this involves exploring user interaction with the notation, method and tools in general and usability of the modeling notation and language and tool support in particular.

The aforementioned research objectives are quite broad and should rather be seen as outlining a research agenda that involves different research strands, each focusing on a different aspect of the problem of facilitating the adoption and use of the i* modeling framework to support enterprise architectural design within an enterprise organizational setting.

## 3. Scientific contributions

While much work has been done on exploring different areas in which the i* modeling framework can be applied to advantage, little work to-date has focused on the application and use the i* modeling framework to support distributed reasoning and decision-making in enterprises in general and during enterprise architectural decision-making in an enterprise organizational setting in particular [8, 10]. More specifically, little work has been done on exploring the use of intentional actor and goals to represent and analyze design reasoning and decision-making of different organizational participants in enterprise organizations who hold different

organizational responsibilities, and pursue conflicting and/or synergistic business or system goals.

This research can also be seen as contribution to the body of works related to Enterprise Architecture [1, 3], which deals with the continuous and iterative improvement of existing and planned IT support for an organization to support the organizations business processes, goals and strategies [1]. Current works on Enterprise Architecture mainly focus on the artifacts produced and evolved during enterprise architectural efforts. Little work has looked at the intentional and organizational decision-making dimension that occurs during architectural and architectural relevant decision-making [10, 11].

## 4. Ongoing and future work

Some initial work has been done in applying i* within an enterprise organizational setting. One work explored the use of i* to support understanding stakeholders decision-making viewpoints during an enterprise-wide architectural evolution effort towards service-orientation, while another work looked a tactical change in a product during an enterprise wide architectural evolution effort [8, 10, 12]. During these works some possible extensions to i* have been identified such as intentional viewpoint concept to support representing and reasoning about alternative argumentations put forward by architectural designers, who have overlapping areas of responsibilities in the organization; as well as the linking of i* models to models of business and architectural design artifacts produced as a result of decision-making.

During this initial research work, the need for a broader investigation into the adoption and use of i* in enterprise organizations, as outlined in Section 2, was identified.

Feedback on the use of i* in an enterprise organization for example indicated the need to tailor i* models, and in particular strategic rationale models, to the specific needs of stakeholders and designers in the organization. Some designers require a deeper understanding of the design issues and argumentation at hand and thus need detailed rationale models; while other designers only need brief reminders of rationales for choices to continue committing to them. For such purpose a simplified version of a rationale model is already sufficient. Managerial stakeholders in enterprise organizations were only interested in governance aspects – in what way architectural choices downstream support (or hinder) short term or longer termed organizational goals. For such managers a higher level "dashboard" offering a (real time) views on goal achievement in the organization, including qualitative goals, would be appropriate.

Feedback also indicated the essential need for adequate tool support to facilitate the adoption and use of i* in an enterprise organizational setting. This research therefore includes in its future research effort the exploration and development of adequate tool support to facilitate the adoption of i* modeling and analysis approach within an enterprise organizational setting.

Developing tools that practitioners can use is in particular important since without adequate tool support that is to a sufficient extend integrated with already existing

enterprise tools, it is difficult to encourage organizational participants to adopt and use on their own a novel modeling approach. Having adequate tool that practitioners derive value from and are willing to use, would also be a useful vehicle to study the adoption and use of i* in an enterprise, and to derive additional modeling and analysis requirements. Such an intertwined approach would be consistent with the action research approach [13], a participatory research approach where researchers and study participants are jointly engaged in problems solving (i.e. such as to improve the organizational distributed decision-making capability) through iterative learning cycles and knowledge creation.

## References


1. Alexander, M.E. Tool Support for Enterprise Architecture Management - Strengths and Weaknesses. in 10th IEEE International Enterprise Distributed Object Computing Conference (EDOC'06). 2006.
2. Erl, T., SOA Principles of Service Design (The Prentice Hall Service-Oriented Computing Series from Thomas Erl). 2007.
3. Engels, G., et al., Quasar Enterprise: Anwendungslandschaften serviceorientiert gestalten. 2008: dpunkt verlag.
4. Farenhorst, R. and H. van Vliet. Understanding how to support architects in sharing knowledge. in Sharing and Reusing Architectural Knowledge, 2009. SHARK '09. ICSE Workshop on. 2009.
5. Curtis, B., H. Krasner, and N. Iscoe, A Field Study of the Software Design Process for Large Systems. Communications of the ACM, 1988. **31**(11): p. 1268-1287.
6. Galbraith, J.R., Organization design. 1977, Reading, Mass.: Addison-Wesley Pub. Co. xvi, 426 p.
7. Rasmussen, J., B. Brehmer, and J. Leplat, eds. Distributed decision making - cognitive models for cooperative work. New Technologies and Work. A Wiley Series. 1990.
8. Gross, D. and E. Yu, Supporting the evolution of software architectures in development organizations using intentional agents. Fourth International i* Workshop - istar10, 2010.
9. Zimmermann, O., et al., Reusable Architectural Decision Models for Enterprise Application Development, in Software Architectures, Components, and Applications. 2008, Springer Berlin /Heidelberg. p. 15-32.
10. Gross, D., Software architecture decision-making in organizational settings, in Faculty of Information. 2011, University of Toronto: Toronto. p. 240.
11. Clements, P. and L. Bass, Relating Business Goals to Architecturally Significant Requirements for Software Systems, in Research, Technology, and System Solutions Program. 2010, Software Engineering Institute, Carnegie Mellon University.
12. Gross, D. and E. Yu, Evolving System Architecture to Meet Changing Business Goals: An Agent and Goal-Oriented Approach, in Proceedings of the Fifth IEEE International Symposium on Requirements Engineering. 2001, IEEE Computer Society. p. 316.
13. Kock, N.F., Information systems action research: an applied view of emerging concepts and methods. Integrated series in information systems;. 2007, New York: Springer. xxv, 425 p.

# Towards an i*-based Architecture Derivation Approach

Diego Dermeval[1], Monique Soares[1], Fernanda Alencar[2], Emanuel Santos[1], João Pimentel[1], Jaelson Castro[1], Márcia Lucena[3], Carla Silva[4], Cleice Souza[1]

[1]Universidade Federal de Pernambuco - UFPE, Centro de Informática, Recife, Brazil
{ddmcm,mcs4,ebs,jhcp,jbc,ctns}@cin.ufpe.br
[2]Universidade Federal de Pernambuco - UFPE, Departamento de Eletrônica e Sistemas, Recife, Brazil,
fernanda.ralencar@ufpe.br
[3]Universidade Federal do Rio Grande do Norte - UFRN, Departamento de Informática e Matemática Aplicada Natal, Brazil,
marciaj@dimap.ufrn.br
[4]Universidade Federal da Paraíba - UFPB, Centro de Ciências Aplicadas e Educação, Rio Tinto, Brazil
carla@dce.ufpb.br

**Abstract**. Goal orientation, in particular the i* (iStar) framework, offers expressive models that support requirements engineering. On the other hand, the understanding of how requirements models are related to architectural design is still somewhat limited. In the past years, we have been investigating how to derive architectural models from i* (iStar) models, focusing on modularity. As a result we proposed a Strategy for Transition between Requirements and Architectural Models – STREAM. In this paper, we summarize the current state-of-the-art of the STREAM approach, point out its challenging aspects and describe current ongoing research. Our challenge is to support a broader set of architectural decisions as well as to provide means for partially automating the models transformations.

**Keywords:** iStar, Requirements Engineering, Architectural design, Architecture Documentation, Architectural Decisions, Automation, Model Transformations

## 1 Introduction

Despite Requirements Engineering and Architectural Design being strongly related activities, there is a lack of techniques and methods handling the integration of these activities. Therefore, one of the major research challenges in software engineering is to provide systematic methods for designing software architecture from requirements models [2] [3]. The STREAM (Strategy for Transition between Requirements and Architectural Models) process [4] [14] presents a model-driven approach for generating initial architectures - in Acme [7] - from i* requirements models [13]. The STREAM approach consists of the following steps: (i) *Prepare Requirements Models*, (ii) *Generate Architectural Solutions*, (iii) *Choose an architectural solution*, and (iv) *Derive Architecture*. Horizontal and vertical model-transformation rules were proposed in order to perform the steps (i) and (ii), respectively. Non-Functional Requirements are used in the step (iii) to guide the selection of alternatives in the

architecture. Lastly, in the step (iv) the architecture is refined by using architectural refinement patterns.

Based on the generic STREAM process, some others extensions were proposed: STREAM-Adaptive [11] and F-STREAM [5]. The STREAM-Adaptive approach supports the generation of architectures for self-adaptive systems. This is achieved by enriching the i* models with information required to perform the reasoning related to adaptation, which is performed by pre-defined components. The F-STREAM [5] (Flexible STREAM) uses Software Product Lines principles aiming to make it easier to integrate the STREAM approach with other approaches that are able to handle some specific NFRs.

However, there are still some limitations. For example, only one of the possible architectural views [1] is supported. In addition, no support is given to document the different types of architectural decisions [9]. Finally, the model transformations are not yet automated. Hence, in this paper, we show how we intend to improve the family of STREAM approaches in order to face the last two shortcomings: supporting and documenting a broader set of architectural decisions and automating the model transformations required in the process.

This paper is organized as follows. Section 2 presents the goals of the research. Section 3 describes our proposals towards these goals. Section 4 presents the conclusions while Section 5 points out ongoing and future research.

## 2 Objectives of the Research

The general goal of this research is to enhance the STREAM approach, allowing it to be more complete and viable for industrial use. Therefore, we propose two specific objectives. Firstly, we derive an architectural specification that encompasses the documentation of a broader set of architectural decisions. Secondly, we intend to provide tool support to automate the transformations presented in the STREAM approach and its extensions. Thus, we aim to facilitate and promote the use of those approaches. As a side effect, we contribute to the improvement of the modularity and understandability of i* models.

## 3 Scientific Contributions

This section presents the proposed approaches to satisfy the research goals. Section 3.1 describes how we intend to include architectural decisions in STREAM, while Section 3.2 presents how we plan to automate its model transformations.

### 3.1 Architectural Decisions in the STREAM Process

Based on the classification scheme of architectural decisions proposed by [9], we noticed that the STREAM process only allows the decision-making of a subset of architectural decisions types. In this way, we are extending the STREAM approach in order to support two specific kinds of architectural decisions: the existential and technology decisions. So, to systematize the specification of architectural decisions in the extended STREAM process, we combined the step (iii) and (iv) into a single

activity named *Refine Model with Architectural Decisions* that encapsulates the design choices of the former steps in the classification scheme (existential and technological decisions) that we are using. Moreover, we renamed the steps (i) and (ii) to, respectively, *Requirements Refactoring* and *Generate Architectural Model* (Fig. 1).

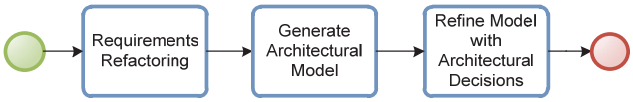



**Fig. 1 Extended STREAM Process**

The aim of this new *Refine Model with Architectural Decisions* activity is to sharpen up the generated architectural model by considering existential and technology decisions. Moreover, through the documentation of these decisions using some documentation template, it is possible to capture the context, rationale and other relevant information about the decisions. A set of documented decisions are the output of each decision-making activity. In this paper, we do not have sufficient space to describe how we plan to record the architectural decisions, see [6] and [9].

Fig 2 illustrates the sub-process that presents the architecture refinement with decisions. In the *Make Existence Decisions* activity, the architect defines elements or artifacts that are required for the system's design or implementation. This kind of decision includes structural as well as behavioral decisions. For example, structural decisions lead to the creation of subsystems, layers, partitions, components, etc. Behavioral decisions are usually related to how the elements interact together to provide functionality or to satisfy a non-functional requirement [9]. For instance, the choice of a specific architecture pattern can be seen as an existence decision, so it is specified in this activity of the process.

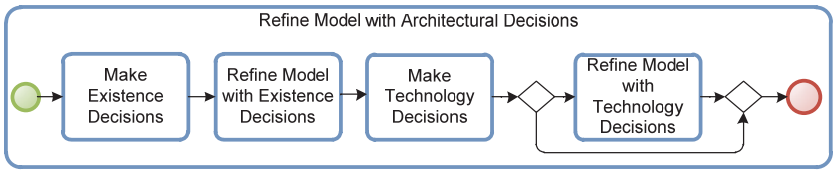



**Fig. 2 Refine Model with Architectural Decisions Sub-process**

The aim of the Refine Model with Existence Decisions activity is to refine the architectural models to reflect the existence decisions made during the earlier activity. Thus, the outputs of this activity include a refined ACME architectural model together with the list of existence decisions made.

The *executive decisions* are the decisions that do not relate directly to the design elements or their qualities, but are driven more by the business environment (financial), the development process (methodology), the people (education and training), the organization, and to a large extent the choices of the technologies and tools [9]. There are different kinds of executive decisions, but at this time we will focus only on technology decisions. So, the *Make Technology Decisions* activity involves decisions that should be part of an architectural specification, mainly, to guide the implementation of the architecture. Examples of technology decisions are the choice of a programming language and the choice of a specific framework.

There is a need to assess if the selected technology architectural decision affects or impacts the ACME architectural model. If this is the case, these decisions are considered in the *Refine Model with Technology Decisions* activity to further refine the ACME architectural model. For instance, selecting a specific API to be integrated with the architecture. Otherwise, if the decision does not affect the architectural model, the process is concluded. For example, the choice of a programming language.

In the next subsection we examine another challenge: the need to provide some degree of automation (tool support) for the approaches.

## 3.2 Automating Model Transformations

Some activities of the family of STREAM approaches can be time consuming. Hence, we should examine if some kind of tool support could be provided, at least to partially automate the processes. The (i) Prepare Requirement Models and (ii) Generate Architectural Solutions steps of STREAM are amenable to some degree of automation, since they rely on model transformations. The first activity relies on horizontal rules to refactor the i* requirement models prior to the architectural model generation. The second activity applies vertical rules to derive architectural models from the refactored i* models.

These transformation rules can be precisely defined using the QVT transformation language (Query/View /Transformation) [12], in conjunction with OCL (Object Constraint Language) [10] to represent the constraints. The transformation process requires the definition of transformation rules and metamodels for the source and target languages. The horizontal rules that aims to refactor the i* models have the i* language both as source and target language. On the other hand, recall that the vertical rules are used to generate architectural models (in ACME) from modularized requirements models (in i*). Hence, our vertical rules have i* as the source language and ACME as target language. Once defined and specified using QVT and OCL, the transformation rules could be incorporated in a tool, such as the iStarTool [8].

Note that the iStarTool already has internal representation of the i* metamodel and could be extended to allow the implementation of the new transformation rules. In doing so, the *Prepare Requirement Models* activity could become semi-automatic. The user would still need to select the candidate sub-set of elements to be factored out. After this selection, all the other steps of the activity could be automated. As a result, the refactored i* model could now be obtained with the press of a button.

The *Generate Architectural Solutions* activity generates candidate Acme models from the modularized i* models. The alternative solutions are derived from the inherent variability of i* models (e.g., due to the Means-Ends relationships). The choice of the candidate solution can be influenced on softgoal or quality attributes present in requirements models. Hence, we envisage including in the iStarTool the ability to generate all possible set of candidate architectures. Moreover, the tool could indicate the degree of satisfaction of a given set of softgoals for each architecture. Furthermore, the generated Acme models are used in subsequent steps (iii) and (iv) of the STREAM Approach. It remains to be studied how these steps could be partially automated.

By automating the model transformations, several experiments can be performed to evaluate different architectural models without additional costs.

## 4 Discussion

In this paper, we have proposed two approaches aiming to improve the systematic process that generates architectural models from i* models. We have presented an approach to include support for recording architectural decisions in STREAM. Furthermore, we have indicated how the horizontal and vertical model transformations presented in the process could benefit from automation and tool support.

The first approach improves the family of the STREAM approaches, by allowing the rationale of the decisions made to be recorded. With this extension, it is possible to specify a more complete architecture by defining a broader set of architectural choices - for example, technology decisions. Moreover, by documenting the architectural decisions, the information that underlies the context of a decision can be recorded. However, such extra information may overload the refinement step of STREAM with documentation activities. Nonetheless, we believe that the benefits of documenting an architectural decision [6] far compensate the extra effort required for recording the rationale. We also need to investigate if we can anticipate specific kinds of decision-making that are common to these in earlier steps of the process.

The second improvement proposed in this paper minimizes the effort of applying the model transformation rules manually. Besides, it eliminates the possibility of making mistakes when manually applying these rules. Since the transformation process could be automatically supported, another positive aspect of this improvement is the increase of productivity, as it enables a simplification of the process and reduces the amount of manual activities.

## 5 Ongoing and Future Work

We offer a family of a systematic method that derives (with semi-automatic support) a candidate architectural design from i* models. With this in mind, we can describe specific ongoing and future work for each approach presented in this work.

On one hand, we are evolving the approach to include architectural decisions. We are defining how we will document the architectural choice. Our first attempt is to use a template as the proposed by Garlan et al. [6]. Hence, we need to evaluate how the i* models can guide or aid the documentation of the decisions. We are also investigating where does design decisions take place in STREAM. As future works, we will specify an extended STREAM approach that results in an architectural design that encompasses both the architectural decisions and the representation views. Furthermore, we need to further validate the approach with several case studies. We also intend to integrate the new process with a tool to manage the artifacts produced in the architectural design step.

On the other hand, we are extending the iStarTool [8] to support the horizontal mapping rules which modularize the i* models. The rules are specified in QVT and OCL. As future work, we plan to develop an iStarTool API to incorporate the vertical mapping rules, which generates the initial model Acme from modularized i* models.

Last but not least, experiments are required to validate the family of STREAM approaches as well as the new iStarTool functionalities.

## References


1. Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice (2nd Edition) Addison-Wesley Professional, 2003.
2. Berry, D. M., Kazman, R., Wieringa, R.: Second international workshop on from software requirements to architectures (straw'03). SIGSOFT Softw. Eng. Notes 29, 1–5, 2004.
3. Castro, J., Kramer, J.: From software requirements to architectures (STRAW'01). SIGSOFT Software Eng. Notes 26, 49–51, 2001.
4. Castro, J., Lucena, M., Silva, C., Alencar, F., Santos, E., and Pimentel, J.: Changing Attitudes Towards the Generation of Architectural Models. In: Journal of Systems and Software, 2011 (Accepted for Publication).
5. Castro, J., Pimentel, J., Lucena, M., Santos, E., Dermeval, D.: F-STREAM: A Flexible Process for Deriving Architectures from Requirements Models In: 9th International Workshop on System/Software Architectures (IWSSA'11), 2011 (Accepted for publication).
6. Garlan, D., Bachmann, F., Ivers, J., Stafford, J., Bass, L., Clements, P., Merson, P.: Documenting Software Architectures: Views and Beyond, 2nd ed. Addison-Wesley Professional, 2010.
7. Garlan, D., Monroe, R., Wile, D.: Acme: An Architecture Description Interchange Language. In: Proc.CASCON'97, 1997. Toronto, Canada.
8. IStarTool Project: A Model Driven Tool for Modeling i* models. Available at http://portal.cin.ufpe.br/ler/Projects/IstarTool.aspx, June (2011)
9. Kruchten, P., Lago, P., van Vliet, H.: Building up and reasoning about architectural knowledge. In QoSA, pp. 43–58, 2006.
10. OCL 2.0. Available in < http://www.omg.org/spec/OCL/>. Last access in 2011, June.
11. 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. In: Requirements Engineering Journal, 2011 (Accepted for Publication).
12. QVT 1.0 - Query View Transformation. Available in < http://www.omg.org/spec/QVT/1.0/ >. Last access in 2011, June.
13. Yu, E.: Modeling Strategic Relationships for Process Reengineering. Ph.D. thesis. Department of Computer Science, University of Toronto, Canada, 1995.
14. Lucena, M.: STREAM: A Systematic Process to Derive Architectural Models from Requirements Models. Ph.D. Thesis, CIn, Federal University of Pernambuco, Recife, 2010.

# *Nòmos*: from Strategic Dependencies to Obligations

Silvia Ingolfo[1], John Mylopoulos[1], Anna Perini[2], Alberto Siena[1], and Angelo Susi[2]

[1] University of Trento, via Sommarive, 14 - Trento, Italy
{silvia.ingolfo,jm,a.siena}@disi.unitn.it
[2] Fondazione Bruno Kessler, via Sommarive, 18 - Trento, Italy
{perini,susi}@fbk.eu

**Abstract.** New laws are increasingly constraining informations systems.To prevent misuses of the law, requirements engineers are faced with the problem of incorporating legal prescriptions into requirements analysis. *Nòmos* is an extension of *i**, which allows to build models of legal prescriptions alongside intentional elements, and derive this way requirements that at the same time fulfill stakeholder needs and comply with relevant regulations.

## 1 Introduction

Over the past decades, information and communication technologies have steadily evolved, so that the concept of calculus or data processing machine has been replaced by that of a socio-technical system, consisting of software, human and organizational actors and business processes, running on an open network of hardware nodes and fulfilling vital functions for large organizations. Such systems gained the attention of governmental bodies, which are responsible for regulating them through laws, regulations and policies to ensure that they comply with security, privacy, governance and other concerns of importance to citizens and governments alike. The impact of this situation has been immense on Software Engineering as much as on business practices. It has been estimated that in the Healthcare domain, organizations have spent $17.6 billion over a number of years to align their systems and procedures with a single law, the Health Insurance Portability and Accountability Act (HIPAA), introduced in 1996[3]. In the Business domain, it was estimated that organizations spent $5.8 billion in one year alone (2005) to ensure compliance of their reporting and risk management procedures with the Sarbanes-Oxley Act[4]. In this setting, requirements engineers are faced with new challenges in eliciting requirements that at the same time fulfill the needs of stakeholders and are compliant with relevant legal prescriptions. However, unlike stakeholder requirements, which can be validated thanks to the intervention of the stakeholders themselves, requirements introduced for compliance purposes need to objectively evaluated for alignment against their originating prescriptions.

[3] Medical privacy - National standards to protect the privacy of personal health information. Office for Civil Rights, US Department of Health and Human Services, 2000.
[4] Online news published in dmreview.com, november 15, 2004.

## 2 Objectives

The objective of the present work is to support requirements engineers when facing domains, in which laws play a role in defining the requirements for the system-to-be. Actors are subject to legal prescriptions, which they have to adhere to, or, they may decide not to comply. To make a decision – whether to comply or not, and what tasks to undertake – it is necessary to represent both, the applicable prescriptions and the evidence of compliance, if any. Afterwards, a systematic modeling process is needed for going from an initial model of law to a set of domain-specific requirements.

The underlying issue is that the design of requirements, induced by the need to adhere to laws, and requirements, generated by rational agents, is essentially different. Rational agents do what they can to fulfill their goals. With obligations, we are assuming possibly non-cooperating agents who will not necessarily do what they can to fulfill obligations. Our objective is therefore to model the different kind of obligations for and between agents established by the laws and explore designs that include safeguards and incentives that motivate agents to fulfill their obligations.

## 3 Contribution

*Nòmos* [4, 1, 6] is a goal-oriented, law-driven framework intended to generate requirements through which a given information system can comply to a given law. Such requirements are referred to as compliance requirements. *Nòmos* is based on the *i** framework, and exploits its capability to model: the actors of a given domain; their goals and the operationalization of goals into tasks; and the strategic relations among them. In addition, *Nòmos* provides the capability to model law prescriptions and the link between intentional elements and legal elements.

**Concepts**. The core elements of legal prescriptions are *normative propositions* (NPs), which are the most atomic propositions able to carry a normative meaning. NPs contain information concerning: the subject, who is addressed by the NP itself; the legal modality (i.e., whether it is a duty, a privilege and so on); and the description of the object of such modality (i.e., what is actually the duty or privilege). The legal modality is one of the 8 elementary rights, classified by Hohfeld as privilege, claim, power, immunity, no-claim, duty, liability, and disability. **Claim** is the entitlement for a person to have something done from another person, who has therefore a **Duty** of doing it. **Privilege** is the entitlement for a person to discretionally perform an action, regardless of the will of others who may not claim him to perform that action, and have therefore a **No-claim**. **Power** is the (legal) capability to produce changes in the legal system towards another subject, who has the corresponding **Liability**. **Immunity** is the right of being kept untouched from other performing an action, who has therefore a **Disability**. Complex legal prescriptions are created in law documents by structuring NPs through conditions, exceptions, and other conditional elements. Such elements are captured in *Nòmos* by introducing priorities between NPs. For example, a data processor may be allowed (i.e., it has a privilege) to process the data of a subject; but the right of the subject to keep his/her data closed w.r.t. third parties has a higher priority on the privilege, thus constraining the way data is used by the processor.

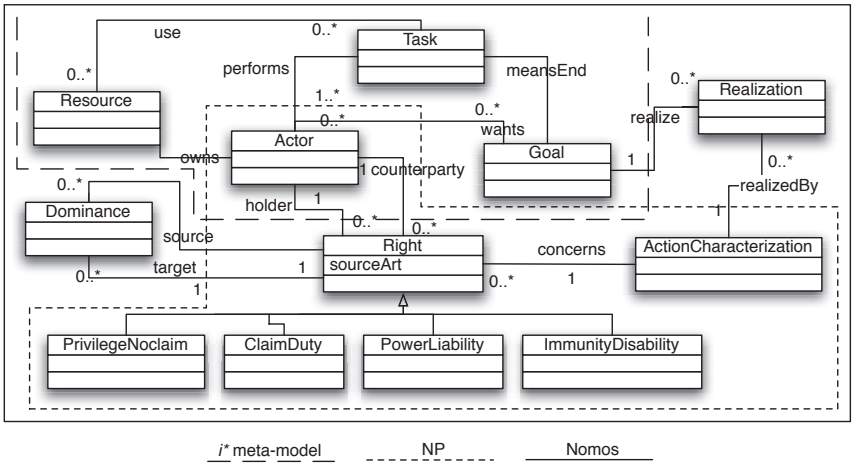



**Figure 1.** The meta-model of the *Nòmos* language.

**Metamodel**. Figure 1 depicts the *Nòmos* meta-model, and shows how it integrates the representation of NPs with the representation of goals. The dashed line contains a part of the *i** meta-model, including to the `Actor` class and its `wants` association with `Goal`. The dotted line contains the elements that form a NP. The join point between the goal-oriented and the legal part of the meta-model is the class `Actor`, which is at the same time stakeholder in the domain and subject of NPs. Actors are associated to the class `Right` by means of the `holder` relation. So on the one hand actors want goals, perform tasks, own resources; on the other hand, they are addressed by rights carried by NPs. Rights also impact on the social interaction of actors - in the Hohfeldian legal taxonomy, rights are related by correlativity relations: for example, if someone has the claim to access some data, then somebody else will have the duty of providing that data. This means that duty and claim are correlatives. Similarly, privilege-noclaim, power-liability, immunity-disability are correlatives: they *describe the same reality from two different points of view*. So instead of defining two separate classes for "duty" or "claim", we have a single class, `ClaimDuty`, which is able to model both. Similarly the classes `PrivilegeNoclaim`, `PowerLiability` and `ImmunityDisability`, each of them sub-class of the abstract class `Right`. Priorities between rights are captured in the meta-model by means of the Dominance class, which connects two rights. The `ActionCharacterization` class contains the actual object of the NP. Such prescribed action is bound to the behaviour of actors by means of the `Realization` class. It specifies that a certain goal is wanted by the actor in order to accomplish the action prescribed by law. For more information on the *Nòmos* meta-model, see [5].

**Visual notation**. Figure 2 exemplifies the *Nòmos* visual notation, as applied in a study about legal compliance of requirements for a healthcare information system [3]. When a patient ([User]) accesses a health care center, at the check-in the EHR of the patient has to be retrieved from the system. In the health care centre accessed by the

patient, the system (a [Local Authority]) executes a query on the local database, and the [S1] service furnishes such data. If the data is not found in the local database, the [Local Authority] forwards the request to the [S2] service, which returns the name of the reference [Certificate Authority]. The Authority is queried to have certified data. But [Certificate Authority] can also be unable to provide the requested data. In this case, the local authority contacts another Local Authority (the actor [Peer Local Authority] in the diagram), which in turn executes a local search or queries its own reference Certificate Authority. If the searched data don't exist in the system, the Local Authority proceeds inserting it, and marking it as "dirty". In this case, after the data insertion, the Local Authority invokes the [S3] service, which broadcasts the data to the whole system. When the broadcast notification is received, each Local Authority updates its local database. However, the privacy law lays down many prescriptions concerning the processing of personal data (in particular, sensitive data) of patients. For example, the law requires the owner's confirmation for the data being processed. In Figure 2, this is depicted by means of the normative proposition [Confirmation as to whether or not personal data concerning him exist]), extracted from article 7.1. The normative proposition is modeled as a claim of the patient, held towards the Local Authority, which has therefore a corresponding duty. This results in two additions to the diagram. The first one concerns the insertion of the data into the local database, and subsequent broadcast to the system. In this case, before the broadcast is executed, it is necessary to obtain the patient's authorization (goal [Ask user authorization]), and to add such information in the broadcast message. The second case concerns the reception of the broadcast system by a Local Authority. In this case, before updating the local data with the received one, the Local Authority must verify that in the broadcast message the authorization to data processing is declared (task [Verify user authorization]). This approach allows for distinguishing goals with respect to their role in achieving compliance: strategic goals are those goals that come from stakeholders and represent needs of the stakeholders; compliance goals are those goals that have been developed to cope with legal prescriptions. In the figure, the goal [Update data locally] is a strategic goal, because it is only due to the reason-to-be of the owning actor; viceversa, [Ask user authorization] is a compliance goal, because it is due to the need of complying with the [Confirmation as to whether or not personal data concerning him exist] claim of the user. So, we can infer that while the first can be dropped according to stakeholders needs, the latter can not, unless its impact on the compliance condition is evaluated.

**Process**. The definition of compliance we have provided before, clearly outlines that reaching compliance is an iterative process that revises the initial requirements model to guarantee that these two properties are met by the final model. The procedure we propose is structured along three logical phases [2]:

1. The *analysis phase* takes as input the model of requirements, expressed as a set of goals to be achieved and tasks to be performed by stakeholders, and a set of NPs with possible irregularities highlighted. We define as irregular a situation where either an element of the model directly violates a norm, or where an element is addressed by a regulation and therefore needs to be checked for compliance.

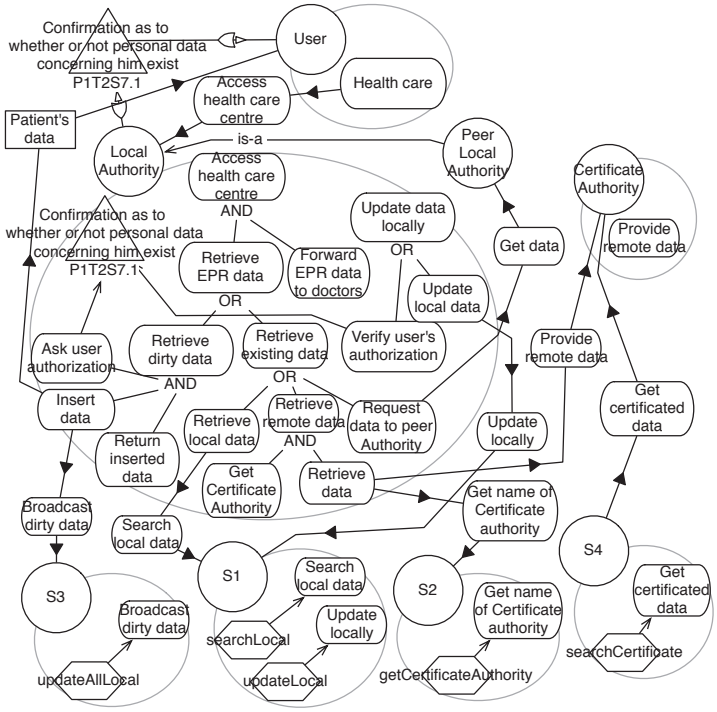



**Figure 2.** A goal model for the demo scenario of the Amico project.

2. The model is then followed by a *compliance check* where the criteria for compliance are evaluated. If the model is compliant, the process returns the model, else we move to the next phase.

3. The *modeling phase* aims at amending a requirements model that is not compliant. The model is expanded and revised by requirement engineers to satisfy the two compliance constraints. A discussion evaluates the acceptability and validity of the solution proposed.

Since this last step modifies the initial model, the process is iterated to ensure that no irregularities have been introduced during modeling. When a cycle is completed without introducing modifications in the solution layer – i.e., in the models – the process ends and compliance is said to be achieved. The key part of the approach is to be able to guarantee that the revisions actually make the system compliant: all the corrections made to the system — as well as the assumptions behind the corrections — are based on the fundamental concept of providing validation through argumentation.

## 4 Conclusions

The *Nòmos* framework has allowed the assessment of a class of problems, which were otherwise difficult to model in vanilla *i**. Specifically, it has allowed to add to the descriptions of legal prescriptions, that are not intentional elements but shape the way intentional elements are designed and analyzed.

## 5 Ongoing and future work

The complexity of laws and regulations dictates the need for new design and analysis techniques for software systems. The *Nòmos* meta-model currently supports a basic representation of conditional elements that are typically found in laws. It is necessary to enrich the expressiveness of this specific aspect of *Nòmos* to capture so-called *legal alternatives* — i.e., alternative ways of being compliant, which are implicit in legal prescriptions.

From a different standpoint, a key issue for the requirements compliance problem concerns the form of evidence provided that indeed a requirements model complies with a given law (fragment). Formal method techniques are generally heavy-weight in the notations they use for modeling laws and requirements, as well as in the reasoning tools they employ to establish compliance, and as a result they have not succeeded in being the proper solution to prove compliance. Alternatively, we aim at establishing compliance through argumentation among the stakeholders who state positions, e.g., "this requirement does not comply with this part of the law" and argue for or against them until (hopefully) consensus is reached.

## References

1. S. Ghanavati, D. Amyot, L. Peyton, A. Siena, A. Perini, and A. Susi. Integrating business strategies with requirement models of legal compliance. *IJEB*, 8(3):260–280, 2010.
2. S. Ingolfo. Establishing compliance of software requirements through argumentation. Master's thesis, University of Trento, Italy, 2011.
3. A. Siena, G. Armellin, G. Mameli, J. Mylopoulos, A. Perini, and A. Susi. Establishing regulatory compliance for information system requirements: An experience report from the health care domain. In J. Parsons, M. Saeki, P. Shoval, C. C. Woo, and Y. Wand, editors, *ER*, volume 6412 of *Lecture Notes in Computer Science*, pages 90–103. Springer, 2010.
4. A. Siena, J. Mylopoulos, A. Perini, and A. Susi. Designing law-compliant software requirements. In *Conceptual Modeling - ER 2009*, pages 472–486, 2009.
5. A. Siena, J. Mylopoulos, A. Perini, and A. Susi. A meta-model for modeling law-compliant requirements. In *2nd International Workshop on Requirements Engineering and Law (Relaw'09)*, Atlanta, USA, September 2009.
6. A. Villafiorita, K. Weldemariam, A. Susi, and A. Siena. Modeling and analysis of laws using bpr and goal-oriented framework. In L. Berntzen, F. Bodendorf, E. Lawrence, M. Perry, and Å. Smedberg, editors, *ICDS*, pages 353–358. IEEE Computer Society, 2010.

# Technology Representation in *i** Modules

Eliel Morales[1], Xavier Franch[2], Alicia Martínez[1], Hugo Estrada[1], and Oscar Pastor[3]

[1] Centro Nacional de Investigación y Desarrollo Tecnológico, Computer Science Department, Cuernavaca, México
{eliel, amartinez, hestrada}@cenidet.edu.mx
[2] GESSI Research Group, Universitat Politècnica de Catalunya, Barcelona, Spain
franch@essi.upc.edu
[3] Universitat Politècnica de València, Valencia, Spain
opastor@dsic.upv.es

**Abstract.** In current business practice an integrated approach to represent at the design level the technological infrastructure that gives support to business processes is needed. We argue that it is highly complex considering technology in terms of specific functionalities from the beginning because these functionalities depend on new business requirements produced continually by internal and external changes. However, business-technology integration has been largely neglected in the modeling of business processes, including *i** models, considering the technological components as highly abstract entities that do not require further description. In this paper, an overview of our approach to deal with technology representation in *i** business process models is presented, which focuses on the identification of quality attributes that are offered by specific technologies and the representation of these technologies using a particular class of *i** module. This approach has been explored in a previous work developing an example of a library in which an automatic identification technology is required to support some specific business processes.

**Keywords:** *i** framework, iStar, *i** modules, technology modeling, business processes, business services

## 1 Introduction

Nowadays, the use of technology is an important aspect for the implementation of efficient business processes, being the indispensable infrastructure for exchanging and persisting information among business actors. In this context, technology can improve the performance of business processes insofar as it is correctly adapted to the organizational context. However, the integration of business and technology at the design level is a current issue that applies both to general software solutions (like ERP systems) and technological infrastructures, such as Radio Frequency IDentification (RFID) or mobile technologies.

We consider that modeling business-technology integration is needed in software and business process design, because embedding technology in the organization can modify the workflow of business processes, and thus the manner in which the analysts should design the software system. However, one issue we found at developing such an integrated modeling technique, is the high complexity of considering technologies

in terms of specific functionalities from the beginning. This is mainly because the number of functionalities and characteristics of technology to be handled can be very high, and the business requirements to which technology should provide support are continually modified by internal and external factors. A natural approach for overcoming this complexity is to rely on goals in the early stages of business and technology infrastructure design, rather than on detailed requirements, functionalities or quality models. We argue that because of its intentional nature, the *i** framework is suitable to be used as a basis for such approach, enabling the analysts to incorporate technological components in the definition of business processes, in order to better consider the possibilities to incorporate the technology at design level. Therefore, we propose a new business model that extends the capabilities of the service-oriented approach for the *i** framework defined in [1], considering technologies as a key element for effective operation of the organization and representing them within *i** modules [2], in order to provide a framework for analysis and design of strategies for integrating business and technology.

The proposed business model deals with technology in a more natural and convenient manner considering technology directly in relation to business requirements independently of their functional capabilities, by means of specifying its quality attributes. We applied this approach in a previous work to a library example [3], in which technology for the automatic identification of items (e.g., books) was required to support specific business processes.

## 2 Objective of the Research

The goal of this paper is to present an *i**-based approach for analysis and design of business processes, considering technology representation as a key modeling aspect of business process models. To achieve it, on one hand our work applies a service-oriented approach as a strategy for managing the complexity and size of *i** business process models [1], the intention for doing this is only to isolate each business process in order to focus on how a given technology may be applied to it, and to analyze contributions and dependencies that are generated when integrating the technology within a business process. On the other hand, our work uses a modular approach for describing technological entities in *i** modules [2], in terms of quality attributes offered, and conditions of operational environment required by technology functioning.

## 3 Scientific Contribution

The main contributions of our approach are: First, the definition of a framework for technology integration analysis and design based on its quality attributes. This framework describes how to model differentiation, compositional, refinement, and integration features of technology. And second, the integration of two approaches (services and modules) for incorporating modularity capabilities into *i** business process models at architectural and detailed modeling levels. In brief, we utilize an *i**

service-oriented approach for modeling the global business architecture, and low level *i** modules for encapsulating the technology specification. Due to space limitations, in this paper we only focus on the first contribution, particularly on the specification of technologies using *i** modules; the reader is referred to our previous works to know the details of this approach.

Our approach include four types of information to specify technology to be included in business process models: a) *differentiation features*, which include information that makes a technology different from others and may serve to assess the usefulness of the technology in the organizational context; b) *compositional features*, which refer to the several components that a particular technology may be composed of; c) *refinement features*, which enable us to deal with the varieties of a given technology, derived from features of specific components of that technology; and d) *integration features*, which enable us to be aware of the requirements that technology is claiming and satisfying in regard to specific business processes. It is important to point out that this paper only focuses on the representation of technological aspects at the design level, and it does not present details about the development method associated to the framework.

The information describing a particular technology is modeled in a technology module. This allows us to create a portfolio of technologies which could be reused in several organizations according to their necessity. Therefore, our approach consists of defining which elements of the *i** metamodel are to be included in this type of modules to consider the features stated above. We use the *i** metamodel proposed in [2], which includes some classes for representing modules in a separate package from the *i** core metamodel (those classes represent the elements to be considered in the module definition), as described in [3]: a) a set of *properties* for representing quality attributes associated to quality characteristics of technology (e. g., efficiency, usability), covering differentiation features; b) a network of *actors* (named technology actors) connected by means of *is-part-of* and *is-a* links, which represents a technology and its basic internal components, covering compositional and refinement features; c) a set of *one-side incoming dependencies*, or *dependencies without depender*, entering into technology actors, which specify the functionalities, resources and behavior that the organization could obtain when using this technology (in particular, for those dependencies whose dependum is a softgoal, there must be a relationship among the softgoal and the quality attributes and their values, e.g. , a softgoal "information be encrypted" may correspond to the quality attribute "encryption algorithm" with value "MD5"); and d) a set of *one-side outgoing dependencies*, or *dependencies without dependee*, stemming from technology actors, which represent relevant external conditions required for the proper functioning of technology. Dimensions c) and d) of technology representation together are for covering integration features.

To sistematize the process of identifying the information to be included in a technology module with regard to *differentiation* and *composition*, the first step is to build a quality model as proposed in [4], which specify a hierarchy of technology characteristics, subcharacteristics, attributes and metrics. Some of this information wont be shown graphically, but will remain as the source of a technology module representation. Fig. 1 shows an example of a technology module for a RFID system,

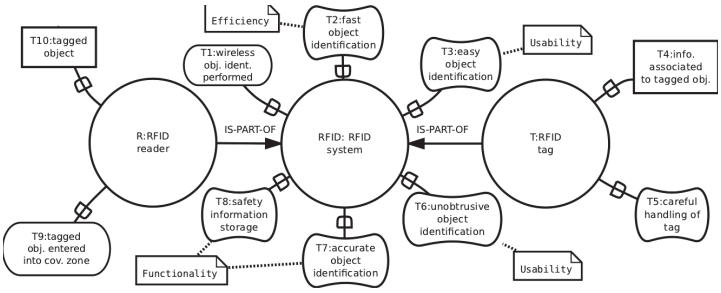



**Fig. 1.** Technology module of a generic RFID system.

annotated with the main quality characteristics specified in the quality model (not shown here) of this technology (functionality, usability, and efficiency) and specifying its quality attributes (`T1-T3`, `T6-T8`) and main components (`R` and `T`). Although in the graphical model is only shown limited information, in the quality model we specified more detailed information, such as the metrics upon which rests the statement of quality attributes. For example, for the attribute stated in softgoal `T2` we idenfied two metrics on which it depends: "speed of the tag response" and "maximum write/read distance of reader." It is convenient that the information to be specified in a technology module and in its underlying quality model is defined with the assistance of an expert, in order to identify the relevant general features and components of a technology, avoid to fail in excessive details or in the lack of meaningful information, and reduce the time required for the description of technology.

In relation to *refinement*, the second step is to define concrete types of technology to be effectively used in organizations. This is done by extending the base technology module into new modules that include new components and dependencies. For example, to specify a passive RFID, we can refine the general RFID technology module by adding it more specific features of passive tags. Fig. 2 depicts the technology module of a passive RFID system, to which we have added two new features: "efficient coverage in short area range" (`A1`), and "reliable functioning in interference environment" (`A2`). Elements (actors and links) from the base module within the refined ones appear in dotted lines as proposed in [5]. It is important to point out that defining more concrete types of technology involves the refinement of the initial quality model into new quality models, by adding new attributes or discarding some of the existent ones; for example, to the attribute `C2` of the active RFID system (Fig. 2) corresponds the addition of the new metrics "sensor integration," "real time location," and "processing capability."

Finally, concerning the *integration* features, the last step is to determine the correspondence among the offering features (incoming dependencies) of technology and its claiming requirements (outgoing dependencies) on one hand, and the business process requirements on the other, in order to obtain an integration model such as the one shown in [3]. Starting with the analysis of technology contributions to each business process, we can explore the way in which the technology features might correspond into business requirements captured in the business process model. Fig. 3 shows the analysis of some contributions of a passive RFID system to a checking-out

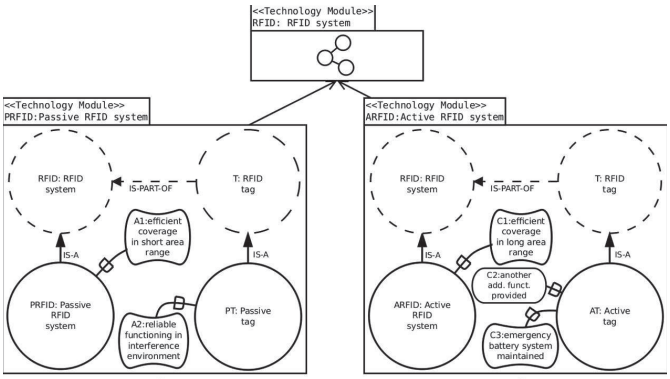



**Fig. 2.** Extending the base RFID module into passive and active RFID modules.

process of a library (an integration model obtained from this analysis has been presented in [3]). Thus, for example, we have that the attribute "fast object identification" (`T2`) of the passive RFID system (in fact, inherited from the general RFID module) has a correspondence with both "fast checking-out" (`D1.1`), required by library patrons, and "fast checking-out management," required by the library, at contributing positively to both of them. Continuing in this way the technology module application to each business process can enable us to think in a passive RFID system for a library.

## 4 Conclusions

In this paper we have presented an approach for modeling technology available for supporting business process. Our approach is based on the concept of module which allows us to create technology modules, i. e. , specifications of technological entities

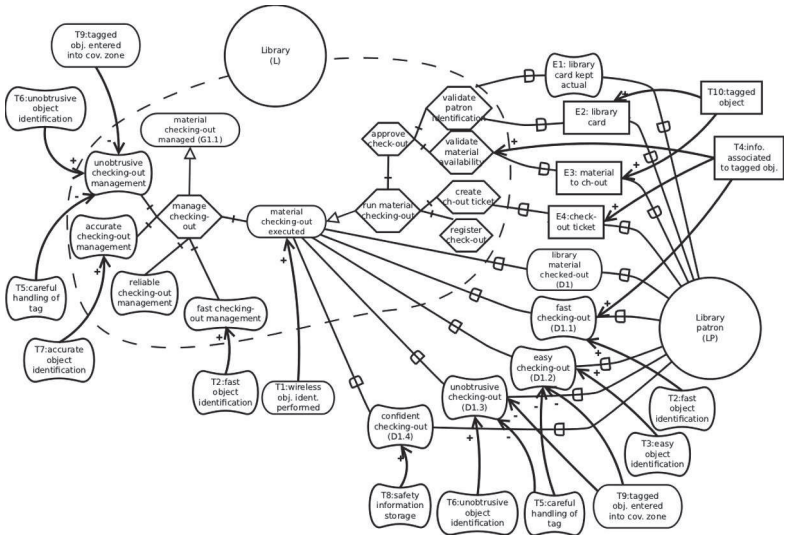



**Fig. 3.** Business-technology correspondence between passive RFID and checking-out process

that can be then integrated into several business processes by means of a correspondence analysis of technology features and business requirements. We have considered four types of information required to specify technology modules (differentiation, composition, refinement, and integration features). For the sake of brevity, we have described just an overview of the approach focusing on technology modeling and suggested the business-technology integration process allowed from this approach.

## 5 Ongoing and Future Work

Other relevant aspects of our current work are: to formalize the notion of integration using the concept of matching as introduced in [6]; to explore the possibility of adding adaptation strategies depending on the results of technology evaluation as done in [7]; to define a portfolio of patterns which describe the impact of technologies using some predefined roles (e.g. , technology provider, technology manager, etc.); to implement a support tool for concepts adopted (module, service, process, etc.); and to evaluate the approach developing a real case study.

**Acknowledgments.** This work has been partially supported by the Spanish project TIN2010-19130-C02-01. Eliel Morales's work has been supported by the CONACYT grant 327254/229895.

## References

1. Estrada, H., Martínez, A., Pastor, O., Mylopoulos, J., Giorgini, P.: Extending Organizational Modeling with Business Services Concepts: An Overview of the Proposed Architecture. In: Parsons, J., Saeki, M., Shoval, P., Woo, C., y Wand, Y. (eds.) ER 2010. LNCS, vol. 6412, pp. 483-488. Springer Berlin / Heidelberg (2010).
2. Franch, X.: Incorporating modules into the *i** framework. In: Pernici, B. (ed.) CAiSE 2010. LNCS, vol. 6051, pp. 439-454. Springer Berlin / Heidelberg (2010).
3. Morales, E., Franch, X., Martínez, A., Estrada, H.: Considering Technology Representation in Service-Oriented Business Models. Presented at the REFS 2011: The 5th International IEEE Workshop on Requirements Engineering for Services (2011).
4. Franch, X., Carvallo, J.P.: Using Quality Models in Software Package Selection. IEEE Softw. 20, 34–41 (2003).
5. López, L., Franch, X., Marco, J.: Defining Inheritance in *i** at the Level of SR Intentional Elements. iStar 2008. pp. 71-74 (2008).
6. Franch, X.: On the Lightweight Use of Goal-Oriented Models for Software Package Selection. In: Pastor, O. y Falcão e Cunha, J. (eds.) CAiSE 2005. LNCS, vol. 3520, pp. 1-15. Springer Berlin / Heidelberg (2005).
7. Dalpiaz, F., Giorgini, P., Mylopoulos, J.: An Architecture for Requirements-Driven Self-reconfiguration. CAiSE 2009. LNCS, vol. 5565. pp. 246–260. Springer-Verlag, Berlin, Heidelberg (2009).

# An extension of *i** to Model CSCW Requirements Applied to a Collaborative Conference Review System

Miguel A. Teruel, Elena Navarro, Víctor López-Jaquero, Francisco Montero, Pascual González

LoUISE Research Group, I3A,
University of Castilla- La Mancha, Spain
{miguel, enavarro, victor, fmontero, pgonzalez}@dsi.uclm.es

**Abstract.** In collaborative systems, users work together in order to collaborate, communicate and coordinate each other. To perform these tasks, users should be aware of other user's actions, usually by means of a set of awareness techniques. In this paper, CSRML (*Collaborative System Requirements Modelling Language*) is presented as an extension of *i** to deal with the specification of the CSCW requirements. In these systems collaboration and awareness of other users' presence / actions are paramount. We apply CSRML to a conference review system, where papers are reviewed in a collaborative way.

Keywords: Collaborative systems, Awareness, Requirements Engineering, Goal-Oriented, *i**, CSRML

## 1 Introduction

Requirements elicitation can be considered the cornerstone to achieve the quality of the developed systems. Failing in accomplishing this phase can make the rest of the development process also fail, with the consequent cost in terms of time and money. Therefore, a correct requirements specification is paramount for any kind of system.

As in traditional single-user systems, CSCW (Computer Supported Cooperative Work) systems are not exempt from this need. They are a special kind of software whose users can perform collaboration, communication and coordination tasks. These systems have to be specified by using a special set of requirements, usually of a non-functional nature. They usually result from the users' need of being aware of the presence and activity of other remote or local users, with who they perform the above mentioned collaborative tasks. This is the so-called *Workspace Awareness*, which can be defined as the up-to-the-moment understanding of another person's interaction within a shared workspace.

Then, a proper specification of the system, identifying clearly the requirements of the system-to-be, specially the awareness requirements, is one of the first steps to overcome this problem.

## 2 Objectives of the research

In previous works [1], we analyzed which requirements engineering (RE) technique: Goal-Oriented (GO), Use Cases or Viewpoints is more appropriate to specify the requirements of collaborative systems, and we found that GO provides more facilities to model the requirements of this kind of systems. Once we determined GO as the most suitable technique, we analyzed which GO approach deals with CSCW systems in a better way [2]. The analyzed approaches were NFR Framework, *i** Framework and KAOS Methodology for the specification of collaborative systems, paying special attention to awareness requirements. As a result of this experiment, we concluded that the analyzed GO approaches are not fully appropriate to model collaborative system characteristics and its relationships with awareness and quality requirements. These conclusions, together with the results of [1] support our initial hypothesis: a RE technique to address the problems detected during this study is required. This technique should adopt some features from the analyzed GO approaches and should cover the lack of expressiveness in certain aspects that current GO techniques present. This constitutes the main aim of this work: to adapt/extend a GO notation for this kind of systems. Concretely, and according to the conclusions of our previous study [2] the most appropriate approach to deal with this kind of systems is *i**. Therefore, in this paper CSRML (*Collaborative Systems Requirements Modelling Language*) [3] is described, by extending *i** to provide the required expressiveness to model the special characteristics of CSCW stakeholder requirements.

## 3 Scientific contributions

Because of the special kind of requirements of CSCW systems, we present CSRML as an extension of *i** that includes some elements for modelling the special collaboration features of CSCW systems. The elements of CSRML (Fig. 1), excluding those whose meaning is the same as in *i*,* are:

- *Role*: A role is a designator for a set of related tasks to be carried out. The difference between *i** and CSRML is that an actor playing a role can participate in individual or collaborative tasks (through participation links) and can be the responsible for the accomplishment of a goal (through responsibility links). Thus, an actor can both dynamically change the roles it plays, and simultaneously play several roles. In addition, the graphical notation is also different from the *i** role (the concept of role/actor boundary is not used in CSRML).
- *Actor*: An actor is a user, program, or entity with certain acquired capabilities (skills, category, and so forth) that can play a role in executing (using devices) or being responsible for actions. An actor has to play a role (specified by means of a playing link, see Fig. 1) in order to participate in the system.
- *Task:* The concept of task in CSRML is the same as in *i**. They only differ in the introduced notation to define the importance of a task: one, two or three exclamation signs, depending on the importance of the task. Two kinds of CSRML tasks have been identified:

- Abstract task: This kind of task consists in an abstraction of a set of concrete tasks and, possibly, other elements. We are not able to assign participation links directly to this kind of tasks.
- Concrete task: These are the tasks the participants are involved to. The abstract tasks are refined in these ones. Participants will be assigned to the task through participation links. There are four types of these tasks:
  - *Individual task* is a task that an actor can perform without any kind of interaction with other actors.
  - *Collaboration / Communication / Coordination task* two or more actors are involved in order to perform any kind of collaboration / communication / coordination among them.

- *Awareness softgoal*: CSRML refines the *i** concept of softgoal into a new specialization: awareness softgoal, that represents a special need of perception of other user's presence / actions, without which the task the user wants to perform would be affected negatively or even could not be done.
- *Awareness resource*: This special kind of resource corresponds to an implementation or a design solution to accomplish an awareness softgoal.
- *Playing link*: A playing link is used to represent when an actor assumes a role. This link has a guard condition that represent when a role can be played by an actor.
- *Participation link*: A participation link denotes who are involved in a task. This link has an attribute to specify its cardinality, i.e., the number of users that can be involved in a task.
- *Responsibility link*: A responsibility link assigns a role (played by an actor) to a (soft)goal or task. This link represents who is the stakeholder responsible for a goal/task accomplishment. It is not necessary that this stakeholder is involved in the goal sub-tasks. Nevertheless, if the role is responsible for a goal or task, this role is also responsible for the elements it is divided into, unless a responsibility link reaches one of the elements it is divided into.

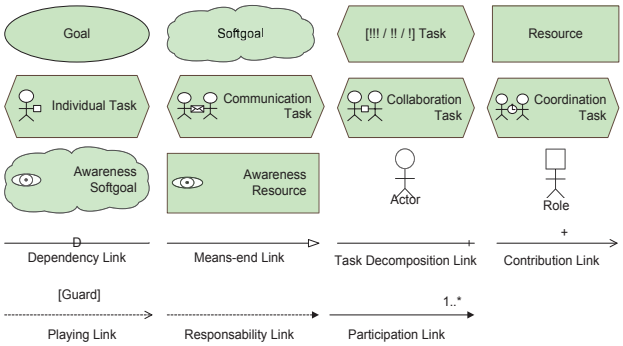



**Fig. 1.** CSRML elements

It is worth noting an additional difference between CSRML and *i**: CSRML is practically hierarchical (see Fig. 2 (a) (d)). Thus, it fosters the scalability of the model

created by using this notation. In a first level, we have the *Responsibility diagram*, in which the system's main goal is decomposed into main tasks and quality softgoals. Also, in this diagram, the goals and tasks responsibilities are defined.

In a second level appear *Task refinement diagrams*, in which the system's main tasks are decomposed into new goals, softgoals, tasks and resources, and roles are assigned to tasks. This constitutes another difference between CSRML and *i**. Because CSRML has been thought for collaborative systems, *i** boundaries for actors/roles were discarded, since they would not support assigning a task to more than one role. In addition, the *Quality factors diagram* completes the system specification showing the quality softgoals and the elements that contribute to their accomplishment.

### 3.1 Case Study: Collaborative Conference Review System with CSRML

To check out the validity of our proposal, we are going to use the CSRML notation to model a case study based on a collaborative conference review system in order to illustrate its expressiveness capacity for CSCW systems. First, in Fig. 2 (a), we can see the *system goals diagram*, in which the system main goals are defined. As shown, we are going to achieve the system goals by means of the realization of the system's main task: *the preparation of the review process of papers for a conference by using techniques of collaboration among users*.

Fig. 2 (b) shows the *responsibility diagram* with the main system's task and its decomposition in quality softgoals and tasks. In this figure, it can be observed that the use of *responsibility links* shows who is responsible for goals and tasks. Note that if a role is responsible for a goal or task, this role is also responsible for the elements it is divided into, unless a responsibility link is specified to one of the elements it is divided into. Also, the *playing links* are used to represent the condition that must be met for an actor to play a role. For the sake of model readability, a task decomposition will be shown in Fig. 2 (c).

Fig. 2 (c) depicts *Papers review* task refinement diagram. In this figure, tasks are refined into more specific ones or new goals, until individual or collaborative (collaboration, coordination or communication) tasks are specified. It can be observed that for collaborative tasks, more than an actor (playing a role) is involved through *participation links*. This figure includes two awareness softgoals. One of them is related to the knowledge of who reviews each paragraph, and the other one corresponds to the use of remote cursors. In this figure, different cardinalities for *participation links* are used. For example, for *Paragraph review*, three experts must participate. Also, this figure illustrates some degrees of priority that can be assigned to tasks: normal, high ([!]), very high ([!!]) and highest ([!!!]).

Finally, Fig. 2 (d) depicts the *Quality factors diagram*. In this model, the quality factors that contribute to achieve the conference review with a high quality level are shown. These factors are represented as softgoals and they are related to the main quality softgoal by means of contribution links with positive contributions. The achievement of all these quality softgoals is obtained in different ways. For instance, the *Helpfulness* softgoal is achieved by means of an awareness softgoal and its corresponding awareness resource consists in a remote cursors implementation.

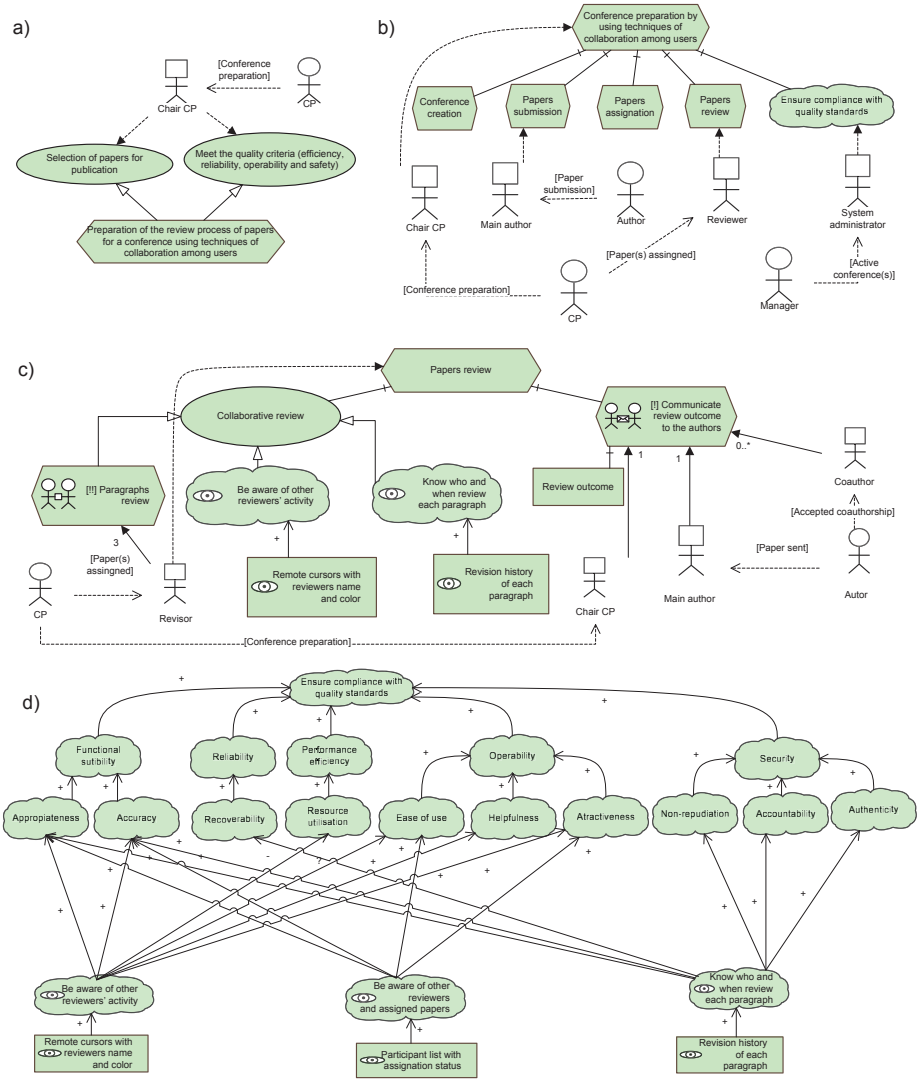



**Fig. 2.** (a) System goals diagram (b) Responsibility diagram (c) *Papers review* task refinement diagram (d) Quality factors diagram

## 4 Conclusions and future work

We found out in two previous works [1,2] that Goal-Oriented Requirement Engineering techniques (and especially *i**) can be used to deal with collaborative systems requirements modelling. Nevertheless, we also found out that this kind of specifications suffer from an important lack of expressiveness for some characteristics related to user collaboration, awareness representation or quality factors. To address

these shortcomings, we propose CSRML, an extension of *i** Goal-Oriented specification to model CSCW systems requirements.

In order to check out the suitability of this language, we have modelled a collaborative system. For the sake of clarity, in this paper an excerpt of this system consisting in a conference preparation system with collaborative reviews has been presented. This case study was modelled because it has a set of characteristics that were hard or impossible to be represented with the original *i** notation. These characteristics were properly described by introducing a set of new elements and links into *i** notation. The quality and awareness representation has been made possible by means of new awareness elements and the inclusion of a new set of diagrams in order to provide some structure to the specification.

Resuming, CSRML helps in improving understandability [3] and maintainability of requirements models for CSCW systems by adding new elements and relationships to *i**. These new elements facilitate the specification of awareness requirements, which are paramount in the development process of any CSCW systems.

One of our ongoing works is closely related to the development of e-learning systems. Since LoUISE research group has been working during the last years in this kind of systems, several patterns have been described up to date. One of the main problems they have is that they have been specified in an informal way that cannot be easily reused for the specification of different systems. Therefore, we are studying how CSRML can be used to improve their specification.

Another future work consist in a validation procedure to validate the developed CSCW system against the initial set of requirements specified with CSRML and his compliance with the ISO 25010 quality in use factors.

## Acknowledgements

This work has been supported by the following projects: (PEII09-0054-9581) from the Junta de Comunidades de Castilla-La Mancha and (DESACO, TIN2008-06596-C02-01) from the Spanish Government.

## References

[1] M.A. Teruel, E. Navarro, V. López-Jaquero, F. Montero, and P. González, "An Empirical Evaluation of Requirement Engineering Techniques for Collaborative Systems," *15th Int. Conf. on Evaluation and Assessment in Software Engineering*, Durham, UK: 2011.

[2] M.A. Teruel, E. Navarro, V. López-Jaquero, F. Montero, and P. González, "A Comparative of Goal-Oriented Approaches to Modelling Requirements for Collaborative Systems," *6th Int. Conf. on Evaluation of Novel Software Approaches to Software Engineering*, Beijing, China: 2011.

[3] M.A. Teruel, E. Navarro, V. López-Jaquero, F. Montero, and P. González, "CSRML: A Goal-Oriented Approach to Model Requirements for Collaborative Systems," *30th Int. Conf. on Conceptual Modeling*, Brussels, Belgium: Springer Berlin, 2011.

[4] M.A. Teruel, E. Navarro, V. López-Jaquero, F. Montero, and P. González, "Assessing the Understandability of Collaborative Systems Requirements Notations: an Empirical Study," *1st Int. Workshop on Empirical Requirements Engineering*, Trento, Italy: 2011.

# Modeling Strategy Maps and Balanced Scorecards using iStar

Constantinos Giannoulis, Jelena Zdravkovic

Department of Computer and Systems Sciences (DSV),
Stockholm University
Forum 100, SE-164 40 Kista, Sweden
constantinos,jelenaz@dsv.su.se
http://www.dsv.su.se

**Abstract.** Aligning business strategy to enterprise models requires explicit models from both areas, mapped to each other. Mapping existing business strategy definition approaches to requirement engineering practices improves strategy dissemination towards development. In this paper we present an illustration of such a mapping using the Strategy Maps and Balanced Scorecards as a business strategy approach and iStar (i*) as a requirements engineering practice exemplified using a case scenario.

**Keywords:** business strategy, strategy maps, balanced scorecards, iStar, requirements, SMBSC

## 1 Introduction

Organizations aim at enabling the communication of business strategy by linking decision makers to practitioners, to align people, products and services with long-term visions, and help in ensuring the strategy's successful implementation. Various alignment efforts have addressed the alignment between business strategy and requirements for system development in accordance to stakeholders needs and intentions [1–4]. However, there still exists an understanding gap between the business world and the IT world, which constitutes business strategy unknown, thus hindering business-IT alignment [3, 5, 6].

To address this gap, in a previous study [7], we have developed a meta-model of Strategy Maps & Balanced Scorecards [8] (named SMBSC onwards). Consequently, we aim to explore how can our meta-model influence the application of business-IT alignment methods, which requires defining mappings of our meta-model towards distinct requirement languages to complement alignment methods. Therefore, in this paper we extend our meta-model by providing mappings to i* [9], a goal modeling technique used in requirements engineering, and particularly to the unified meta-model proposed by Lucena et al [10]. In contrast to Babar et al [11], where mappings were provided based on the original form of SMBSC and constructs of i* [9], we have chosen sources with a formal basis.

Section 2 presents our proposed mappings, section 3 illustrates how the mappings have been used in a case scenario and section 4 provides our conclusions and sets the steps forward.

## 2 Mapping of SMBSC to i*

In this section we present how concepts of SMBSC can be mapped to i* in respect to their meta-models.

The *Strategy Map* class is used to capture the complete SMBSC including all causality relationships among all goals across an organization, therefore, using i* to capture the complete SMBSC requires both the *Strategic Dependency model (SD)* as well as the *Strategic Rationale Model (SR)* [1], which capture respectively, all the dependencies within the organizational context modeled as well as all the intentional elements.

The notion of grouping is present in both meta-models. In SMBSC there exists a *Group* class that captures all groupings of goals, where the highest level of grouping is among the four perspectives expressed through a specialization to a *Perspective* class. Other groupings within each perspective are captured by a recursive association, enhanced by constraints that make sure groups form a tree structure through nesting. In i* the notion of grouping is not present as such, however, the abstract notion of an actor is used to include the relevant intentional elements and there is a distinction between the dependencies among actors (SD) and the detailed rational of their dependencies (SR). Therefore, the notion of actor in i* can be related to the group of SMBSC and instead of constructing actor models, we are constructing group models. The i* actor can facilitate the Group class of SMBSC by extending its boundaries to facilitate organizational groupings, hence represent an organizational entity with defined dependencies. Therefore, for SMBSC, the SD is fixed with four abstract actors which refer to the organizational perspectives of SMBSC. The dependencies between those perspectives adhere to the i* meta model (the *DependencyRelationship* class); one is a dependee and the other is the depender. Similarly, dependencies may exist for any subgrouping (various *Group Types*), across the actors defined within actors that represent different perspectives.

In SMBSC the *Goal* class encompasses all goals defined across the four perspectives which are not necessarily measurable. Measurable goals extend the strategy map into balanced scorecards. A measurable goal, which is an objective in SMBSC, is also a goal in i*, whereas a non-measurable goal, which is not an objective in SMBSC, is a soft-goal in i*. The *Milestone* class, as well as the *Target* class, are intermediate states of an objective, usually related to some deadline or some value as mandated by the *Measure* class, used to demonstrate an objective's achievement. Both milestone and target, in conjunction to measure, are expressed as i* goals. The *Initiative* class in SMBSC can be either a *Task* or a *Plan* or a *Resource (consumed or produced)* in i*. In SMBSC,

[1] The instantiation of the *Dependency* class and the *InternalElement* class indicates the existence of the SD model the SR model respectively.

the associations linking goals (*influences*, *is influenced by*) adhere to the *InternalElementRelationship* of i* in a constraint manner. i* goals originating from SMBSC objectives are linked to i* goals originating from SMBSC milestones and targets through *MeansEnd*, i* goals originating from SMBSC objectives can be linked to i* soft-goals originating from SMBSC goals (non-measurable) through *MeansEnd*, and the opposite, i* tasks or plans (not resources) originating from SMBSC initiatives can be linked to i* goals originating from SMBSC milestones and targets, not objectives, through *MeansEnd*.

In SMBSC a theme captures a particular selection of goals across the four perspectives, with significant interest. This can be expressed in i* using the *IntentionalType* of the *Dependency* class (critical, open, committed). Therefore, all dependencies of type critical constitute a theme. Similarly to classes and associations, constraints defined for the SMBSC meta-model have also been considered when defining the mappings. Due to space limitations we present an example of two constraints for the goal class.

In SMBSC, every goal included in a theme is also included in the strategy map for which the theme is defined. In i*, a Theme consists of all the nodes whose DependencyRelationship is of critical DependencyStrength (SD models), which when expanded they include InternalElements, such as goals. Therefore, goals included in actors who are related with critical dependencies belong to a theme and also belong to the SR and SD model, ergo to the complete Strategy Map, as mapped earlier. In SMBSC, goal influences are restricted according to the perspective they belong to, therefore, financial goals can be influenced by customer goals and other financial goals while they can only influence other financial goals only. Therefore, financial goals can be dependers to customer goals and other financial goals while they can be dependees to other financial goals only. Customer goals can be influenced by internal goals and other customer goals while they can influence financial goals and other customer goals. Therefore, customer goals can be dependers to internal goals and other customer goals while they can be dependees to financial goals and other customer goals. Internal goals can be influenced by learning and growth goals and other internal goals while they can influence customer goals and other internal goals. Therefore, internal goals can be dependers to learning and growth goals and other internal goals while they can be dependees to customer goals and other internal goals. Learning and growth goals can be influenced only by other learning and growth goals while they can influence internal goals and other learning and growth goals. Therefore, learning and growth goals can be dependers to only other learning and growth goals while they can be dependees to internal goals and other learning and growth goals. In i* this is captured by the fixed dependencies among perspectives.

## 3 Example case: ABB's SMBSC in i*

To illustrate the applicability of our mappings, we use the case of ABB Industrie AG [12] modeled using the SMBSC meta-model [7] and due to space limitations we present the Potential perspective (Learning and Growth). The potential perspective includes two strategic goals. The goal, *our employees are competent and*

*motivated*, is measured by the *average number of jobs to which an employee can be assigned*, has milestones *5 for the end of 1st year* and *7 for the end of 2nd year* and targets at *9 for the end of 3rd year*. The goal, *we pursue a proactive human resource management*, is measured by the *average number of months needed until free resources are available to fulfill a new task*, has milestones *5 for the end of 1st year* and *3 for the end of 2nd year* and targets at *2 for the end of 3rd year*.

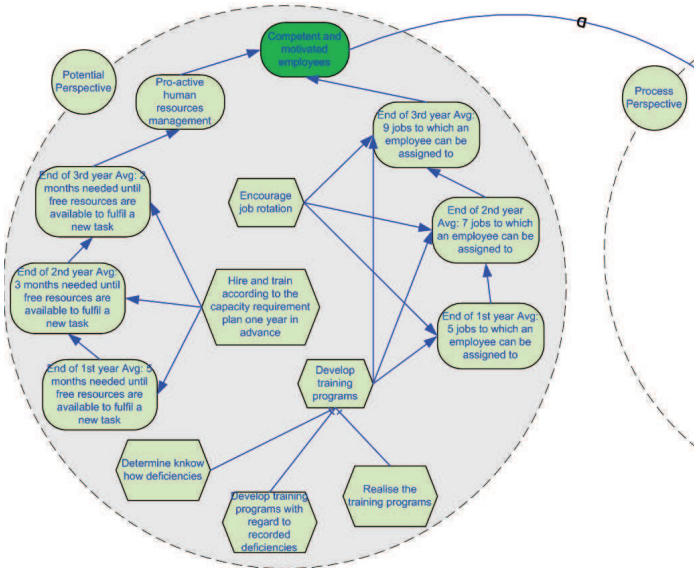



**Fig. 1. Fig. 1.** The SR model: The Potential perspective for the ABB case scenario.

Based on the aforementioned mappings, the SD model consists of the four perspectives of ABB Industrie which are presented as actors along with their fixed dependencies following the constraints exemplified.

For the SR model, the SD model is expanded to provide the *InternalElements* of each perspective. Milestones and targets of SMBSC are mapped to i* goals and are linked through mean-end links both between themselves as well as with i* goals originating from the SMBSC goals. For the potential perspective (figure 1) the goal *End of 1st year Avg: 5 jobs to which an employee can be assigned to* originates from the SMBSC milestone *End of 1st year: 5*, where the SMBSC measure is the *Average number of jobs to which an employee can be assigned.* Therefore, this goal is means to the end expressed by the goal *End of 2nd year Avg: 7 jobs to which an employee can be assigned to*, which originates from the SMBSC milestone *End of 2nd year: 7*, where the SMBSC measure is the *Average number of jobs to which an employee can be assigned.* Consequently, this goal is the means to the end expressed by the goal *End of 3rd year Avg: 9 jobs to which an employee can be assigned to*, which originates from the SMBSC target *End of 3rdd year: 9*, where the SMBSC measure is the *Average number of jobs to which an employee can be assigned.* Finally, this goal is one of two means to the end expressed by the goal *Competent and motivated employees*, which originates

from the SMBSC goal *Our employees are competent and motivated*, where the SMBSC measure is the *Average number of jobs to which an employee can be assigned*.

Initiatives (named *Strategic Programs* in [12]) are mapped to i* tasks and are linked through mean-end links only to i* goals originating from SMBSC milestones and targets. *Actions* in [12]) included in initiatives are mapped to i* tasks and are linked through decomposition to i* tasks originating from SMBSC initiatives. For example, in the potential perspective, the task *Develop training programs* originates from the SMBSC initiative *Development of training programs*. The task is the means to the ends expressed by the goals *End of 1st year Avg: 5 jobs to which an employee can be assigned to*, *End of 2nd year Avg: 7 jobs to which an employee can be assigned to* and *End of 3rd year Avg: 9 jobs to which an employee can be assigned to*. Additionally, the task *Develop training programs* is linked through decomposition to the tasks *Determine the know-how deficiencies*, *Develop training programs with regard to recorded deficiencies* and *Realize the training programs* which respectively originate from the SMBSC actions originating from the *Determination of know-how deficiencies*, *Development of training programs with regard to recorded deficiencies* and *Realization of the training programs*.

## 4 Conclusions and future work

In this paper, despite the different purpose and domains of use of SMBSC and i*, we have provided concept mappings between the two meta-models exemplified with an illustrative case scenario for which we have successfully modeled SMBSC using i* (figure 1).

Using i* to model SMBSC allows transition to requirements engineering supporting business-IT alignment methods. When i* is used during the early phase of requirements engineering, it can be enriched with stakeholders' intentional elements from SMBSC. The unified i* meta-model supports *OrDecomposition*, which can facilitate SMBSC with alternatives for initiatives. *Contribution* links provided by the unified i* meta-model (*enough, positive, notenough, negative*) could be used among goals and soft-goals in SMBSC allowing the identification of possible conflicts or synergies among the goals set, which is currently not present. By using the i* unified meta-model, our mappings are applicable to two variants of i*, resulting into greater applicability.

Additionally, the mappings have brought up some unaddressable issues, which could be used to extend the i* unified meta-model. (a)The class *Measure* in SMBSC has not been mapped directly to any notion or construct of the i* meta-model but it has been used implicitly when expressing i* goals originating from SMBSC milestones and targets. (b)The links between milestones, targets and objectives; in SMBSC there is a sequence expressed between these notions and i* does not support any kind of timeliness. The result is that each task is linked to every goal originating from SMBSC milestones and targets through means end links. The introduction of a *Precedence* link as a construct to address the issue of sequences and priorities has been proposed in [13], which would allow

timely appropriate links between tasks and goals originating from SMBSC milestones and targets. (c)According to the unified i* meta-model, means-end links are allowed between goal and goals in both variations of i* described, however, the i* guide [14] explicitly mentions that means-end links between goals is wrong, rather only tasks are linked through means-end to goals.

Finally, our future research steps include the evaluation of our mappings within a case where SMBSC will be the starting point but it will involve the early phase of requirements engineering, to illustrate the potential for traceability from strategy to concrete requirements.

## References

1. Thevenet, L.H., Salinesi, C.: Aligning IS to organization's strategy: the INSTAL method. In: 19th International Conference on Advanced Information Systems Engineering (CaiSE'07), (2007)
2. Bleistein, S.J., Cox, K., Verner, J.: Validating strategic alignment of organizational IT requirements using goal modeling and problem diagrams. J. Systems and Software. 79, pp. 362–378 (2006)
3. Singh, S.N., Woo, C.:Investigating business-IT alignment through multi-disciplinary goal concepts: Requirements Engineering, 14, pp. 177–207 (2009)
4. van der Raadt, B., Gordijn, J., Yu, E.: Exploring web services ideas from a business value perspective. In: 13th IEEE International Conference on Requirements Engineering (RE05), pp. 53-62, IEEE CS (2005)
5. Chan, Y.E., Horner, R.B.: IT alignment: what have we learned? Journal of Information Technology, 22, 4, pp. 297 (2007)
6. Luftman, J.: Assessing business-IT alignment maturity: Communications of the Association for Information Systems, 4, (2000) article 14
7. Giannoulis, C., Petit, M., Zdravkovic, J.: Modeling Business Strategy: A Metamodel of Strategy Maps and Balance Scorecards. In: 5th IEEE International Conference on Research Challenges in Information Science (RCIS2011) (2011)
8. Kaplan R.S., Norton D.P.: Strategy Maps: Converting Intangible Assets into Tangible Outcomes. Harvard Business School Press, Boston (2004)
9. Yu, E. Modeling strategic relationships for process reengineering: PhD Thesis, Department of Computer Science, University of Toronto, (1995)
10. Lucena, M., Santos, E., Silva, C., Alencar, F., Silva, M.J., Castro, J.: Towards a unified metamodel for i*. In: 2nd International Conference on Research Challenges in Information Science (RCIS 2008), pp.237–246 (2008)
11. Babar, A., Zowghi, D., Chew, E.: Using Goals to Model Strategy Map for Business IT Alignment. In: 5th International Workshop on Business/IT Alignment and Interoperability (BUSITAL 2010), pp. 1630 (2010)
12. Ahn, H.: Applying the Balanced Scorecard Concept: An Experience Report. Long Range Planning, vol. 34, pp. 441-461 (2001)
13. Liaskos, S., Mylopoulos, J.: On Temporally Annotating Goal Models. In: Proceedings of the 4th International i* Workshop (iStar2010), CEUR vol 586, pp. 62-66 (2010)
14. i* wiki, *http://istar.rwth-aachen.de/tiki-index.php?page_ref_id=271* (last accessed on 05-06-2011)

# Capturing Contextual Variability in *i** Models

Alexei Lapouchnian[1] and John Mylopoulos[2]

[1] Department of Computer Science, University of Toronto, Canada
alexei@cs.toronto.edu
[2] Department of Information Engineering and Computer Science, University of Trento, Italy
jm@disi.unitn.it

**Abstract.** Exploration and analysis of alternatives is one of the main activities in requirements engineering, both in early and in late requirements phases. While *i** and *i**-derived modeling notations provide facilities for capturing certain types of variability, domain properties (and other external influences) and their effects on *i** models cannot be easily modeled. In this paper, we propose to explore how our previous work on context-dependent goal models can be extended to support *i**. Here, we examine how *i** modeling can benefit from both monitorable (i.e., defined through real-world phenomena) and non-monitorable (e.g., viewpoints, versions, etc.) contexts defined using our context framework.

## 1 Introduction

*i** is an agent-oriented modeling framework that centers on the notions of intelligent actor and intentional dependency. The Strategic Dependency (SD) model of *i** focuses on representing the relevant actors in the organization together with their intentional dependencies, while the Strategic Rationale (SR) model captures the rationale behind the processes in organizations from the point of view of participating actors.

Variability in requirements and in design has been identified as crucial for developing future software systems [4,5]. Moreover, flexible, robust, adaptive, mobile and pervasive applications are expected to account for the properties of (as well as to adapt to changes in) their environments. Thus, modeling variability in the system environment and its effects on requirements and on other types of models is a highly desirable feature of a modeling framework. However, *i** does not support capturing of how domain variations affect its diagrams. This leads to two situations. First, an oversimplification of the diagrams through the assumption of domain uniformity with the hope of producing an *i** model that is adequate for the most instances of a problem. Second, the production of multiple *i** models to accommodate all domain variations. The former case leads to models that fail to account for the richness of domain variations, while the latter introduces serious model management problems due to the need to oversee large numbers of models as well as to capture their relationships. In this paper, we adapt the ideas from [3] to the *i** modeling framework and propose an approach that uses *contexts* to structure domain variability and to concisely represent and analyze the variations in *i** models resulting from this domain variability as well as from other external factors such as viewpoints, etc. Using the proposed approach,

we are able to capture in a single context-parameterized model how varying domain characteristics affect stakeholders, their goals, and their intentional dependencies.

## 2 Research Objectives

While *i** has capabilities to represent certain types of variations in it diagrams, they are not adequate to capture the effects of domain variability on the models. In SD models, one cannot state that the actors and dependencies appear in the model only in certain circumstances. E.g., if we look at a system where a Distributor accepts orders from Customers, fulfills them through Suppliers, and then ships those orders through Shipping companies, it is not possible to capture in a single SD diagram the fact that in the case of an international order, another actor comes in, the Customs Broker, through which the Distributor clears the order before shipping it.

In SR models, OR decompositions (or means-ends links) are the tools to represent variation points. Still, they are not enough to capture all the possible effects that domain variations can have on SR models, such as varying sets of top-level actor goals, different goal refinements, and changing evaluations of alternatives w.r.t. softgoals.

We propose to adapt the ideas of [3] to *i** and use contexts as a way to parameterize *i** models in order identify changes due to such external factors. A *context* is an abstraction over relevant domain properties. *internationalOrder*, *largeOrder*, *importantCustomer* are examples of contexts that influence the *i** diagrams modeling the Distributor system. Additionally, we show how related contexts can be organized into inheritance hierarchies and how this simplifies the modeling process as well as discuss the notion of visibility of model elements as a way to combine model variants.

## 3 The Context Framework for *i**

### 3.1 Visibility of Model Elements, Contexts, and Context Inheritance

The main idea of our context framework [3] is that models (e.g., ER and *i** diagrams, or knowledge bases) are viewed as collections of elements (i.e ., nodes, edged, facts), some associated with conditions that describe when the elements are *visible* – i.e., present in the model. These conditions are captured through (possibly many) sets of *contextual tags* assigned to model elements. The tags model the (many) *contexts* in which the elements are *valid*. E.g., the tag assignment *{{largeOrder}, {mediumOrder, importantCustomer}}* indicates that some model element is visible either when the order is large or with a medium-sized order from an important customer. The absence of any condition indicates that a model element is valid in all contexts (visible in all model variants). Each contextual tag has a definition describing when it is *active*. Thus, a set of tags can be viewed as a propositional DNF formula. Through their definitions, contexts can be monitored in the environment of the system to determine when they are active. Therefore, a context-parameterized model will be changing as the environment conditions change. In [3], which presents the details of this formal visibility framework, we applied the framework to goal models, while here we do so

for *i**. We are interested in capturing the effects of two types of external factors on *i** models: monitorable contexts and changes due to viewpoints, model versions, etc.

For added flexibility, the formal context framework supports non-monotonic inheritance of contextual tags. This way, the modeler can declare that a new contextual tag (e.g., *mediumOrder*) inherits from an existing one (e.g., *substantialOrder)*. Thus, model elements tagged with *mediumOrder* (i.e., valid/visible in that context) are automatically tagged with *substantialOrder*. Additional model elements can be explicitly assigned the derived tag, while others, to which the parent tag had been previously applied, can be excluded from the derived tag (hence the non-monotonicity of the inheritance). This mechanism is a means for structuring the domain and supports incremental development of context-dependent models through tag reuse.

The framework states that an element is visible in the model if the DNF formula derived by substituting contextual tags with their definitions (and also taking into account contextual tag inheritance) holds. So, given a domain in some state, we evaluate the tag definitions to determine which ones are active and then conclude which model elements are visible *in the current domain state*. Since this framework is model-agnostic and does not take into consideration the syntax/semantics of the *i** modeling framework, we need to create a method to process context-parameterized *i** models and produce, for each model element, the expression that determines its visibility.

### 3.2 Applying the Framework to *i** Models

To apply the above-described contextual framework to *i**, we need to associate contextual constraints to *i** model elements (actors, goals, dependencies, and so on). We use *contextual annotations* to specify that certain *i** model elements are only visible in particular contexts, thus taking domain variability into account. Once these annotations are applied to SD/SR diagrams, an algorithm similar to the one presented in [3] for goal models will process these diagrams and the context hierarchies that accompany them, propagate appropriate contextual tags (see below) and generate for each model element a contextual tag expression defining when they are visible. Due to space constraints, we do not present the algorithm in this paper.

In SD diagrams, actor nodes and dependencies can both be parameterized with contextual annotations. For instance**,** Fig. 1A shows that the Customs Broker agent and its incoming dependency from Distributor are only visible in the context of international orders (we are using a simplified form of contextual annotations compared to [3]).

To avoid dangling dependencies, if there is a dependency Dep from actor A1 to actor A2, and these are parameterized with context annotations $C_D$, $C_{A1}$, and $C_{A2}$ respectively (Fig. 1B), then the dependency visibility is defined by the conjunction of these annotations since for a dependency to appear in the model, its own context and the contexts for its source and destination actors have to be active (they have to be in the model). This illustrates that one of the annotations in Fig. 1A is, in fact, redundant. Overall, we are utilizing the hierarchical nature of *i** (i.e., the decompositions) as well as its navigational rules (through dependencies) to minimize the number and size of contextual annotations that are needed. So, for the dependency in Fig. 1B, the complete visibility constraint can be automatically generated from up to three annotations.

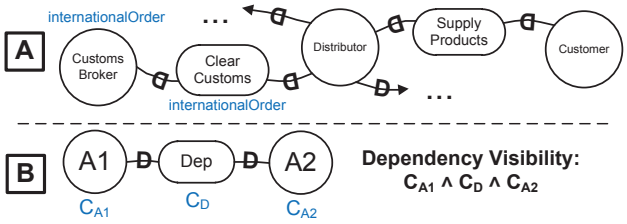



**Fig. 1**. Applying contexts to SD models

In [3], we described how contextual annotations could be applied to goal models, which are, in essence, actor-less, SR models. The main idea was that a contextual annotation applied to a (soft)goal node is automatically propagated to the subtree rooted at that node (i.e., its refinement). This greatly reduces the number of annotations one needs to apply. Moreover, multiple annotations within the same subtree are combined in a way shown in Fig. 2A: when the annotation $C_2$ is applied to a node G1 within the subtree already adorned by $C_1$, both contexts must be active for G1 and its descendants to be visible in the model. For context-parameterized SR models, resource and task nodes are handled similarly. Annotations can also be applied to softgoal contribution links to capture the fact that the evaluation of alternatives can be different in different contexts. E.g., the automatic approval of orders may have a negative contribution to the softgoal Minimize Risk (Fig. 2B) for low-risk customers, but in the case of a high-risk one, the contribution is changed to *break*.

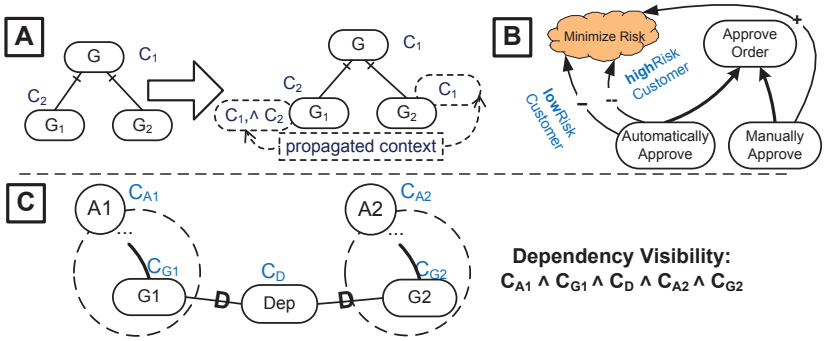



**Fig. 2**. Context propagation (A); contextual annotations applied to contribution links (B); (C) Context-parameterized dependencies in SR diagrams

The key additions to SR diagrams compared to goal models of [3] are actor bubbles and dependencies. Since some actors may be present only in certain contexts (e.g., the Customs Broker in the context of an international order), in SR models they too can be annotated with contexts. Such annotations are implicitly applied to all the model elements within that actor's bubble. The actor's context will be combined with other context annotations within the bubble as shown in Fig. 2A. This way, whenever the context associated with the actor node is not active, the whole bubble disappears.

When looking at a context-parameterized intentional dependency at the SR level, we treat its edges together with the dependum as a single model element parameter-

ized with a context. The contexts for its source node, its target node, and the dependency itself must be active for the dependency to be visible in the model (see Fig. 2C).

### 3.3 Using Context-Parameterized *i** models

With the above approach, we produce a context-parameterized *i** model, in which each model element is associated with a visibility condition (this, in fact, combines into a single model many different model variations). Evaluating these conditions in some domain state produces a model variant with only a subset of the elements. Thus, in effect, contexts act as filters on *i** models by removing irrelevant model fragments.

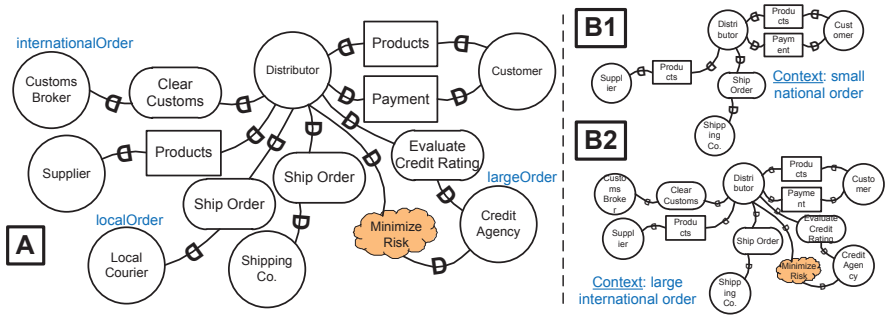



**Fig. 3**. Using contexts as filters on SD models

Given the context definitions that are rooted in real-world phenomena (i.e., *monitorable contexts*), *i** models can be dynamically adapted to changing domain conditions. E.g., the context-parameterized SD model for the distributor scenario in Fig. 3A is seen differently in the context of a small national order (Fig. 3B1) compared to the context of a large international order (Fig. 3B2). Note that these two models are no longer parameterized with contexts – their contextual variability has been *bound*. Thus, they can be analyzed using conventional goal model or *i** analysis techniques.

The same framework can be used to integrate a number of variations of the same model, each capturing a particular model version, a viewpoint, etc. In this case, viewpoints, versions, etc. are represented by contexts that are not monitored, but can be manually turned on or off (e.g., *version1 = true*). These contexts can also be organized into inheritance hierarchies to better facilitate incremental model development and even to mimic approaches like [2]. The idea of context-based visibility can likewise be employed to label alternative sets of leaf-level nodes in SR models (i.e., *strategies* for achieving high-level goals) with contexts, and then turning these contexts "on" or "off", and running analysis algorithms on the resulting model variations.

## 4 Ongoing and Future Work

We are working on the flexible implementation of the context framework to support various flavours of *i**-based and goal modeling notations. This will also help with the validation of the approach. In addition, we are exploring the links between

contexts and business rules. We also plan to identify synergies with the approach of [1], which while having a lot of similarities with our proposal, has a different focus.

We currently view contexts as global and thus shared among actors. However, in some applications, it may be beneficial model contexts on per-actor basis, which implies that in that case, the context-parameterized models would capture the actors' possibly incompatible viewpoints on the system. Also, there may be flavours of *i** modeling, which do not comply with the handling of dependencies in SR models illustrated in Fig. 2C. We are looking into a number of context propagation customizations to accommodate these modeling techniques.

Contexts can be thought of as specifying dynamically loadable model fragments. When the formula defining the context holds, the context becomes active and the model elements that are "in" context become visible in (or loaded into) the model. We plan to explore the idea of context encapsulation as a way to improve scalability in *i**.

## 5 Conclusions

In this paper, we applied to *i** a context mechanism based on the flexible idea of model elements' visibility defined by external factors such as domain characteristics, viewpoints, etc. With the proposed framework, we are able to capture the effects of domain variability on a system using a single context-parameterized *i** model and to automatically produce variations of that model based on the currently active contexts. In essence, contexts here act as filters on *i** models by removing model elements not applicable in the current state of the domain. Contextual variability is thus bound, so conventional goal analysis techniques can be utilized with the generated models. The framework can be used for both monitorable and non-monitorable contexts. Context inheritance is another feature of the approach. It allows for adding structure to domain models, for context reuse, and for incremental *i** model development.

The context framework's aim is not to address the scalability/complexity issues in *i** – it is to help integrate multiple model variations into a single diagram. Here, one has to balance the need for and the benefits of adjusting the model to different domain characteristics and thus the need to have a large number of system variations against the complexity of eliciting and maintaining this large number of system variants.

## References

1. R. Ali, F. Dalpiaz, P. Giorgini. A Goal-based Framework for Contextual Requirements Modeling and Analysis. REJ, 15(4):439-459, 2010.
2. S. M. Easterbrook. Domain Modelling with Hierarchies of Alternative Viewpoints. In Proc. *RE'93*, San Diego, January 1993.
3. A. Lapouchnian and J. Mylopoulos. Modeling Domain Variability in Requirements Engineering with Contexts. In Proc. *ER 2009*, Gramado, Brazil, Nov 9-12, 2009.
4. A. Lapouchnian, Y. Yu, S. Liaskos, J. Mylopoulos. Requirements-Driven Design of Autonomic Application Software. In Proc. *CASCON 2006*, Toronto, Canada, Oct 16–19, 2006.
5. S. Liaskos, A. Lapouchnian, Y. Yu, E. Yu, J. Mylopoulos. On Goal-based Variability Acquisition and Analysis. In Proc. *RE'06*, Minneapolis, USA, Sep 11-15, 2006.

# Security Requirements Engineering for Service-Oriented Applications

Fabiano Dalpiaz, Elda Paja, Paolo Giorgini

University of Trento - DISI, 38123, Povo, Trento, Italy
{fabiano.dalpiaz, paja, paolo.giorgini}@disi.unitn.it

**Abstract.** Security Requirements Engineering (SRE) is concerned with detecting and analysing security issues early in the software development process. Some variants of *i** start since early requirements and rely on modelling actors and their dependencies. Though useful for traditional information systems development, these approaches adopt a bird's eye perspective that is inadequate for service-oriented applications, in which multiple autonomous and heterogeneous agents interact to achieve their own strategic interests.
In this paper we present SecCo (Security via Commitments), a novel SRE framework expressly thought for service-oriented settings. The key intuition is to relate security requirements to interaction. In order to do so, we specify security requirements in terms of social commitments, promises with contractual validity between agents. These commitments describe the security properties the service provider commits to ensure to the consumer while delivering the service.

## 1 Introduction

Software systems are subject to security threats, which may influence organisational assets. Thus, the specification of *security requirements* as well as their implementation by means of *security mechanisms* are of utmost importance. Many security threats are not technical; rather, they are *social*, as they originate from the interactions between social actors (humans and organisations).

Several proposals consider social threats by modelling and analysing security since the early requirements phases. Many frameworks extend *i** [1–3] to model the environment in terms of social actors and their dependencies. Though they represent dependencies between actors, these approaches adopt a centralised view on the modelled setting, which leaves little space to the autonomy and heterogeneity of the actors. Indeed, they implicitly assume that participating actors will behave as described in the goal models.

Service-oriented computing enables cross-organisational business processes and, more generally, black-box interactions among autonomous actors (on the basis on service interfaces). In our previous work [4], we revisited service-orientation in terms of goal-oriented agents that adopt specific roles and interact with others by making and getting social commitments [5]. A commitment is a quaternary relation C(debtor, creditor, antecedent, consequent) in which the debtor promises (*commits*) to the creditor that, if the antecedent is brought about, the consequent will be brought about. As soon as the antecedent holds, the debtor becomes unconditionally committed to bring about the

consequent. Unlike dependencies, commitments are a purely social abstraction. Dependencies imply the intention on the part of the dependee to bring about the dependum [6]. Commitments, on the other hand, imply no intention, they exist as a consequence of the interaction between the debtor and the creditor agent [5]. A service interface is a set of commitments the service provider makes to prospective service consumers.

This paper introduces SecCo [7] (Security via Commitments), a novel SRE framework that combines the early-requirements perspective of SI* [2] with ideas from service-oriented applications [4]. The key idea is to relate security requirements to *interaction*. This means adding constraints to the way actors exchange resources, and to the delegation of responsibilities. These constraints are commitments the actors shall comply with while interacting. For example, a service provider could commit to the privacy of the consumer's personal data, which is required as input to the provided service. Again, the same service provider may commit to the non-delegation of a service to other actors.

## 2 Objectives of the research

The main objective of this research thread is to support security requirements engineering in the context of service-oriented settings. Service orientation is becoming a popular paradigm, especially for cross-organisational settings. The analysis of security aspects is of utmost importance, since information is disclosed (and tasks are executed) beyond the "safe" boundaries of a single organisation. Our goal is to devise a modelling framework as well as algorithms that enable the automated discovery of security issues in the models. We are mainly concerned with composite services, which comprise multiple services that relate different actors acting both as service consumers and providers.

Additionally, we aim to derive security requirements that can be effectively used to specify the service under development. Our purpose is to express security requirements within service interfaces. This ensures that the security needs expressed by the stakeholders result in actual commitments the provider makes to the consumer to satisfy these constraints (the security needs) while delivering the service. For instance, if consumers are concerned with the disclosure of personal data, the service interface may declare that the data will not be disclosed to other actors. Irrespective of the service implementation, such interface makes the provider committed for non-disclosure.

## 3 Scientific contributions

Figure 1 outlines SecCo and shows how security requirements for the composite service under design are derived from the security needs expressed by the stakeholders. SecCo allows to model interaction among actors from different perspectives (views). Security needs are *expressed in* the business view that describes multiple orthogonal perspectives over the considered setting. SecCo currently includes three views: social, authorisation, and resource. We provide more details about these views in Section 3.1. Together, these views provide a comprehensive picture of the setting which includes both business concerns and security aspects.

Security needs are *supported by* the commitments view, which consists of a set of commitments between actors. Such view is a high-level specification of the security

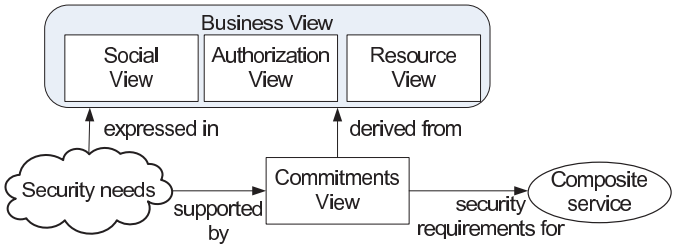



**Fig. 1.** Outline of SecCo: from security needs to security requirements

requirements for the system-to-be. As long as the actors do not violate those commitments, the security needs expressed by the interacting parties are ensured. The link between goals and commitments has already been discussed in [4], in this work we will concentrate on how commitments can guarantee the security needs. The commitments view can be automatically derived from the business view.

### 3.1 Multi-view modelling

SecCo relies on multiple views of the same model, each representing a specific perspective on the analysed setting. Multi-view modelling promotes modularity and separation of concerns. Currently, SecCo includes three views:

- *Social view*: a variant of traditional *i**-based frameworks, more specifically of SI*. As such, it models actors and their dependencies. SecCo supports two types of actors: *agent* and *role*. An agent can play multiple roles. Each actor is characterised in terms of *goals* he wants to achieve. Goals can be *AND/OR-decomposed*. Differently from SI*, in which resources are a means to achieve a goal, here goals are linked to *tangible resources*—information represented on some support means—in various ways: a goal can *read*, *produce*, *modify*, or *distribute* a resource. Resource *possession* indicates that an actor can dispose of certain resources without interacting with other actors. There are two social relationships: *resource provision*—representing the exchange of tangible resources—and *goal delegation*. The distinguishing feature of our language is that in the social view one can express security needs by means of annotations associated to elements of the model (e.g. delegation). We discuss the supported security needs in Section 3.2.
- *Resource view*: focuses on the way resources are structured. We distinguish between tangible resources (introduced in the social view) and *intangible resources*, which denote information irrespective of its representation. For instance, Jim's birthday is an intangible resource. Resources can be hierarchically structured via the *part-of* relation, which relates homogeneous resources (intangible to intangible, tangible to tangible). Intangible resources are *made tangible by* tangible resources. For example, Jim's birthday is made tangible by his identity card.
- *Authorisation view*: represents the authorisations actors grant one to another. We distinguish between delegation of *authority* and delegation of the *authority to delegate*. The second type implies that the delegatee can further delegate the received

authorisation. Authorisations are granted by an actor to another for one or more intangible resources, and specific *operations* are authorised: read, produce, modify, and distribute. An authorisation can be limited to a *scope*—a set of goals—that determines the purposes why the delegatee can use tangible resources that represent those intangible resources.

### 3.2 Expressing security needs

We outline the security needs supported by the SecCo business view with the aid of a small scenario concerning stay permits for international students.

**Scenario.** An international *student* enrolled at the University of Trento needs a stay permit. To obtain it, he needs an official document to prove his enrolment and that his incomes are enough to afford the stay. He asks the *programme coordinator* to issue the document. For this reason, he has to provide his personal data, as well as financial information. His personal data is stored in the university information system. The *programme coordinator* delegates his task to the *secretary*. She retrieves personal data from the information system and drafts the document, then gives the draft to the programme coordinator who has to sign it. The *IS manager* manages authorisations in the university information system in accordance with confidentiality restrictions.

SecCo currently supports the following security needs:

- *Non-repudiation*: in a goal delegation, the delegator wants to prevent the delegatee from challenging the validity of the delegation (repudiating the delegation). For instance, **(Ex1)** the programme coordinator wants to ensure non-repudiation for the delegation of goal "Write document" to the secretary.
- *Redundancy*: in a delegation, the delegator wants the delegatee to adopt redundant strategies for the achievement of the goal. He can either use different internal capabilities, or can rely on multiple actors. For example, **(Ex2)** the secretary wants the IS manager to adopt redundant strategies to obtain the student's income statement, since provided data is often inconsistent.
- *No-delegation*: the delegator requires the delegatee not to further delegate goal fulfilment. This security need is closely related to *trust*: the delegator trusts *that* specific delegatee for some goal, and does not trust other actors. For example, **(Ex3)** the secretary wants the IS Manager not to delegate goal "Get student personal data", as she is afraid someone else would violate data confidentiality.
- *Non-disclosure*: authority over a resource is granted without transferring authority to delegate. For example, **(Ex4)** the IS Manager authorises the secretary for resources personal data and financial status, but requires non-disclosure of such data.
- *Need to know*: when the granted authority to delegate is limited to a goal scope. The actor granting the authority enables the second actor to delegate permission to others as long as other actors conduct operations on the resources within the specified scope. For example, **(Ex5)** the student authorises the IS Manager on a need-to-know basis: personal data and financial status should be produced or distributed in the scope of goal "Write document for immigration office".
- *Integrity*: an actor does *not* delegate the authority to modify a resource. For example, **(Ex6)** the IS Manager authorises the secretary for personal data and financial status as long as these resources are not modified.

### 3.3 Deriving security requirements as commitments

SecCo represents security requirements as commitments between actors. In particular, these commitments are between roles, implying that the actual agents playing those roles are expected to make and comply with those commitments. These commitments are created as a consequence of the security needs expressed by the roles while interacting, namely for each security need expressed by a role on an interaction with another, a commitment in the opposite direction will be created. If all agents playing those roles comply with their commitments [4], the security needs will be guaranteed.

Security requirements are automatically derived from the business view. We sketch some security requirements derived from the scenario in Section 3.2 related to the examples of security needs Ex1-Ex6. In the commitments below, debtor and creditor are roles, whereas antecedent and consequent are propositions.

**Ex1.** The non-repudiation security need results in a commitment from the secretary (sec) to the program coordinator (pc) that, if goal "write new document" is delegated to her, she will not repudiate the delegation:
C(sec, pc, $d_1$ =delegate(pc,sec,writeDocument), non-repudiation($d_1$))

**Ex2.** The redundancy security need implies a commitment from the IS manager ism to the secretary that, if goal "obtain income statement" is delegated from sec to ism, goal redundancy will be guaranteed:
C(ism, sec, delegate(sec,ism,obtainIncomeStmt), redundancy(obtainIncomeStmt))

**Ex3.** The no-delegation security need for getting student personal data results in a commitment from ism to sec that, if the goal is delegated to sec, she will not further delegate the goal.
C(ism, sec, delegate(sec,ism,getStudentData), no-delegation(getStudentData))

**Ex4.** Since our modelling language does not make assumptions on the timing of authorizations, the non-disclosure security need of ism results in an unconditional commitment by sec for the non-disclosure of personal data and financial status. This means that, at any moment the secretary is in possession of those resources, she commits to not disclose them.
C(sec, ism, $\top$, non-disclosure(personalData $\wedge$ financialStatus))

**Ex5.** The need-to-know security need requested by the student (stud) for his personal data and financial status results in an unconditional commitment by ism that those resources will be used only in the scope of writing a document for the immigration office. Granted permissions are to produce and distribute tangible resources that represent those intangible resources.
C(ism, stud, $\top$, ntk(personalData $\wedge$ financialStatus, writeDocumentForIO, p $\wedge$ d)

**Ex6.** The integrity security need expressed by ism results in an unconditional commitment from sec to ism that personal data and financial status will not be modified.
C(sec, ism, $\top$, integrity(personalData $\wedge$ financialStatus))

## 4 Ongoing and future work

We are currently working on SecCo, which will be the socio-technical security modelling language for the EU-funded Aniketos project. Aniketos is about ensuring trustworthiness and security in composite services. Some topics we will investigate are:

- *Obligations view*: laws and organisational rules impose constraints on the way data is exchanged, stored, and used. This implies security needs that are not expressed by stakeholders, but that the service under design shall preserve.
- *Security needs*: we intend to significantly increase the number of security needs our language supports. Some examples are least privilege, anonymity, auditability, and written consent.
- *Formalization and reasoning*: we need to formally represent the models in the business view so to support automated reasoning to verify consistency (e.g. unauthorised resource provisions or delegations) as well as to derive security requirements for the composite service (the commitments that have to be preserved).
- *Deriving service interface specifications languages*: transforming the security requirements expressed as commitments into existing service interface specification languages. Our choice will depend on both language expressiveness and the existence of monitoring frameworks able to check service compliance with its interface.
- *Methodology and tool support*: we will devise a companion methodology in order to make the language applicable, and will build a full-fledged development tool based on the Eclipse Rich Client Platform.

## Acknowledgements

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grants no 257930 (Aniketos) and no 256980 (NESSOS).

## References

1. Liu, L., Yu, E., Mylopoulos, J.: Security and Privacy Requirements Analysis within a Social Setting. In: Proceedings of the 11th IEEE International Conference on Requirements Engineering (RE 2003), IEEE Computer Society (2003) 151–161
2. Giorgini, P., Massacci, F., Mylopoulos, J.: Requirement Engineering meets Security: A Case Study on Modelling Secure Electronic Transactions by VISA and Mastercard. In: Proceedings of the 22nd International Conference on Conceptual Modeling (ER 2003). Volume 2813 of LNCS., Springer (2003) 263–276
3. Mouratidis, H., Giorgini, P.: Secure Tropos: A Security-Oriented Extension of the Tropos methodology. International Journal of Software Engineering and Knowledge Engineering **17**(2) (2007) 285–309
4. Chopra, A.K., Dalpiaz, F., Giorgini, P., Mylopoulos, J.: Modeling and Reasoning about Service-Oriented Applications via Goals and Commitments. In: Proceedings of 22nd International Conference on Advanced Information Systems Engineering (CAiSE'10). Volume 6051 of LNCS., Springer (2010) 113–128
5. Singh, M.P.: An Ontology for Commitments in Multiagent Systems: Toward a Unification of Normative Concepts. Artificial Intelligence and Law **7**(1) (1999) 97–113
6. Yu, E.S.K.: Modelling Strategic Relationships for Process Reengineering. PhD thesis, University of Toronto, Toronto, Ont., Canada, Canada (1996)
7. Dalpiaz, F., Paja, E., Giorgini, P.: Security Requirements Engineering via Commitments. In: Proceedings of the 1st Workshop on Socio-Technical Aspects in Security and Trust (STAST 2011). To appear.

# Goals and Scenarios for Requirements Engineering of Software Product Lines

Gabriela Guedes[1], Carla Silva[2], Jaelson Castro[1]

[1] Centro de Informática – Universidade Federal de Pernambuco (UFPE)
CEP 50740-540, Recife/ PE – Brasil
{ggs, jbc}@cin.ufpe.br
[2] Departamento de Ciências Exatas, Centro de Ciências Aplicadas e Educação
Universidade Federal da Paraíba (UFPB) – CEP 58297-000, Rio Tinto/ PB – Brasil
carla@dce.ufpb.br

**Abstract.** Goal-oriented requirements engineering (GORE) approaches offer a natural way to capture similarities and the variability in software product lines (SPLs) development. Besides, they can effectively capture both the stakeholders' objectives and the system requirements. From i* models, for example, it is possible to systematically obtain feature models. To complement the requirements specification of SPLs, their behavioral characteristics can be captured by using a scenario specification technique. This paper presents a process. An extension of i* that includes cardinality is used in connection with feature models and a use case scenarios to support the requirements engineering phase in SPLs development. This process also includes activities to aid the configuration of requirements artifacts for a specific product in the SPL. The paper also presents the case study being used to illustrate the proposed process.

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

## 1 Introduction

Requirements Engineering (RE) is the phase of software development concerned with producing a set of software systems specifications that satisfy the *stakeholders* needs and can be implemented, deployed and maintained [1].

In RE for Software Product Lines (SPL), feature models are used to capture similarities and the variability of product families. However, according to Borba and Silva [2] it is a great challenge to establish a relationship between features of a software product and the objectives of the stakeholders. In this context, we proposed a Goal-Oriented Requirements Engineering (GORE) approach that provides a systematic way to discover the features that will be part of a SPL and also allows the systematic selection of the features for a particular product [3].

It is worth to complement the requirements specification obtained with this GORE approach. The dynamic aspect of a SPL may be described by a scenario specification technique. Scenarios describe the behavior of the system functionality and are widely

used in requirements engineering because they are easily understood by stakeholders [4]. In this paper, we present a process that integrates a GORE approach for SPL, feature modeling and a scenario specification technique.

## 2 Objectives of the Research

Many goal-oriented approaches were proposed to model requirements variability in SPL [5, 6, 7, 8]. A comparison of these approaches was presented in [2] and motivated the definition of the G2SPL (Goals to Software Product Lines) approach [3]. It relies on the i*-c (i* with cardinality) language, which is used to (i) structure requirements according to the stakeholders intentions for the SPL, (ii) facilitate the gathering of the features that define the SPL and (iii) aid the configuration of an individual product.

In SPL, specifying non-trivial features can cause the scattering of the SPL variation points on the line's artifacts. Moreover, some feature specifications combine, in their artifacts, information from the SPL variants and the product configuration. The scattering and tangling of features related concerns can also be observed in the scenario specifications of the SPL. These concerns are, therefore, crosscutting and may compromise the maintainability and understanding of the SPL artifacts [9].

Crosscutting concerns are requirements which may impact multiple modules or components. Thus, the crosscutting concerns (representing functional or non-functional requirements) are properties that affect various parts of the system. The importance of their proper handling is evident. We must take into account the way in which the crosscutting concerns interact with other concerns, otherwise there is the risk that the nature of these interactions only becomes clear in later stages of software development. This can cause a higher cost in solving problems related to the system evolution and maintenance [10].

One of the studies concerned with the separation of crosscutting concerns in scenario specifications is the technique MSVCM (Modeling Scenario Variability as Crosscutting Mechanisms) [9]. This technique improves the separation of concerns between the variability management and the scenario specifications of the SPL. It deals with scenario variability as a composition of different artifacts such as use case specifications, feature models, product configuration and configuration knowledge.

Another study that is concerned with the separation of crosscutting concerns in scenario specifications is MATA (Modeling Aspects using a Transformation Approach) [10]. MATA is an aspect-oriented modeling approach that uses graph transformations for specifying and composing aspects. Scenario specification in MATA is performed as follows: a non aspectual base scenario may be specified by a sequence diagram, while an aspectual scenario is described by a sequence diagram enhanced with roles. These roles work as variables that must be instantiated when the aspectual scenario and the base scenario are composed. Recently, MATA was integrated with a GORE approach to obtain a systematic identification of crosscutting concerns in the use case scenario specification [11].

This paper proposes the definition of a RE process for SPL that integrates a GORE technique and a scenario specification technique with separation of crosscutting

concerns. In particular, we are extending the G2SPL approach to include activities related to the generation and configuration of scenarios specifications for SPL.

## 3 Scientific Contributions

The extended G2SPL process, shown in Fig. 1, was modeled using the BPMN (*Business Process Modeling Notation*) [12]. It consists of eight activities, explained as follows:

1. Creation of the SR (Strategic Rational) Model: this activity consists of modeling the stakeholders' goals using i* framework. The output of this activity is a SR Model.
2. Identification of the Candidate Elements to be Features: in this activity, the Domain Engineer identifies the elements of the SR Model that could represent features. According to Silva et al. [3], features can be extracted from Tasks and Resources.
3. Reengineering the SR Model: in this activity, cardinality is added to the SR model. Restructuring is based on some heuristics tailored for i*-c language [3]. The output is a SR Model with cardinality.
4. Elaboration of the Feature Model: this activity is concerned with the derivation of the Feature Model of a SPL. The input artifacts are some heuristics and the SR Model with cardinality and the output is the Feature Model.
5. Reorganization of the Feature Model: this activity is considered optional. If the feature model has repeated features, sub-features with more than one father or different features with the same meaning, reorganization is required. This activity can be performed as many times as the domain engineer believes it is necessary [3].
6. Elaboration of the Use Case Scenarios: the SPL use case scenarios are specified according to an adaptation of the guidelines defined by Castro et al. [13]. This activity uses the SR Model with cardinality as input and the output is the use case scenarios of the SPL.
7. Generation of Use Case Scenarios with Separation of Crosscutting Concerns: in this activity, both the use case scenario specification and the feature model are used to generate use case scenarios with separation of crosscutting concerns. In order to accomplish this, we should choose between the MSVCM (Modeling Scenario Variability as Crosscutting Mechanisms) [9] and the MATA (Modeling Aspects Using a Transformation Approach) techniques [10].
8. Configuration of the Product Artifacts: the purpose of this activity is the derivation of the artifacts for a specific product of the SPL. The outcomes of this activity are the use case scenario description, the configuration model (containing the chosen features) and the SR model of a particular product.

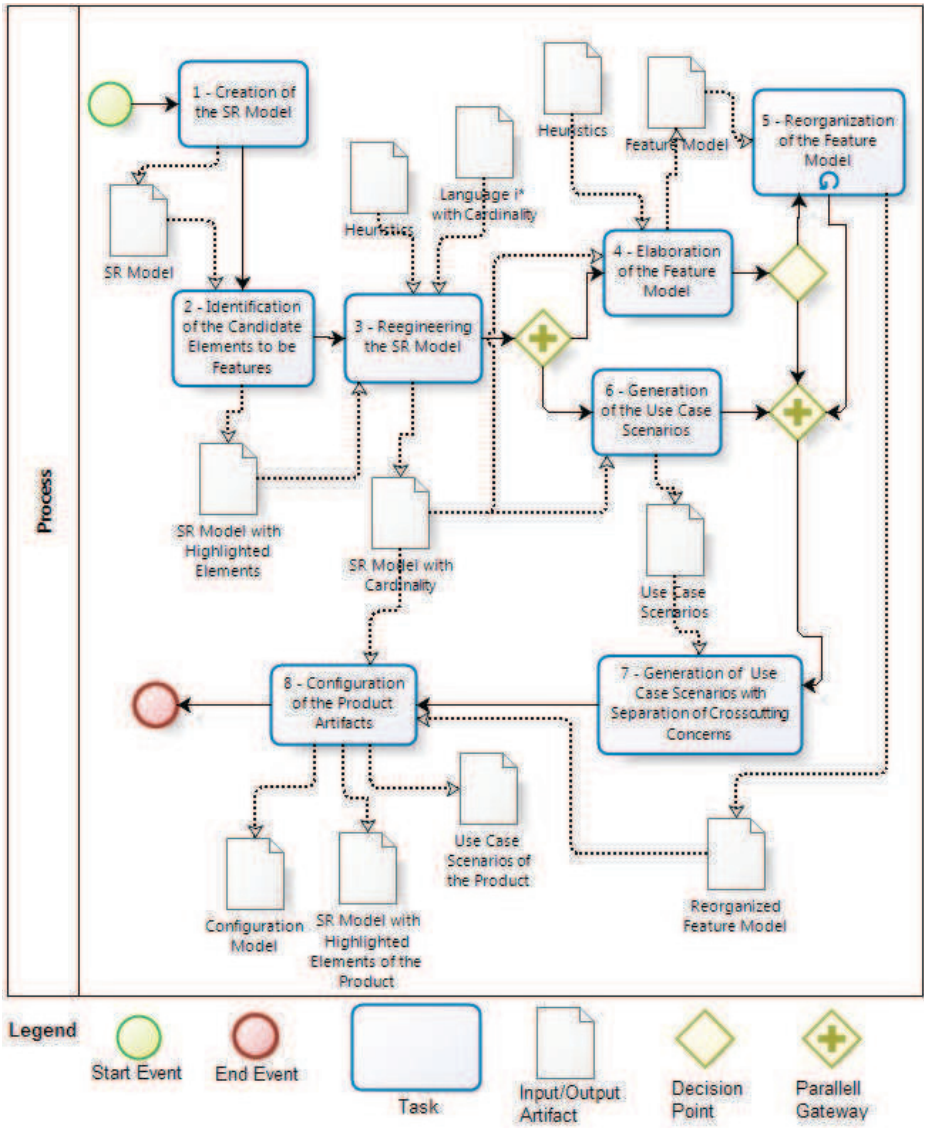



**Fig. 1** Extended G2SPL process model

### 3.1 Case Study

We chose the Motorola TaRGeT (Test and Requirement Generation Tool) project [14] as our case study. TaRGeT is a SPL whose products are tools that automatically generate tests suites from scenario specifications written in a given template. In this

case, the productivity is increased, since it is only necessary to generate Tests Suites from the Scenarios Description.

The SR model of TaRGeT SPL is shown in Fig. 2. Note that we are using the i*-c notation to represent some optional elements. The optional elements are "Detect Scenario Changes and Update Test Cases" and "Verify Scenarios Syntactically" tasks, since they have cardinality [0..1]. The tasks involved in means-end relationships are optional too.

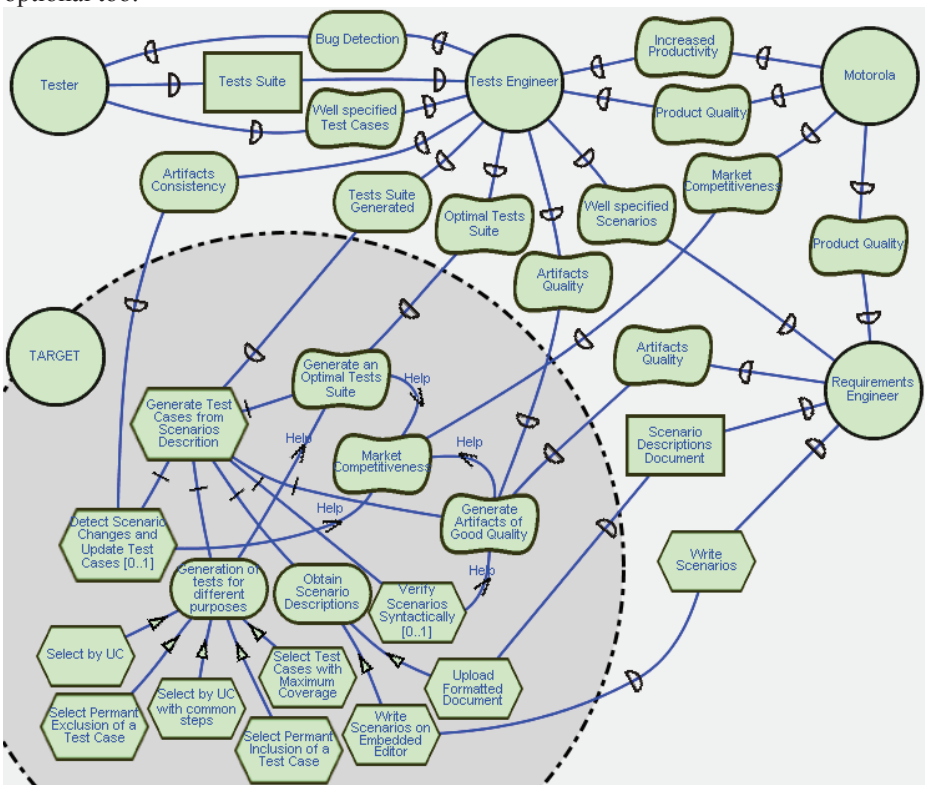



**Fig. 2** TaRGeT SR model

## 4 Conclusions

We presented a process to support RE of SPL in regard of the elaboration of requirements artifacts. The proposed process aims to (i) provide the development of more complete requirements artifacts, (ii) enable the systematic construction of model features, (iii) allow the systematic generation of artifacts (goal models, feature models and scenarios specification) for a specific product, and (iv) support the systematic configuration of the artifacts of a product.

Regarding to the case study, we have performed the first three activities of the process and, as a result, we have produced the TaRGeT SR Model (see Fig.2).

## 5 Ongoing and Future Work

So far, we have held meetings with members of TaRGeT project for requirements elicitation and validation purposes. Currently, we are carrying out the remaining activities of the process. As future work, we suggest the development of a tool to support the whole process, since only two activities have tool support ("Creation of the SR Model" and "Elaboration of the Use Case Scenarios") [15, 16].

## References


1. Lamsweerde, A.: Goal-Oriented Requirements Engineering: A Guided Tour. In: RE '01: Proceedings of the Fifth IEEE International Requirements Engineering Conference. Washington, DC, USA. IEEE Computer Society, p.249-263, 2001.
2. Borba, C., Silva, C.: A comparison of goal-oriented approaches to model software product lines variability. In: LNCS, Vol. 5833, pp. 244-253, Springer-Verlag, 2009.
3. Silva, C., Borba, C., Castro, J.: A Goal Oriented Approach to Identify and Configure Feature Models for Software Product Lines. In: Proc. of the WER'11, Rio de Janeiro, Brazil (2011)
4. Maiden, N, Alexander, I.: Scenarios, Stories, Use Cases: Through the Systems Development Life-Cycle. 1 ed., Wiley (2004)
5. Yu, Y., Leite, J.C.S.P., Lapouchnian, A., Mylopoulos, J.: Configuring features with stakeholder goals. In: Proc. of the ACM SAC'08, Fortaleza, Brazil, pp. 645-649 (2008)
6. Mussbacher, G., Amyot, D., Araújo, J., Moreira, A. Modeling Software Product Lines With AoURN. In: Ws on Early Aspects at AOSD'08, Brussels, Belgium. ACM (2008)
7. Silva, C., Alencar, F., Araújo, J., Moreira, A., Castro, J.: Tailoring an Aspectual Goal-Oriented Approach to Model Features. In: Proc. of the 20th Intl. Conf. on Software Engineering and Knowledge Engineering (SEKE'08), San Francisco Bay, USA (2008)
8. Silva, L., Batista, T., Soares, S., Santos, L.: On the Role of Features and Goals Models in the Development of a Software Product Line. In: Ws on Early Aspects at 9th Annual Aspect-Oriented Software Development Conference (AOSD'10), Rennes, France (2010)
9. Bonifácio, R. and Borba, P.: Modeling Scenario Variability as Crosscutting Mechanisms. In: AOSD'09, Charlottesville, Virginia, USA (2009)
10. Whittle, J., Araujo, J.: Scenario modelling with aspects. Software, IEE Proceedings. 151(4): p. 157-171 (2004)
11. Oliveira, C., Araújo, J., Silva, C. Integração de KAOS com Cenários Aspectuais (In English: Integration of KAOS with Aspectual Scenarios). In: XV Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2010), Valencia, Spain (2010)
12. Business Process Modeling Notation, V1.1. OMG Available Specification, 2008. Available at: http://www.omg.org/spec/BPMN/1.1/PDF/. Last access: June 2011.
13. Castro, J., Alencar, F., Santander, V., Silva, C.: Integration of i* and Object-Oriented Models. In: Yu, E., Giorgini, P., Maiden, N., Mylopoulos, J. (eds). Social Modeling for Requirements Engineering. 1st Ed., MIT Press, 2011. Chapter 13, pp. 457-483
14. Test Research Project (Motorola Brazil Test Center), CIn-UFPE/Brazil and UFCG/Brazil, http://twiki.cin.ufpe.br/twiki/pub/LabPS/ModulosApredizagem/TreinamentoTaRGeT.pdf
15. IStar Tool - IStar modeling Tool support. Available at: http://portal.cin.ufpe.br/ler/Projects/IStarTool.aspx. Last access: June 2011.
16. Vicente, A., Santander, V., Castro, J., Freitas, I., Matus, F.: JGOOSE: A Requirements Engineering Tool to Integrate I* Organizational Modeling with Use Cases In UML. Ingeniare. Revista Chilena de Ingeniera 17(1) (2009) 6-20. Available at: http://andvicente.googlepages.com/aav(undergraduate). Last access: June 2011.

# Bridging User-Centered Design and Requirements Engineering with GRL and Persona Cases

Shamal Faily

Department of Computer Science, University of Oxford
Wolfson Building, Parks Road, Oxford OX1 3QD UK
shamal.faily@cs.ox.ac.uk

**Abstract.** Despite the large body of i* research, there has been comparatively little work on how goal-modelling techniques can help identify usability concerns. Recent work has considered how goal models might better integrate with User-Centered Design. This paper takes an alternative perspective by examining how work in User-Centered Design, specifically Persona Cases, can be re-framed as goal models. We briefly describe an approach for doing this, and present some preliminary results from applying this approach using the Goal-oriented Requirements Language and existing tool support.

## 1 Introduction

i* and related agent-oriented requirements engineering techniques are useful for modelling complex relationships between social agents and intentional concepts. Surprisingly, however, there appears to be comparatively little work on the usefulness of these techniques for eliciting concerns affecting the usability of systems for its participating users.

Previous work has considered how i* might be integrated with User-Centered Design techniques to facilitate communication between requirements engineers, stakeholders, and designers [1]. This work argues that such an approach adds a creative element to the engineering perspective associated with goal-oriented techniques. Follow on work by Leonardi et al. [2] proposes the use of visual scenarios to contextualise Tropos models, and using personas [3] to fulfil the role of actors. Personas are behavioural specifications of archetypical users which embody their needs and goals; since their initial introduction by Cooper [3], personas have become a mainstay in User-Centered Design. Leonardi et al. identify several issues associated with translating formal models to more engaging artifacts like scenarios, but attention also needs to be paid to the validity of the personas used. If personas are not carefully developed then criticisms about their validity may also threaten the validity of any artifacts they influence [4].

*Persona Cases* have recently been proposed as a technique for providing independent validation of personas [5]. Persona cases are personas whose characteristics are both grounded in, and traceable to, their originating source of

empirical data. As a validation tool, persona cases are built on the premise that sense-making in qualitative data analysis is an argumentative activity, and the elements of Grounded Theory [6] can be re-framed as an argument using Toulmin's model of argumentation [7]. Expressing persona data using i* can also provide a means of validity by eliciting intentional relationships that support or challenge aspects of a persona's behaviour. However, aside from providing a means for persona validation, there are three additional reasons why integrating this work with i* and related goal-oriented approaches might be useful from a design perspective.

First, given the analogies that can be drawn between i* and other approaches for design rationale, and Toulmin's model being the basis upon which these approaches are built, it seems reasonable to expect alignment between i* concepts and persona cases. Alignment between concepts may allow qualitative models to be re-framed as goal models in the same sense that they can currently be re-framed as persona skeletons.

Second, current efforts to support interchange between different i* modelling tools also facilitate the generation of goal models by requirements management tools that support aligning concepts.

Third, because goal models provide an alternative way of contextualising personas, an integrated approach benefits UX (User Experience) designers as well as requirements analysts. When augmented with tasks that stakeholders might carry out, designers may be more interested in using these models to understand user activities in context [8] rather than as a vehicle for directly eliciting requirements. Consequently, framing goal modelling as a UX design technique may lead to an expanded audience for i* and related techniques, who may identify hitherto unseen affordances in both goal models and goal modelling techniques.

In this short paper, we describe preliminary work bridging User-Centered Design and Requirements Engineering by re-framing persona cases as goal models. In section 2, we describe our research objectives and the research approach adopted before describing our contributions to date in section 3. We summarise these contributions in section 4, and describe on-going work in this area.

## 2 Objectives

The objectives of this research are two-fold.

First, we want to better understand how personas and associated concepts align with i*. Unlike previous work, we wish to align concepts *to* i* rather than the other way around; this is because we are considering i* as a tool to support User-Centered Design, rather than vice-versa. Because personas are often described using scenarios, we expanded the scope of analysis to include use cases carried out by personas. Because previous work suggests that goal models align with use cases [9], it is possible that the alignment relationship between goal models and use cases is bi-directional.

Second, we want to understand how existing tool support can exploit these relationships such that goal models can be automatically generated based on pre-existing analysis. As a baseline for this research, we use the Goal-oriented Requirements Language (GRL) as the goal-modelling language for aligning concepts, and the jUCMNav [10] Eclipse plugin because of its support for importing XML based GRL files. The CAIRIS tool [11] was used for managing persona case and use case elements, and generating GRL files.

## 3 Contributions

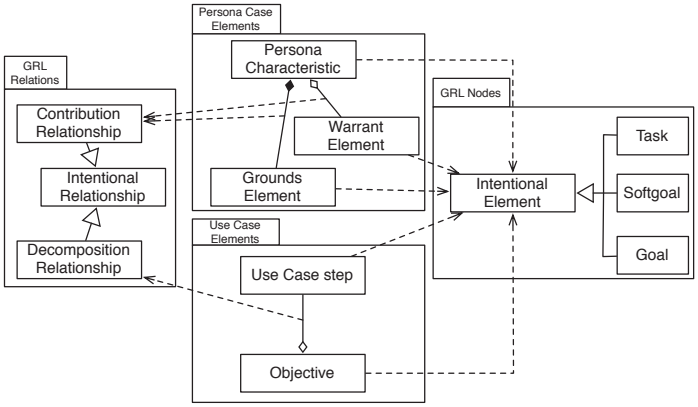



**Fig. 1.** Conceptual relationships between persona case and use case elements with GRL

To date, we have made contributions to each of our research objectives.

We identified several aligning relationships between persona case and use case elements and GRL; these are summarised in figure 1. The elements in this model align with concepts from the IRIS (Integrating Requirements and Information Security) meta-model: a conceptual model aligning elements from Requirements Engineering, HCI, and Information Security [12].

Because the [long] names associated with persona characteristics and their supporting elements are displayed on a goal model, a short synopsis was associated with each; a similar synopsis was assigned to each associated use case objective and step. As figure 1 shows, each stipulated persona case and use case element was associated with a GRL intentional element. To date, our preliminary research has considered only the sub-set of tasks, goals, and soft-goals as candidate aligning elements.

To support the bounding of elements associated with i* Strategic Rationale models, persona characteristic synopses and supporting elements were automatically bounded by persona name. Synopses associated with use case steps were

associated with either personas or concepts associated with the system being designed.

Associations between persona characteristics and their supporting argumentation elements were aligned to GRL contribution relationships. Our initial work indicates that GRL contribution relationships are semantically richer than the relationships between persona characteristics and their supporting grounds and warrants. For this reason, it was necessary to associate the navigability between elements in the contribution relationship with each characteristic-grounds and characteristic-warrant relationship, together with the strength of the contribution. This strength is based on the qualitative contribution values associated with GRL; these range from *Make* and *SomePositive* to *SomeNegative* and *Break*. At present, the analyst is responsible for deciding both the navigability direction and the strength of the qualitative contribution. This decision is based on the use case's impact on the persona, and activities related to both the persona and the use case. Associated with each use case step is a decomposition relationship between the intentional element associated with the use case objective and the element associated with the step.

Figure 2 shows a partial goal model reflecting one characteristic of a persona (Helen) that personifies a mother of a young child. The figure shows how the characteristic *Maintain work-life split* is modelled as a soft-goal, and the grounds contributing to it are modelled as goals which help or make this soft-goal. The figure also illustrates how this characteristic is a means for the soft-goal *Mother young child*; this soft-goal is a warrant for the characteristic. The figure also shows how the step *Set device to sharing* associated with the *Content sharing and storage* use case hurts one of the goals contributing to Helen's ability to maintain a work-life split.

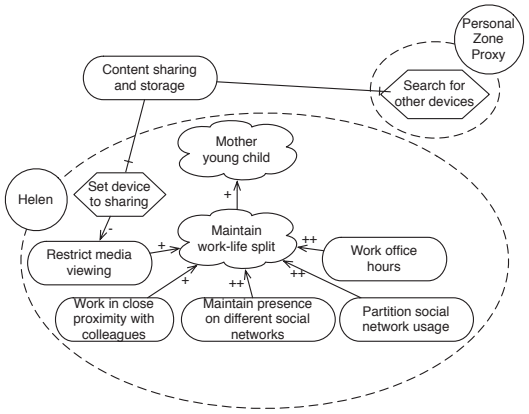



**Fig. 2.** Partial i* model of a persona characteristic and related use case steps

CAIRIS was updated to support the association of synopses with persona case and use case elements, and the association of the requisite contribution link attributes – contribution direction and strength – to each characteristic-grounds and characteristic-warrant association. The tool was also updated to allow a GRL model file to be generated for a selected persona and associated use case. When models are imported into jUCMNav, additional contribution links were added to elements associated with use case steps to indicate whether these help or hinder persona goals or tasks. Such links are added at this late stage as these may not be obvious until the initial goal model is displayed. Figure 3 illustrates a complete GRL model generated by CAIRIS and imported into jUCMNav for all of Helen's characteristics and all steps of the *Content sharing and storage* use case.

## 4 Conclusion and Ongoing Work

This paper presented an approach for generating goal models from persona cases and their associated elements. We described how persona case and use case elements align with complementary elements of GRL, and demonstrated how existing tool support can take advantage of this alignment to generate GRL models from pre-existing model data in CAIRIS.

As part of the EU FP7 *webinos* project, we are currently using this approach to model the impact of personas carrying out use cases which help or hinder their personal or occupational goals. Insights gleaned from this and other goal models are currently being used to develop scenarios illustrating the unintentional impact of webinos to prospective users and their security and privacy expectations.

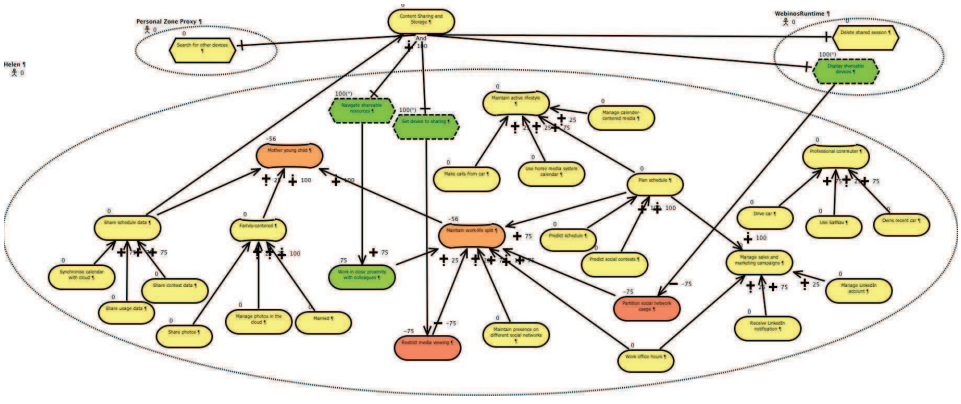


**Fig. 3.** Generated GRL model displayed in jCUMNav based on a persona case and an associated use case modelled in CAIRIS

## 5 Acknowledgements

The research described in this paper was funded by the EU FP7 *webinos* project (FP7-ICT-2009-05 Objective 1.2). We are also grateful to Daniel Amyot and Sepideh Ghanavati for their advice on using jUCMNav and GRL.

## References

1. Francescomarino, C.D., Leonardi, C., Marchetto, A., Nguyen, C.D., Qureshi, N.A., Sabatucci, L., Perini, A., Susi, A., Tonella, P., Zancanaro, M.: A bit of "persona", a bit of "goal", a bit of "process" ... a receipe for analyzing user intensive software systems. In Castro, J., Franch, X., Mylopoulos, J., Yu, E., eds.: Proceedings of the 4th International i* Workshop. Volume 586. CEUR Workshop Proceedings (2010) 36–40
2. Leonardi, C., Sabatucci, L., Susi, A., Zancanaro, M.: Ahab's leg: exploring the issues of communicating semi-formal requirements to the final users. In: Proceedings of the 22nd international conference on Advanced information systems engineering. CAiSE'10, Berlin, Heidelberg, Springer-Verlag (2010) 455–469
3. Cooper, A.: The Inmates Are Running the Asylum: Why High Tech Products Drive Us Crazy and How to Restore the Sanity (2nd Edition). Pearson Higher Education (1999)
4. Chapman, C.N., Milham, R.P.: The persona's new clothes: Methodological and practical arguments against a popular method. Proceedings of the Human Factors and Ergonomics Society 50th Annual Meeting. Available at http://cnchapman.files.wordpress.com/2007/03/chapman-milham-personas-hfes2006-0139-0330.pdf (2006) 634–636
5. Faily, S., Fléchais, I.: Persona cases: a technique for grounding personas. In: Proceedings of the 29th international conference on Human factors in computing systems, ACM (2011) 2267–2270
6. Corbin, J.M., Strauss, A.L.: Basics of qualitative research : techniques and procedures for developing grounded theory. 3rd edn. Sage Publications, Inc. (2008)
7. Toulmin, S.: The uses of argument. updated edn. Cambridge University Press (2003)
8. Norman, D.A.: Logic versus usage: the case for activity-centered design. Interactions **13**(6) (2006) 45–ff
9. Castro, J., Alencar, F., Santander, V., Silva, C.: Integration of i* and Object-Oriented Models. In Yu, E., Giorgini, P., Maiden, N., Mylopoulos, J., eds.: Social Modeling for Requirements Engineering. MIT Press (2011) 457–483
10. Mussbacher, G., Ghanavati, S., Amyot, D.: Modeling and Analysis of URN Goals and Scenarios with jUCMNav. In: Proceedings of the 2009 17th IEEE International Requirements Engineering Conference, RE. RE '09, Washington, DC, USA, IEEE Computer Society (2009) 383–384
11. Faily, S., Fléchais, I.: Towards tool-support for Usable Secure Requirements Engineering with CAIRIS. International Journal of Secure Software Engineering **1**(3) (July-September 2010) 56–70
12. Faily, S., Fléchais, I.: A Meta-Model for Usable Secure Requirements Engineering. In: Proceedings of the 6th International Workshop on Software Engineering for Secure Systems, IEEE Computer Society (2010) 126–135

# Issues and Challenges in Coupling Tropos with User-Centred Design

L. Sabatucci, C. Leonardi, A. Susi, and M. Zancanaro

Fondazione Bruno Kessler - IRST CIT
sabatucci,cleonardi,susi,zancana@fbk.eu

**Abstract.** Goal-oriented requirements engineering aims at eliciting, elaborating, structuring, specifying, analyzing and documenting functional and non-functional requirements. This activity must include the involvement of final users of the system across the whole process to reduce the risk of misunderstanding the domain, missing important details and to increase the final value of the product. User-Centred Design is an approach that focuses on the continuous communication between requirements engineers and stakeholders, thus distributing responsibilities of the decision process about the requirements.
In this paper we explore the issues and challenges of coupling User-Centred Design and Goal-Oriented methods as we experienced in a real project aiming at developing smart environment for nursing home to support medical and assistance staff.

## 1 Introduction

When facing the problem of designing technologies, two roads diverge in the wood: Requirements Engineering (RE) and User-Centred Design (UCD). Indeed, both approaches ground their processes in information about the people that are directly or indirectly involved by the technology that has to be developed. Yet, they not only have different set of techniques and incompatible vocabularies but also they are based on two diverging epistemological foundations. UCD practitioners shun from any formal method at risk of compromising the actual use of the knowledge gained in the field. On the other side, RE practitioners often loose contacts with real people because formalizations cannot easily be shared with them: user analysis thus becomes a single-player game rather than a meaningful dialogue with stakeholders.

The need of reconciliation was raised in our experience with a large research project aimed at developing a smart environment in nursing home as support to medical and assistance staff. Since the beginning, we adopted the Tropos methodology and encountered a well-known problem in requirement elicitation: the complexity of the domain makes it difficult not only to understand users' needs but also to discuss with them our understanding of the domain.

In parallel, we adopted a UCD approach, mainly based on contextual interviews: our aim was to elicit true needs by observing daily practices and not only prescribed regulations. Activity scenarios were sketched to test with users our

understanding of their working practices. The narrative form of scenarios and the use of fictional personas made easier the confrontation between the analysts and the stakeholders. Later on, those scenarios were augmented with our vision of the technology. Again, the narrative form made the communication easier for them to understand and discuss in a tangible way. The issue with this approach, again well known in literature, was the lack of formality in collecting and analyzing the results.

The contribution of this paper is the analysis of foundations for the integration of the two approaches without compromising their very nature: in the differences there lies the power of the integration and its risks. The practical experience of mediating these two perspectives represents the basis for a discussion of generic issues and challenges that arise for the integration of User-Centred Design and Goal-Oriented methods.

The paper is structured as follows: Section 2 introduces the practical experience that motivates the research objective; Section 3 is a discussion of challenges arising from generalizing the problem; finally in Section 4 conclusions and future works are reported.

## 2 Objective

Our general research objective is to maximize the connection between the very early stages of the domain analysis and requirements elicitation and the phase of their representation into semi-formal artifacts, such as goal-oriented techniques, to reason about them.

An initial approach to this coupling has been exploited in a real project and described in the next section. On the bases of the experience, our aim here is that of describing how, under what conditions and at what extent this experience can be generalized.

### 2.1 The Methodologies

The **Tropos methodology** [5] relies on a set of concepts, such as actors, goals, plans, resources, and dependencies to formally represent the knowledge about a domain and the system requirements. An actor represents an entity that has strategic goals and intentionality within the system or the organizational setting. Goals represent states of affairs an actor wants to achieve. A Plan is a means to realize a goal. Actors may depend on other actors to attain some goals or resources or for having plans executed.

**User Centered Design**. UCD is a design philosophy that exploits a number of different techniques within an iterative design process. Tenets of UCD are: early focus on users, tasks and environment, the active involvement of users in the design process, allocation of functions between user and system, the incorporation of user-derived feedbacks into system design, iterative design whereby a prototype is designed, tested and modified. UCD exploits a series of well-defined methods and techniques coming from social sciences and psychology for analysis, design, and evaluation technologies.

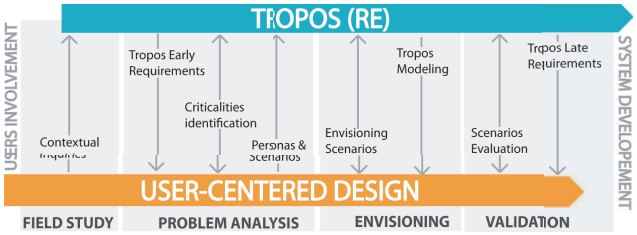



**Fig. 1.** Overview of the requirement elicitation process

### 2.2 The Experience: an Ambient Assisted Living System

The whole experience was articulated in seven phases all of them characterized by the use of both Tropos and UCD (see Figure 1).

*Phase 1 - Field data collection.* The process started with the investigation of the domain in order to understand the organizational setting in four nursing homes and to identify the needs of the stakeholders involved. In order to get rich insights about the context, we used contextual inquiry that demonstrated the capacity to satisfy the needs for a deep but at the same time rapid understanding of complex domain. Contextual inquiry consists in observing and interviewing people in their context, preferably when performing their tasks.

*Phase 2 - Data Interpretation.* The data interpretation is the step in which data coming from the domain is shared across the team and becomes knowledge. In our process, data interpretation is concurrently carried out: i) following contextual design approach dimension (flow model, sequence model, cultural model, artifact model, physical model) and ii) exploiting the early identification phase of Tropos.

*Phase 3 - Problem analysis.* The analysis of critical aspects was developed to highlight main problems that professionals of nursing homes experience in their job. The aim is to highlight every possible breakdown or problem that may occur in the organization that hinders the achievement of goals. A criticality relates an exceptional event to front with Tropos goals and tasks that are identified to receive a negative contribution. It also encapsulates the context in which the problem may occur.

*Phase 4 - Personas and activity scenarios design.* Introducing personas in scenarios-based approach provides an anchor against self- referentiality in design and make stories more concrete. We authored a set of activity scenarios — user stories about problems and criticalities identified through user studies.

*Phase 5 - Envisioning, from data to design.* The phase of envisioning exploited participative workshops in order to move from personas and activity scenarios identified in Phase 4 in order to envision how to introduce the technology in the analyzed domain. As a consequence of the envisioning focus group, and the introduction of the system into the goal-model, the Tropos process moved from the early requirement phase to the late requirement phase.

*Phase 6 - From design ideas to Tropos modeling.* Tropos diagrams and scenarios were jointly used to refine the ideas emerged during the creative workshops. On one hand, technological scenarios were designed to make design ideas concrete and to trigger reflection about possible services. On the other hand, Tropos diagrams were developed to more systematically analyze how the introduction of a system impacts on the domain actors.

*Phase 7 - Evaluation of technological scenarios.* Here visual scenarios were derived from the Tropos models and used for the validation phase (focus groups with stakeholders), where multiple views on the domain are required to drive the negotiation and refinement of requirements with stakeholders and project partners. If envisioning scenarios provided a concrete instance of a particular design solution, that is very helpful to discuss with stakeholders, on the other hand, Tropos diagrams aided designers in reconsidering design solutions and elaborating alternatives thanks to the possibility to trace back design solution to initial abstract requirements. The output of this phase was the agreement on early requirements and the refinement of Tropos late requirement diagrams.

## 3 Contribution: Integration Issues and Challenges

Crossing boundaries between two research approaches requires — borrowing a distinction from social sciences — either an assimilation or an integration process. In the case of assimilation, one approach must be modified to be assimilated into the other approach: while the risk is to loose the strength of the approach itself, the advantage is to work in a situation of methodological purity [6].

In the case of integration, as well as in our experience, practitioners should accept to work in a situation of methodological pluralism: the goal is not to transform a specific approach to make it fit into another one, but rather to bridge the gap between different research traditions and take advantages of their mutual strengths. Usually, the integration is more complex because practitioners work into different methodological traditions that have to be first understood to be integrated [7]. Integration, therefore, requires creating preconditions for a beneficial dialogue.

In the ACube project we recognized the irreducible cultural difference between the two approaches and therefore decided to work in a dialogic perspective, that is grounded on communication and iterative confrontation [4]. The following considerations come from generalizing the practical experience in coupling Tropos and UCD.

### 3.1 Epistemological challenge

The first issue is to consider epistemological foundations and validity criteria of both the approaches to manage the differences without weakening and distorting the two research paradigms. Whereas several methods employed in UCD derive from a constructivist perspective, many requirement engineer approaches are

grounded, instead, on a constructivist perspective — even if the debate on the positivist nature of many RE methods has recently been criticized [3].

*Constructivism* declares there is no single valid methodology and researchers play an active role in defining the reality. User-centered design is grounded on this research tradition: hence the scarce formality of methods, the subjective insights developed by practitioners, and the ambiguities in the analysis are, if correctly managed, not only accepted but actively perused [1].

Concerning the design activity, the *positivist* position suggests a rigid structure for the modeling activity and the reasoning process: while for constructivism, scientific knowledge is built by scientists, for positivism the knowledge is 'discovered' by the use of actual sense experience. The criticism on the positivist nature of goal-models relies on the semi-formality of such languages: there is no one 'right' goal model for describing a domain [3].

The integration challenge must consider these epistemological differences and be grounded on setting the pre-conditions to mediate these philosophical positions and generate a profitable dialogue between the two methods.

### 3.2 Linguistic Challenge

Near the methodological boundary, a linguistic boundary exists. The concurrent usage of both approaches requires that a common language exists in order to make a dialogue possible. For instance, several concepts exist in both Tropos and UCD that suggest an integration is possible and profitable; examples of these are the pairs of goal/need, actor/persona, task/activity; yet these terms have slight different meanings in the two methodologies that hinder the integration process.

The identification of a common language for coupling the methodologies must pass through a reconciliation of terms. Two alternatives are possible: (i) to create a unified meta-model of the integrated process, or (ii) to tie up terms with similar meanings while keeping them separate. The first way is fascinating but it presents some notable risks, such as, for instance, the loosing of the flexibility and expressivity of tools like personas and scenarios that often requires to be unbounded within precise frames. The second way is preferable but it requires an additional effort for creating a framework in which data of different nature can easily collaborate.

### 3.3 Lesson Learned

By recapping the design experience described in Section 2.2, we can recognize three main mechanisms that shaped the relationships and set the dialogue for an efficient cooperation between the two teams:

– Strengths/limits analysis: it relies on the identification of strengths and limits of both the methods while achieving a given design objective. This allows to define integration points between UCD and RE methods and take full advantages of their reciprocal strengths.

– Making the divides explicit: it consists in identifying barriers that may hinder the dialogue between the two methods. The anticipatory exploration of the barriers that can prevent a synergy between the two approaches was pursued.
– Mutual learning: social-based techniques may enable continuous information exchange and communication to overcame the linguistic barriers, to facilitate the negotiation of meanings and to share common modeling tools during the analysis activities.

## 4 Conclusions and Future Works

In this paper we addressed some of the challenges posed by a joint use of the human-centered and goal-oriented requirement engineering approaches, starting from the experience done within in a real ambient assisted living project. We discussed how a dialogic relationship between the disciplines may provide guidance for researchers from requirement engineering and human centered design field that cooperate within the same design process. In this perspective, the orchestration of different contributions, the establishment of communication practices and the engagement within a mutual learning process are presented as crucial steps to take full advantage of different research traditions. We finally made some hypothesis on how to generalize such an approach and discussed some issues related to this generalization.

In the future, we would like to explore in more details some issues emerged during the experience in the project and the subsequent discussions. In particular: from a methodological point of view we would like to explore the interaction of the two different approaches also exploiting principles form meta methodologies [2]; the other side we aim at refining the dialogue issues in linguistic interactions, also exploiting cognitive linguistics and ontology based approaches.

## References

1. W. Gaver, J. Beaver, and S. Benford. Ambiguity as a resource for design. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 233–240. ACM New York, NY, USA, 2003.
2. B. Henderson-Sellers and J. Ralyté. Situational Method Engineering: State-of-the-Art Review. *Journal of Universal Computer Science*, 16(3):424–478, 2010.
3. C. Hinds. The case against a positivist philosophy of requirements engineering. *Requirements Engineering*, 13(4):315–328, 2008.
4. C. Leonardi, L. Sabatucci, A. Susi, and M. Zancanaro. Design as intercultural dialogue: coupling human-centered design with requirement engineering methods. In *INTERACT September, 5-9 2011, (to appear)*, 2011.
5. L. Penserini, A. Perini, A. Susi, and J. Mylopoulos. High variability design for software agents: Extending Tropos. *TAAS*, 2(4), 2007.
6. A. Pickard and P. Dixon. The applicability of constructivist user studies: how can constructivist inquiry inform service providers and systems designers. *Information Research*, 9(3):9–3, 2004.
7. R. Weber. The rhetoric of positivism versus interpretivism: A personal view. *MIS Quarterly*, 28 (1):3–12, 2004.

# Causal vs. Effectual Behavior – Support for Entrepreneurs

Jan Schlüter[1], Dominik Schmitz[2],
Malte Brettel[1], Matthias Jarke[2], and Ralf Klamma[2]

[1] RWTH Aachen University, Lehrstuhl Wirtschaftswissenschaften für Ingenieure und Naturwissenschaftler, Templergraben 64, 52056 Aachen, Germany
{schlueter,brettel}@win.rwth-aachen.de
[2] RWTH Aachen University, Informatik 5, Ahornstr. 55, 52056 Aachen, Germany
{schmitz,jarke,klamma}@dbis.rwth-aachen.de

**Abstract.** "Effectuation" is a new approach to explain the success or failure of entrepreneurs. In contrast to the traditional "causation" approach the entrepreneur is not considered to be driven by a concrete goal and to choose between different alternatives in regard to how well they help to achieve this goal. Instead the entrepreneur evaluates the alternatives, in particular the choice of strategic partners, in regard to their potential for future success. The goals are adapted to the choices and in particular the needs of the strategic partners. Agent-based simulations are intended to help identifying the settings where one approach is more appropriate than the other.

## 1 Introduction

The IMP Boost project "Overcoming Barriers in the Innovation Process" investigates a new approach to explain the success or failure of entrepreneurs. At the center of interest is the notion of "effectuation" (http://www.effectuation.org) [6, 7]. This denotes a fundamentally different way to act in comparison to traditional approaches in economics, now denoted by "causation". A "causal" entrepreneur starts by carrying out comprehensive (and rather expensive) market studies to clearly identify a dedicated market opportunity. This is then settled as a goal and the entrepreneur only decides between different alternatives in regard to their utility to achieve the settled goal. In contrast to this an "effectual" entrepreneur is not committed to a particular product or goal, but only to the desire to run an enterprise. Instead of carrying out expensive market studies she chooses from alternatives in regard to the resulting opportunities and under consideration of the "affordable loss", i. e. how much money she can loose without harming her capacity to act. A major means of an "effectual" actor is to utilize her knowledge and network to find cooperation partners. These can be potential customers as well as money donators. Very much in contrast to the "causal" approach these strategic partners can have a great influence on the actual product or goal to be achieved. This is the "effectual" entrepreneur simply adapts the goal to the partner's needs, including the chance to build a completely different product.

The reward to this flexibility is a definite commitment of the partner to become a part of the new venture. Thus, the two approaches differ considerably in regard to how they address uncertainty. While the "causation" approach employs prediction to reduce uncertainty, "effectuation" controls the unknown future by taking decisions, i. e. explicitly deciding which way to go.

## 2 Objectives of the Research

The aim of the IMP Boost project is to compare the two approaches – causation vs. effectuation – by running simulations [8]. Based on theoretical research neither of the two approaches is to be favored in general. Accordingly, we need to identify the settings, conditions, and constraints that put either of these approaches in front.

From first modeling experiences and foundational considerations, agent-based approaches toward modeling and simulation seem to be well suited. To deepen the basic understanding of the two processes, we have first gone for a qualitatively oriented modeling with the help of the *i** framework. This approach builds on successful earlier experiences with modeling and simulating entrepreneurial networks with *i** [1–4]. Afterward for simulation purposes, we want to go beyond the qualitative logic-based simulations proposed in the earlier investigations, by considering quantitative simulations via the Repast agent-based simulation framework (http://repast.sourceforge.net). This allows for a more thorough investigation of a larger variety of parameter settings.

## 3 Scientific Contributions

### 3.1 A Process Model for "Causation"

Figure 1 shows a preliminary and partial modeling of a "causation" based approach toward venture creation (based on [5]). The main actors are the "entrepreneur", a "market research institute", and "resource providers", in most cases venture capitalists or business angels. As it becomes obvious from Fig. 1, the goal-orientation of the "causation" approach fits well with the traditional understanding and modeling of agents with beliefs, desires, and intentions. The only extension compared to vanilla *i** is the consideration of sequence links and a more general precondition/effect element (graphically depicted by a triangle) as introduced in [1].

### 3.2 A Process Model for "Effectuation"

In contrast to this the modeling of "effectual" behavior runs into some problems (see Fig. 2, again based on [5]). The "effectual" approach is to be considered goal-oriented only in a very generic way, such as "create a venture". The concrete business idea, if formulated as a goal, needs to be alterable over time. Further on, while the "causation" model is mainly sequential, effectuation inherently asks

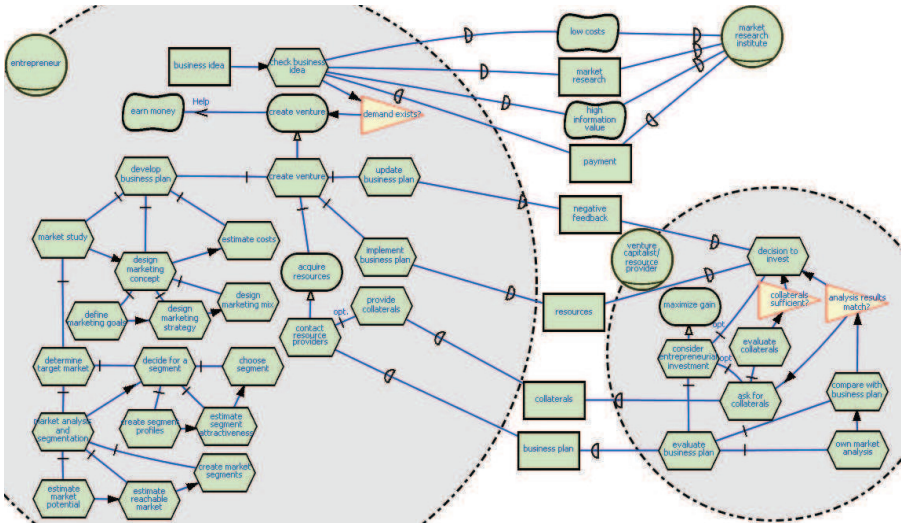



**Fig. 1.** Preliminary $i^*$ Model for "Causation"

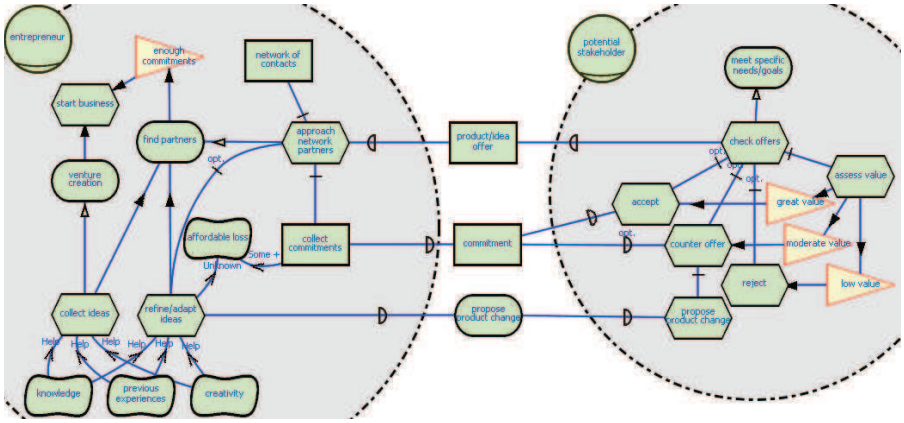



**Fig. 2.** Preliminary $i^*$ Model for "Effectuation"

for loops. The new partners that are sought and from whom commitments are collected can change either the means – i. e. what is possible – or the goals – i. e. what the entrepreneur (concretely) strives for – at a product/widget level. Pre-existing contacts are visited one after the other and each can trigger the same processes on negotiating commitments and/or adapting products. Thus, not only the concrete goals can change dynamically. In just the same way, also more and more means become available, possibly after any new potential partners is contacted.

While there have been proposals to embed control loops into the $i^*$ modeling language (see, for example, [10]), they focus the SR level, in particular decompositions. Thus, it has to be investigated whether these extensions are also suitable to capture repeated interactions between agents as occurring in this setting. Alternatively, a more strategic, high-level annotation of *dependencies* may be considered. Yet the more operationalized "control flow" considerations mentioned above might prove valuable when addressing the model-based transformation toward quantitative agent-based simulations.

### 3.3 Logic-Based Simulations

By referring earlier work on entrepreneurship networks [1–4] we plan to analyze "causation" and "effectuation" with the help of qualitative, high-level logic-based simulations. For this purpose we want to make use of an established model-based mapping from $i^*$ to ConGolog, a logic-based simulation framework enabling concurrency. The particular process models in $i^*$ as preliminarily proposed above are the first steps in this regard. Yet, we still have to investigate in how far the simulation framework and the existing transformation need to be reconsidered once departing from the focus on networks and trust issues therein.

### 3.4 Agent-Based Simulations

In regard to quantitative agent-based simulations first steps have been taken by manually implementing a simplified agent-based simulation model on top of the Repast agent-based simulation framework (http://repast.sourceforge.net/). The model has four key components: (1) product ideas – a feature vector with $n$ elements, (2) three types of agents – causators, effectuators, and consumers, (3) market demand – demand vectors of all agents, and (4) a pay-off landscape to measure performance via market fit. To ease comparison, there are pairs of causator and effectuator agents with identical product ideas. During the simulations, the causator and effectuator agents then finalize their product ideas by filling the remaining flexible features. The causator uses a market study for this purpose while the effectuator relies on her network and the sequential negotiation of commitments with a rather small number of stakeholders that are consulted.

While this simple model does not yet have all the necessary features – for example, the principle of "affordable loss" is not yet considered –, it has already confirmed the basic theoretical results, for example, the role of uncertainty. But

at the same time the simulations have delivered more insights on the fine-granular details of the considered situations. In particular, they have revealed the need to consider the influence of market concentration and fragmentation on the superiority of either approach. For more details see [8].

## 4 Conclusions

The first findings of agent-based simulations are promising in that they confirm that the theory of "effectuation" is plausible. In particular, the important role of uncertainty is acknowledged. On the other hand, the current simple simulations do not correctly reflect all relevant features such as "affordable loss" or a more detailed investigation of the concerned network of contacts let alone its evolution over time (even beyond several different venture "attempts").

We expect refined *i** models to help increasing the foundational understanding and static characterization of the two approaches whereas logic-based simulations are foreseen to shed some light on the dynamic characteristics. The results of these investigations then need to be reviewed and applied to refine and improve the quantitative agent-based simulations in Repast.

Altogether we can summarize that first steps toward a better understanding of "effectuation" have been made. Yet more analysis and simulations need to be run to better understand and answer the question when which of the two approaches is highly-likely more valuable.

## 5 Ongoing and Future Work

Currently, we investigate, improve, and validate our *i** process models for causation and effectuation, in particular in regard to which necessary feature can only badly be represented in *i**. At this point, we are still open in regard to whether new features are needed in *i** or – as it currently seems – some minor extensions that have already been proposed by various authors over the years can be combined to suit the new setting. Further on, we need to consider alternative modeling approaches as well. As already for now, the agent-based simulations are pursued rather independently from the *i** modeling. Accordingly, other modeling approaches such as systemic dynamics etc. similarly can provide valuable input. Of relevance is only that any findings of these pilot modeling studies need to be analyzed to complete, refine, and improve the models for the agent-based simulations (in Repast).

To analyze simulation results and in particular due to the high importance of networking, approaches from social network analysis as well as actor-network theory are likely to become relevant in order to correctly evaluate and interpret the results (see also [9]). This might also establish an interesting feedback loop on (automated) adaptations of the *i** models to reflect the outcome or evolution of situations and settings throughout simulations. Further on, we need to calibrate and validate the simulations by referring historic real world data before being

able to derive guidance on the strengths and weaknesses of the two approaches in regard to various different possible settings.

**Acknowledgment.** This research is funded in part by the RWTH Aachen University Excellence Project House "Interdisciplinary Management Practice" (IMP Boost Project).

## References

1. G. Gans, M. Jarke, S. Kethers, and G. Lakemeyer. Continuous requirements management for organization networks: A (dis)trust-based approach. *Requirements Engineering Journal*, 8(1):4–22, 2003.
2. G. Gans, M. Jarke, S. Kethers, G. Lakemeyer, and D. Schmitz. Requirements engineering for trust-based interorganizational networks. In E. Yu, P. Giorgini, N. Maiden, and J. Mylopoulos, editors, *Social Modeling for Requirements Engineering*. MIT Press, Cambridge, MA, 2011.
3. G. Gans, M. Jarke, G. Lakemeyer, and D. Schmitz. Deliberation in a metadata-based modeling and simulation environment for inter-organizational networks. *Information Systems*, 30(7):587–607, 2005.
4. M. Jarke, R. Klamma, G. Lakemeyer, and D. Schmitz. Continuous, requirements-driven support for organizations, networks, and communities. In J. Castro, X. Franch, A. Perini, and E. S. K. Yu, editors, *Proc. of the 3rd Int. i* Workshop, Recife, Brazil, February 11-12*, CEUR Workshop Proceedings, vol. 322, pages 47–50. CEUR-WS.org, 2008.
5. Christian D. Klusmann. Developing a process model for causal and effectual entrepreneurship (in german). Bachelor thesis, RWTH Aachen University, 2011.
6. S. D. Sarasvathy. Causation and effectuation: Toward a theoretical shift from economic inevitability to entrepreneurial contingency. *Academy of Management Review*, 26(2):243–264, 2001.
7. S. D. Sarasvathy and N. Dew. New market creation through transformation. *Journal of Evolutionary Economics*, 15(5):533, 2005.
8. J. Schlüter and M. Brettel. Simulating the clash of effectual and causal processes: Investigating conditions & boundaries for venture succes. In *Babson College Entrepreneurship Research Conference, Syracuse/NY, USA, June 8-11*, 2011.
9. D. Schmitz, T. Arzdorf, M. Jarke, and G. Lakemeyer. Analyzing agent-based simulations of inter-organizational networks. In L. Cao, A. L. C. Bazzan, V. Gorodetsky, P. A. Mitkas, G. Weiss, and P. S. Yu, editors, *6th Int. Workshop on Agents and Data Mining Interaction (ADMI@AAMAS), Toronto, Canada, May 11, Revised Selected Papers*, LNCS 5980, pages 87–102. Springer, 2010.
10. X. Wang and Y. Lespérance. Agent-oriented requirements engineering using ConGolog and *i**. In *Working Notes of the Agent-Oriented Information Systems Workshop, AOIS, Montreal, QC, Canada, May 28*, 2001.

# A Social Interaction Based Pre-Traceability for i* Models

Maurício Serrano[1] and Julio Cesar Sampaio do Prado Leite[1],

[1] Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro,
Rua Marquês de São Vicente 225, Ed. Padre Leonel Franca 13o. andar,
Rio de Janeiro, Brasil
mauserr@les.inf.puc-rio.br and http://www-di.inf.puc-rio.br/~julio/

**Abstract.** There is a lack of work focusing on i* models pre-traceability. Most of traceability work on i* models is centered on forward or backward traceability, i.e., tracing i* models to design models or design models to i* models. It seems that the stakeholders' needs and the organizational process are first-class citizens, but the requirements engineering process is not. We present in this paper our approach to i* models pre-traceability, which focuses on tracing the requirements engineering process (designing the design). Using ITrace models, our approach focuses on the requirements process. Our approach is based on argumentation graphs to trace stakeholders' arguments on social interactions, as these arguments justify what is represented on i* models.

**Keywords:** i* framework pre-traceability, requirements engineering process, rich picture, social interactions, argumentation.

## 1 Introduction

In 1993, Goguen [1] discussed how social issues affected the Requirements Engineering (hereafter RE) process. Three social groups were clearly identified: the client organization, the requirements team and the development team. In that paper [1], Goguen notes the necessity of documenting and maintaining traces of the RE process. Without the traces, it would be impossible to identify social issues within these groups or on the interactions between them.

Gotel and Finkelstein [2] were the first to distinguish between forward and backward traceability, i.e., from requirements to design artifacts and from design artifacts back to the requirements, respectively. They also reported how the lack of pre-requirements traceability led to the lack of commitment and the lack of accountability on teams.

Based on [2], Gotel and Finkelstein proposed Contribution Structures [3]. These dynamic structures deal with traceability relations between artifacts and trace the individuals and groups that participated in the RE. The individuals' roles and commitment to the requirements development are the focus of their proposal. Contribution Structures is still the foundation for requirements pre-traceability on current traceability metamodels.

The Tropos methodology [4] focused on developing requirements-driven software from goal-oriented requirements modeled with the i* framework [5]. As a model-driven methodology, Tropos support forward and backward traceability. Tropos' backward traceability goes so far as to the Early-Requirements models, thus providing a backward traceability from requirements to organizational processes and actors' needs. However, Tropos' backward traceability from Late-Requirements to Early-Requirements can not be considered pre-traceability. As i* models, Early-Requirements specifications are products of several RE processes.

In this paper, we propose a pre-traceability approach that traces i* models back to the social interactions of the RE process. We propose the use of ITrace [6] to perform this task. ITrace traces RE artifacts back to social interactions, social interactions goals, activities, techniques, social networks, information sources and resources through RichPicture [7]. Our approach also models the stakeholders' arguments on social interactions using an argumentation framework [8]. These arguments justify the i* models contents.

This paper is organized in Sections: Section 2 discusses the main objectives of our research; Section 3 presents some scientific contributions; Section 4 summarizes the proposal by presenting the final considerations; and finally, in Section 5 we consider the ongoing and future work.

## 2 Objectives of the research

The main objective of our research is to offer a lightweight pre-traceability model to i* models. When an i* model is created or discarded or every time an i* model evolves or is analyzed an ITrace model enriched with argumentation should be produced. These RE activities should generate a large set of enriched ITrace models. With our approach, we intend to enable the requirements engineer to answer the following not-comprehensive type of questions:

01. How the requirements were elicited?
02. Which techniques were applied?
03. Who modeled the requirements on the i* model?
04. Who interacted with the stakeholders?
05. Which stakeholders were consulted?
06. Was the development team involved in the process?
07. How many social interactions were necessary to obtain the current version?
08. Were the teams geographically separated?
09. If I need to modify an i* model, which information sources or stakeholders should I consult?
10. Was the i* model validated by all the stakeholders?
11. Who stated that the softgoal X was a relevant quality *criteria*?
12. When and why the tasks (means) Y and Z that achieve the goal (end) X of actor W became part of the model?

## 3 Scientific contributions

Previous work on requirements pre-traceability [9] [10] [11] focuses on proposing metamodels that extensively define what should be traced. The production of these traces is done by using management tools, what demands extra work. As this extra work does not benefit the i* model itself, pre-traceability is commonly set aside.

Our approach is based on RichPicture [7], an informal hand-drawn model to be used on the workplace. ITrace models follow this philosophy[1], allowing the modeler to freely draw the process, i.e., designing the design, during the social interaction with the stakeholders. However, ITrace demands that the drawing have three layers: (i) the Base layer, on the bottom, where the social network, information sources and resources are drawn; (ii) the Interactions layer, on the middle, where the social interactions, the goals, the activities and the applied techniques are drawn, and (iii) the Artifacts layer, on the top, where the i* model and argumentation graph thumbnails or references are drawn. Fig. 1 shows the three layers of an ITrace model. ITrace models are constructed with the collaboration of all participants on the social interaction and, as such, are validated on-the-fly.

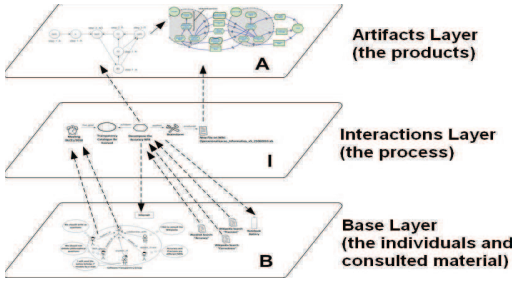



**Fig. 1.** The three layers of ITrace: Base (B), Interactions (I) and Artifacts (A)

We extended our previous approach [6] by adding argumentation graphs [8] to the ITrace model. Argumentation graphs enabled us to attach the stakeholder's arguments to the ITrace model. These arguments justify the main changes to the i* model. We suggest that the argumentation graphs should be hand-drawn while the stakeholders are arguing, capturing only the most relevant arguments. We decided to include on our argumentation graphs traces between the stakeholders' arguments and the video recordings of the meetings. However, this decision implies extra work after the social interactions are finished. Fig. 2 shows an argumentation graph about the contributions of some Lattes-Scholar [12] operationalizations to transparency-related softgoals.

Another scientific contribution of our approach was to attach the RE process to the artifact. Contribution Structures [3] allow direct links between artifacts and individuals. Therefore, they lose, for example, the Why, When, Where, How and How Much dimensions of the trace. ITrace also applies visual symbols (cartoon dialogs) and argumentation graphs to represent the actors' concerns on a social interaction.

[1] All ITrace models and argumentation graphs showed in this paper were re-drawn with Microsoft Visio® to improve the paper presentation.

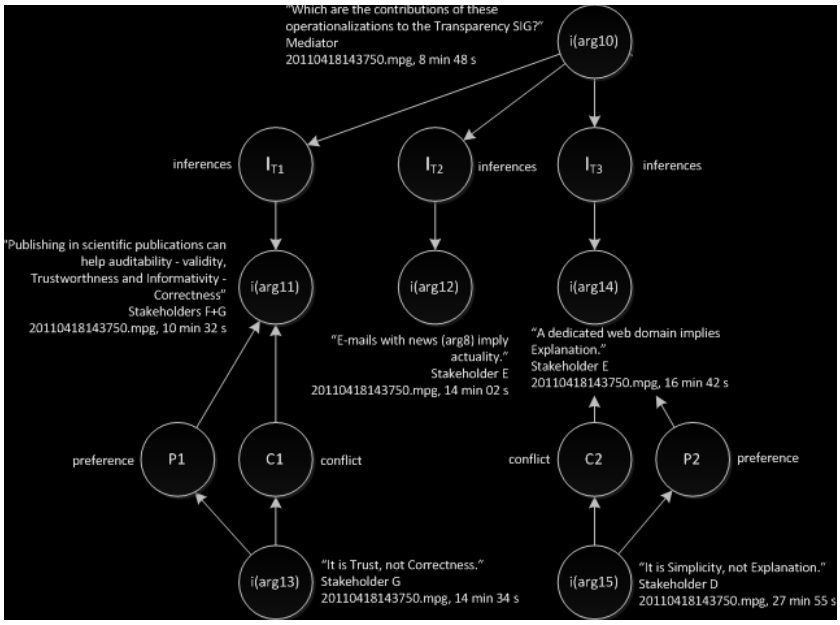



**Fig. 2.** An argumentation graph with traces to video recordings (.mpg files)

Finally, we scanned our ITrace graphs and uploaded them to Flickr®. Using the "add a note" action, it was possible to define hyperlink regions on the picture. We took advantage of this feature to link the i* model thumbnail to the real-size i* model, the actors to their web pages, the information sources to the original documents, the video recordings to their Youtube® videos and so on. Fig. 3 illustrates an ITrace model with some of these "notes" (regions on the picture) [13].

## 4 Conclusions

In this paper we briefly present our proposal to a lightweight pre-traceability model for i* models. ITrace is a simple graphical notation tool to be used at the workplace.

When presented with the idea, Prof. Berry[2] recalled the "POTLO BOAD TIP" - **P**roblem **O**f **T**he **L**ack **O**f **B**enefit **O**f **A** **D**ocument **T**o **I**ts **P**roduces, a problem that may block the efforts of requirements systematization [14]. Pre-traceability is often set aside as it does not benefit the i* modeling, itself. We address this problem by focusing on providing pre-traceability with a minimal effort. As our models are easily hand-drawn during the social interactions, no additional work is needed after the

[2] A meeting with Prof. Daniel M. Berry during his visit to Departamento de Informática at PUC-Rio (6/9/11)

meetings; of course that the organization should have the maturity as to understand that it is worth investing in requirements practices as it pays off in the future.

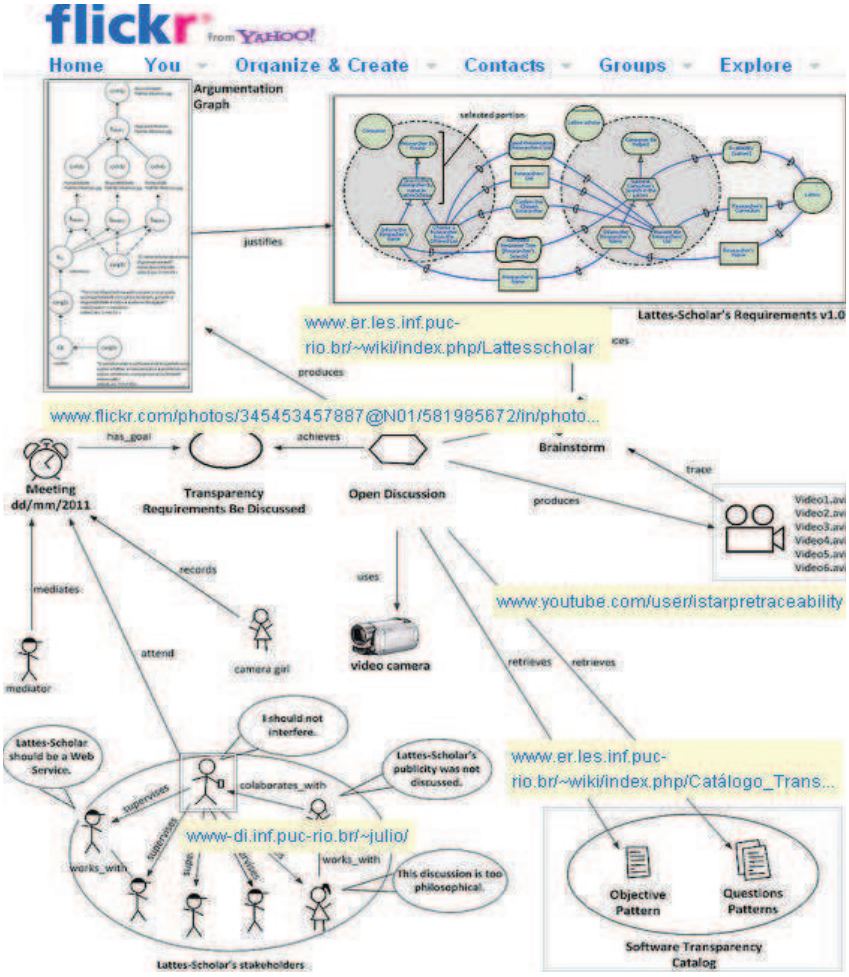



**Fig. 3.** Exploring the hyperlink features of Flickr® [13]

Previous work on requirements pre-traceability [9] [10] [11] proposed metamodels to cast the traces. The biggest problem with their approaches is the difficulty of maintaining the traces while the RE artifacts evolves. Our approach applies an innovative view to this problem: **ITrace models do not evolve**. ITrace models are **snapshots** (as a picture is) of a social interaction that produced or modified an i* model. If an i* model evolves, a new ITrace model will be created to trace this evolution. The new ITrace model does not substitute the older ones. They coexist, each one tracing its respective social interaction.

## 5 Ongoing and future work

Since the middle of 2010, we have been applying our proposal to trace the Software Transparency Group's weekly meetings. We successfully traced the capturing and the evolution of several requirements patterns [15] and the evolution of several i* and NFR Framework models. We are also applying our proposal to trace a transparency-centered software development process, using Lattes-Scholar [12] as a case study.

As future work, we intend to analyze: (i) the use of the Flickr® tags to enrich the ITrace models, and (ii) how these tags would impact searching and tracing within a given set of ITrace snapshots.

## References

1. Goguen, J. A.: Social Issues in Requirements Engineering. In Proc. of the IEEE International Symposium on Requirements Engineering, pp. 194-195, California, USA (January 1993).
2. Gotel, O. and Finkelstein, A.: An Analysis of the Requirements Traceability Problem, in Proceedings of First Int. Conference on Requirements Engineering, pages 94-101 (1994).
3. Gotel, O. and Finkelstein, A.: Contribution structures. Proc. 2nd Intl. Symp. Requirements Engineering, York, pages 100-107 (1995).
4. Bertolini, D.; Delpero, L.; Mylopoulos, J.; Novikau, A.; Orler, A.; Penserini, L.; Perini, A.; Susi, A.; Tomasi, B.: A tropos model-driven development environment. In: Boudjlida, N.; Cheng, D.; Guelfi, N. (eds.) CAiSE Forum, CEUR Workshop Proceedings, vol. 231 (2006).
5. Yu, E.: Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. 3rd IEEE Int. Symp. on Requirements Eng. (RE'97), pp. 226-235 (1997).
6. Serrano, M. and Leite, J.C.S.P.: A Rich Traceability Model for Social Interactions. 6th Int. workshop on Traceability in emerging forms of software engineering. (TEFSE '11). ACM, USA, pp. 63-66 (2011). http://doi.acm.org/10.1145/1987856.1987871
7. Monk, A. and Howard, S.: The Rich Picture: A Tool for Reasoning about Work Context. Methods and Tools, ACM 1072-5220/98/0300, pp. 21-30 (April 1998).
8. Jureta, I., Mylopoulos, J., Faulkner, S.: Analysis of Multi-Party Agreement in Requirements Validation, 17th IEEE Int. Requirements Eng. Conference, pp. 57-66, (2009).
9. Pohl, K.: PRO-ART: Enabling Requirements Pre-Traceability. 2nd. IEEE Int. Conference on Requirements Engineering (ICRE'96), ISBN:0-8186-7252-8, pp. 76-84 (April 1996).
10. Ramesh, B. and Jarke, M. 2001. Toward Reference Models for Requirements Traceability. IEEE Transactions on Software Engineering, Vol. 27, No. 1, pages 58-93 (January 2001).
11. Pinto, R. C.; Silva, C.; Castro, J.: Support for Requirement Traceability: The Tropos Case. 19th Simpósio Brasileiro de Engenharia de Software (SBES'05), Brasil (2005).
12. Lattes-Scholar: Requirements Engineering Group at PUC-Rio. Available at: http://www.er.les.inf.puc-rio.br/~wiki/index.php/Lattesscholar (May 2011).
13. Flickr: ITrace of meeting dd-mm-2011. Available at: http://www.flickr.com/photos/63908029@N02/5819877406/in/photostream (June 2011)
14. Berry, D.M., Czarnecki, K., Antkiewicz, M. and AbdelRazik, M: Requirements Determination is Unstoppable: An Experience Report. 17th IEEE Int. Requirements Eng. Conference, pp. 311-316 (2010)
15. Serrano, M. and Leite, J.C.S.P.: Development of Agent-Driven Systems: from i* Architectural Models to Intentional Agents' Code, to appear at the First Int. Workshop on Requirements Patterns (2011)

# Requirements Engineering for Social Applications

Amit K. Chopra and Paolo Giorgini

University of Trento, Italy
chopra@disi.unitn.it, giorgini@disi.unitn.it

**Abstract.** We characterize social applications as those involving interaction among multiple autonomous agents. We are interested in the essential concepts and approaches for modeling such applications. We make the case that i* has some limitations with respect to the modeling of social applications. The problem is in the intentional nature of i*. The deeper roots though lie in the centralized machine-oriented approach of current requirements engineering approaches. We recommend an interaction-oriented approach to requirements modeling, modeling in terms of social commitments rather than dependencies, and in general, accommodating a distributed perspective right from the earliest phases of software engineering. For clarity, we also distinguish social commitments from various similar-sounding notions in the literature.

## 1 Introduction

Many of the applications that we use are social in the sense that they involve communication among two or more social agents. Banking, healthcare, e-business, emergency services, and meeting scheduling are in this sense social applications.

Let us say we want to design a meeting scheduling application. Let us consider that any meeting scheduling enactment involves agents that play convener, scheduler, and participant. There are two ways to approach the design.

In one approach, you consider a set of requirements and build a machine that meets the requirements. This is the approach one would apply when specifying a washing machine, an LCD monitor, a gate controller, or an aircraft's fly-by-wire controls. In other words, this is the approach we use to design technical artifacts. The convener, scheduler, and the participants are still social agents, but from the perspective of the machine they are merely *users*.

In the other approach, you specify (or choose or compose from existing ones) a *protocol* that supports your goal of scheduling meetings. A protocol is a specification of interaction specified with reference to roles that social agents may adopt. The meeting scheduler protocol would have the roles convener, scheduler, and assistant. It does not matter which particular social agent adopts the role. Nor does it matter what its goals are, if any. This approach preserves the original social nature of the application. *Designing a system with new requirements means designing a protocol that meets those requirements*.

We term the former approach *machine-oriented* and the latter *interaction-oriented*. The latter is about the design of protocols. In fact, the system is in a sense the protocol. There is the separate question though about the design of the participants in a protocol.

Suppose you are a social agent who notices that some other social agents have adopted the roles of convener and participant, but are waiting for someone to adopt the role of scheduler. You are willing to play the role provided some of your requirements are met by participating in the system (for example, payment for scheduling services). You do not know the design or motivations of the other participants, but what you can do is check if the meeting scheduling system, that is, the protocol, supports your requirements. Further, you may design a software artifact that encodes part of your decision-making, interacts accordingly with the other participants, and in effect represents you in the system. For the purposes of this paper, we refer to this artifact as a social agent's *surrogate*.

Interaction-orientation preserves the nature of the social application. It accommodates both the design and runtime autonomy of agents. Machine-orientation does not.

The allusion to current practices in requirements engineering would not be lost upon anyone. Current practices in requirements engineering are machine-oriented. The essential idea is to come up with the specification of a *machine* that along with the domain assumptions satisfies the requirements of the stakeholders. RE emphasizes two kinds of systems: *system-as-is* and *system-to-be*. The former is the system as it exists without the machine. It helps in understanding the environment in which the machine will be introduced. The latter is what the system will be when the machine is introduced. Presumably, in the system-to-be, the stakeholders' requirements are satisfied.

## 2 Objectives

Our broad objective is to understand and improve upon the software engineering of social applications. Two questions follow naturally from the discussion above.

1. What are the methodologies for designing protocols?
2. What are the methodologies for designing surrogates?

Our immediate objective though is to understand whether i* [17] is suitable (in terms of concepts) for the modeling of social applications. i* has had considerable success in the requirements engineering community, and researchers are applying it to model all kinds of applications. Obviously, the question turns on whether i* can be used to model interactions. To do this, in the following section, we outline the i* methodology, and then analyze some of its critical constructs.

## 3 Scientific Contributions

The idea of interaction-oriented programming traces it roots to Singh [11]. It is an idea distinct from that of agent-oriented software engineering, where the focus is on creating intelligent agents. As stated earlier, interaction-orientation says nothing about any particular agent's rationale or the design of its surrogate or any other internal information system [13].

There has been considerable progress in methodologies for designing protocols, especially as concerns the nature of protocol specifications. A great leap here came

with the idea that protocols ought to specify the meaning of communication, not its flow [12]. Social commitments have emerged as a key element of meaning. Further, it is broadly accepted that protocols cannot be formalized in terms of the goals or intentions of agents [1]. Methodologies for protocol specification have also received attention [8, 4].

The second question has received less attention in the software engineering community. In some recent work, we have shown how an individual agent may reason about his goals in light of his potential communications, specifically the social commitments he can potentially be involved in [3, 2]. In further work, we have extended this idea to address adaptation in social applications [7]. The key idea there is that it is not systems that adapt. After all, the system is nothing but a protocol. Instead it is the social agents (or their surrogates) that adapt, each autonomously from other agents in the system.

### 3.1 The Nature of i*

At the basic level, i* follows the standard RE conceptualization sketched above. Indeed, i* is a machine-oriented methodology; it is not interaction-oriented. Where i* differs is in the explicit modeling of organizational agents (*actor* in i* terms) and their requirements. The requirements themselves are understood in terms of the goals of the agents. The agents are related to each other by means of intentional dependencies.

The notion of agent within i* seems to be a broad one. A stakeholder could be agent; so could a legacy system; so could a service. The core idea is to model the relevant aspects of the environment via the notion of agents. The system-as-is is modeled in i* as a network of the existing agents with the intentional dependencies among them as the links. The system-to-be would introduce a machine, also an agent, towards satisfying the requirements in a suitably modified network.

The notion of intentional dependency in i*, a central one, deserves special attention. (We talk about goal dependencies but the discussion applies to other kinds of i* dependencies as well.) An i* goal dependency between two agents $x$ and $y$ for some goal $p$ means that $x$ *wants* $p$, and $y$ is *able* to achieve $p$ and in addition *intends* to achieve $p$.

For example, in i* one would say that the convener depends on the scheduler for its goal that the meeting be scheduled. This dependency means that (1) the convener wants the meeting to be scheduled, (2) the scheduler is able to schedule the meeting, and (3) the scheduler *intends* to schedule the meeting. Equivalently, in i* terminology, one cay say that the convener has *delegated* its goal scheduled to the scheduler.

It is interesting to contrast the notion of goal dependency with the notion of social commitment. A social commitment $\mathsf{C}(x, y, p, q)$ means that $x$ is committed to $y$ for $q$ if $p$ holds. It doesn't mean that $y$ wants $q$ or $x$ wants $p$. Nor does it say anything about the ability of the agents to deliver $p$ or $q$. Nor does it say anything about any agent's intentions. A social commitment only comes about due to interaction among the agents. This is fundamentally different from the notion of a goal dependency, as discussed above. Notably, *intends to* in i* is stated in terms of an *internal commitment*. However, internal commitment is not the same as social commitment [10]. The interaction-oriented approach talks only about social commitments.

Consider that based on the requirements analysis done with the help of the i* strategic dependency and strategic rationale diagrams, a machine for scheduling is deployed.

Later, the organization finds that the maintenance of the scheduler is too costly and therefore decides to outsource meeting scheduling to an external organization. In such a situation, one would not have information about the external organization's goals, intentions, or abilities. Hence, all the analysis done earlier with the in-house meeting scheduler would come to naught.

Consider instead that an organization went about meeting scheduling in an interaction-oriented way. First, it selected an industry-standard meeting scheduling protocol. Then it created a surrogate to play the role of the scheduler. Later it found that the surrogate was too difficult to maintain. So it outsources the scheduling to a service provider that had advertised itself as following the protocol. The interaction-oriented approach naturally supports such substitution.

### 3.2 Social Commitments

We have found that the notion of social commitments we refer to is often confused with other notions in the literature. We take the opportunity here to distinguish social commitments from related concepts in the literature.

**Internal Commitment** Cohen and Levesque's work [6] formulated a rich theory of rational action based on the concepts of *intention* and *internal commitment* to intention. Broadly, the idea is that for an agent to succeed with its intentions, it must be internally committed to realizing them. Internal commitment refers to an agent's psychological entrenchment.

**Obligations** Commitments are not obligations. Commitments come about only due to communication *and* hold irrespective of what exists in the agents' internal states. This is the essence of a public semantics of communication. The representation of a commitment may appear similar to that of an obligation; however, that is where the similarity ends. Obligations, as studied in the literature, represent a mental concept. For example, just because $\mathsf{C}(Barbara, Alice, paid, deliverPhone)$ does not imply $\mathsf{O}(Barbara, Alice, paid, deliverPhone)$. Neither does the implication hold in the other direction. Obligations are in the spirit of internal commitment.

**Responsibilities** The notion of responsibility has been applied toward the modeling of sociotechnical systems [15, 14]. The idea is that if one knows an agent's responsibilities, one can derive its requirements. However, responsibility is modeled as something that comes about because of the passing of an obligation from one agent to another via delegation. For example, a professor can delegate his obligation of going to a meeting to a Ph.D. student. The student then becomes responsible for fulfilling that obligation. However, tying responsibility to delegation definitionally unnecessarily limits the kind of responsibilities that can be captured. When an agent makes a social commitment, he is socially accountable to the creditor for its fulfillment. There is no delegation involved here, but there still is a sense of responsibility. Further, keeping an agent's autonomy in mind, only an agent itself can create social commitments. This is not to say we rule out the notion of delegation. In general, for delegation to succeed, other commitments must first be set up; for example, the delegatee must have a prior commitment to the delegator for honoring delegations [5].

**Norms** We understand norms as (public) conventions. Social commitments are doubly normative. One, protocols are conventions: they specify the commitments that would arise from the agents' communications. Two, commitments form the basis of compliance checking. A violation of a commitment may represent a serious exception, and creditors and the contextual community may request sanctions on the offending debtor for the violation.

## 4 Conclusions and Future Work

In the current paper, we advance the theme we have been pursuing for two years, but with more details specific to i*. We discussed the main limitations with respect to engineering social applications. The intentional approach of i* works for the traditional RE setting where the idea is to design a centralized machine that meets the stakeholders' requirements. However, an intentional approach is an integration-based approach. Social applications, on the other hand, necessarily have to adopt an interoperation-based approach because there is no central machine.

The distinction between mentalist (cognitive) and social notions is worth pointing out. When one talks about agents, concepts such as goals, beliefs, intentions, strategies, plans, and so on are mentalist concepts. Communication, convention, and social commitments are social concepts. When one models a system of social agents using mentalist concepts, the results are vastly inferior to when one models the same system using social concepts [12]. i* models rely exclusively on mentalist concepts (Yu uses the term *intentional*). Nowhere does communication come into the picture. Therefore, it is not clear in which sense i* would be a social approach [16]. Some other recent work [9] also fails to make the distinction between mentalist and social.

Lately there has been some work talking about the unmanageability of i* diagrams. i* has some modularity via the notion of agents; but the problem is there is no encapsulation. Dependencies do not stop at agent boundaries; they connect agents internals. In social systems, the basic assumption is that agent internals are not known. This implies encapsulation is not broken. Social commitments help us reason about such well-encapsulated agents. Social commitments would greatly enhance the manageability of specifications.

In general, RE must move away from the centralized machine-oriented perspective to a more distributed perspective that conceptualizes a system as being constituted from independently designed agents. The methodological ingredients of such an approach is the direction of research we are currently pursuing.

**Acknowledgments.** Numerous discussions with Munindar Singh, Fabiano Dalpiaz, and John Mylopoulos over the past two years led to the ideas in the paper. Amit Chopra was supported by a Marie Curie Trentino award. Paolo Giorgini was supported by the EU-FP7-IST-IP-ANIKETOS and EU-FP7-IST-NOE-NESSOS projects.

## References

1. Amit K. Chopra, Alexander Artikis, Jamal Bentahar, Marco Colombetti, Frank Dignum, Nicoletta Fornara, Andrew J. I. Jones, Munindar P. Singh, and Pınar Yolum. Research di-

rections in agent commmunication. *ACM Transactions on Intelligent Systems*, 2011. To appear.
2. Amit K. Chopra, Fabiano Dalpiaz, Paolo Giorgini, and John Mylopoulos. Modeling and reasoning about service-oriented applications via goals and commitments. In *Proceedings of the 22nd International Conference on Advanced Information Systems Engineering (CAiSE)*, volume 6051 of *LNCS*, pages 113–128. Springer, 2010.
3. Amit K. Chopra, Fabiano Dalpiaz, Paolo Giorgini, and John Mylopoulos. Reasoning about agents and protocols via goals and commitments. In *Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 457–464, 2010.
4. Amit K. Chopra and Munindar P. Singh. Colaba: Collaborative design of cross-organizational business processes. In *Proceedings of the Workshop on Requirements Engineering for Systems, Services, and Systems of Systems*, 2011. to appear.
5. Amit K. Chopra and Munindar P. Singh. Specifying and applying commitment-based business patterns. In *Proceedings of the Tenth International Conference on Autonomous Agents and MultiAgent Systems*, 2011.
6. Philip R. Cohen and Hector J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.
7. Fabiano Dalpiaz, Amit K. Chopra, Paolo Giorgini, and John Mylopoulos. Adaptation in open systems: Giving interaction its rightful place. In *Proceedings of the 29th International Conference on Conceptual Modeling*, volume 6412 of *LNCS*, pages 31–45. Springer, 2010.
8. Nirmit Desai, Amit K. Chopra, and Munindar P. Singh. Amoeba: A methodology for modeling and evolution of cross-organizational business processes. *ACM Transactions on Software Engineering and Methodology*, 19(2):6:1–6:45, 2010.
9. Renata S. S. Guizzardi and Giancarlo Guizzardi. Applying the UFO ontology to design an agent-oriented engineering language. In *Proceedings of the 14th East European Conference on Advances in Databases and Information Systems*, pages 190–203, 2010.
10. Munindar P. Singh. Social and psychological commitments in multiagent systems. In *AAAI Fall Symposium on Knowledge and Action at Social and Organizational Levels*, pages 104–106, 1991.
11. Munindar P. Singh. Toward interaction-oriented programming. TR 96-15, Department of Computer Science, North Carolina State University, Raleigh, May 1996. Available at www4.ncsu.edu/eos/info/ dblab/www/mpsingh/ papers/mas/ iop.ps.
12. Munindar P. Singh. Agent communication languages: Rethinking the principles. *IEEE Computer*, 31(12):40–47, December 1998.
13. Munindar P. Singh and Amit K. Chopra. Programming multiagent systems without programming agents. In *Proceedings of the 7th International Workshop on Programming Multi-Agent Systems*, volume 5919 of *LNCS*, pages 1–14. Springer, 2010.
14. Ian Sommerville, Russell Lock, Tim Storer, and John Dobson. Deriving information requirements from responsibility models. In *Proceedings of the 21st International Conference on Advanced Information Systems Engineering*, pages 515–529, Amsterdam, 2009.
15. Ros Strens and John Dobson. How responsibility modelling leads to security requirements. In *Proceedings of the New Security Paradigms Workshop*, pages 143–149, 1993.
16. Eric S. Yu. Social modeling and i*. In Alexander T. Borgida, Vinay K. Chaudhri, Paolo Giorgini, and Eric S. Yu, editors, *Conceptual Modeling: Foundations and Applications*, pages 99–121. Springer-Verlag, 2009.
17. Eric S.K. Yu. Towards modelling and reasoning support for early-phase requirements engineering. In *Proceedings of the Third IEEE International Symposium on Requirements Engineering*, pages 226–235, 1997.

# Intentional Models based on Measurement Theory

Andre Rifaut[1]

[1] Public Research Centre Henri Tudor, 29, av. J. F.Kennedy,
L-1855 Luxembourg-Kirchberg, Luxembourg
andre.rifaut@tudor.lu

**Abstract.** Metrics and measures have always been the subject of quite a lot of research works in the Requirements Engineering (RE) community, including about intentional models of Goal-Oriented RE (GORE) such as those of ***i****. However, using recent developments of the Measurement Theory, in this paper we show that the concept of Measurement Framework (MF) for soft-systems is useful for the analysis of business service systems that need long-term service agreements based on consistent measurements at all stages of their life-cycle (from inception to operation). We show that with two kinds of goals and softgoals based on MF, it is possible to improve (a) the elicitation of functional and non- functional requirements, (b) the structure of the ***i**** models, and (c) the consistency between run-time measurements and the model-based assessments of business services at early stages of RE.

**Keywords:** goal-oriented requirements engineering, measurement theory, measurement framework, business service, service level management.

## 1 Introduction

IT based business systems are enablers to create business opportunities across business entities boundaries that belong to complex business constellations. Being able to analyze those opportunities contributed to the successful application of the requirements engineering activities based on ***i****, in particular with the use of the concepts and analysis techniques based on the Strategic Dependency Diagrams and the Strategic Rationale Diagrams. However, for the business services, which are means often used in those business constellations, agreements between the service stakeholders (e.g. the service provider and the service client) must be faithful to what actually happens during the service performance. This is why those agreements, often called service-level agreements, must be based on *empirically valid measurements*. If not, some party will assess negatively its business collaborations and can find other business services offered on the market. Progress of our current research projects is reported with an example of a case study followed in the Construction Sector [4].

Section 2 motivates the objectives of the research in the context of GORE, before presenting in Section 3 our method based on measurement frameworks (MF). We conclude and explain future works in Section 4.

Note that this paper uses some terminology of Measurement Theory that may conflict with GORE terminology (e.g. about model-based evaluations).

## 2 Objectives of the research

As explained in the introduction, it is necessary to have consistent assessments of the business services at the different stages of their life-cycle, from requirements engineering to operation. Empirically valid measurements should be the basis of all assessments used thorough the life-cycle of the business services.

Recent developments in Measurement Theory for soft-systems can be used to support the development of RE techniques in ***i**** that focus on those empirically valid measurements provided at run-time. Our aim is to provide RE support for *complex measurements having an empirical validity* in the context of GORE for business services. This aim has come up from needs identified during more than 6 years of research in related contexts: RE for risk management [12, 10], RE for regulated business [3] and RE for business service management systems [6]. In all three aforementioned research contexts there is a negotiation process occurring between stakeholders (business managers, IT service providers, auditors, regulators) and based on empirically valid measurements of attributes of IT based business systems.

## 3 Scientific contributions

### 3.1 Measurement Theory

Since long Measurement Theory [13] has formalized measurement concepts in the context of the measurement of physical phenomena, such as the length of an object. In short, *measurement systems* are composed of procedures and artifacts (e.g. a wood yardstick) that can *assign* a *measurement result,* i.e. a time-varying or time-independent *profile,* (e.g. 11") to an *attribute* (e.g. the length) of the target *empirical phenomena* (e.g. a sheet of paper). The measurement process is a faithful operationalization of the *measurement model* of the attributes. All this, called *measurement framework* (MF), becomes more complex for the measurement of "soft-systems" [5] (to contrast with physical systems), such as organizational processes (e.g. a set of business processes). In that context what is a measurement? Actually, it is mainly the same: an *empirically valid assignment* of a *value* to an *attribute* of the *target system* (e.g. a set of business process).

The recent developments have pointed out three important characteristics of measurements (see Fig. 1): measurements must be *empirical*, *objective*, and *inter-subjective*. Briefly, "empirical" means that the (empirical) attribute must be clearly identified; "objective" indicates that the measurement model must respect all (empirical) properties of the attribute (e.g. measured length must respect the size ordering); and "inter-subjective" expresses the fact that the measurement model (and scales) is a belief shared by everybody using (or referring to) the MF. All three characteristics make a distinction between *measures*, *evaluations*, and *preferences* (see Fig. 1). Measures and evaluations must be empirically validated. During RE elicitation processes sometimes we have to model attributes of systems that are not

yet scientifically well understood. For instance, the usability attribute of IT applications: do we have a precise definition of what "usability" is? Research is still needed for validating empirical and objective measurements of usability. According to the Fig. 1, if neither empirical, nor objective, a wide place for (implicit) preferences is left in those models.

| | Empirical | | Objective | | Inter-subjective | |
|---|---|---|---|---|---|---|
| | yes | no | yes | no | yes | limited |
| Measure | √ | | √ | | √ | |
| Evaluation | √ | | | √ | | agree on common references |
| Preference | | √ | | √ | | shared personal viewpoint |

**Fig. 1.** Characteristics of measurements (using terminology of Measurement Theory)

This motivates us to define specific types of softgoals and goals: *when their description is complete, those types of goal and softgoals can be specified on the basis of measurement frameworks*. For those two specific types of goals and softgoals, a distinction can be made between them: the fulfillment of a goal is *measured*, whereas the fulfillment of a softgoal is *evaluated*. (For now on in this paper, we consider only goals and softgoals of these types.)

### 3.2 Practical uses of those measurement characteristics.

Our research works present two methods: one [9] for extracting and structuring compliance (textual) requirements from regulation through the use of a MF and another [11] for translating those requirements into rigorous requirements modeled in ***i**** also with the help of a MF. Two other works ([6, 4]) show how to derive requirements when using MF.

A partial view on the case study in the Construction Sector [4] is shown in Fig. 2. At the top, is the model of a *generic* MF, ISO/IEC 15504 [8] defining the measurement of assurance management of any business process in terms of its purpose and outcomes (15504 terminology distinguishing softgoals from goals; 15504 indicators are not explained). Then, the middle part shows this generic MF that is instantiated into a specific MF ([2]) for assurance management of business services.

Finally, at the bottom, both MF are instantiated to the specificities of the collaborative work needed in construction projects, in this case, the sharing of documents and expertise. The three horizontal lanes at the bottom separate the high-level strategic diagram, the MF resulting from the instantiation of the two preceding MF and one "solution" (at the very bottom). The actors and their dependencies are not shown. The solution describes functional requirements that are derived from the measurement. The evaluation procedure imposed by the MF rigorously structures the arguments of the model-based assessments: see the shaded area on the evidences showing that the goal "reactions time are monitored" of the MF (i.e. an outcome) will be fulfilled.

Using goals and softgoals based on MF has practical consequences for RE activities and requirements models. First, the measurement model of the MF provides *constructive insights to derive functional and non-functional requirements* (see the top and middle models). Second, our specific kind of goals and softgoals basically

imposes an expected measurement profile to the result of a MF that can be applied to detailed requirements of alternatives solutions. This increases the structure of large ***i**** models *respecting the separation of concerns* of the different MF. This is sketched at the bottom of the Fig. 2. The separation of concerns can be expressed with (a) the attributes (and their MF); (b) the measurement profiles; (c) the solution (as measurement target). Third, the MF provides a complete specification of a measurement system that can be implemented and used at runtime for an objective measurement of the fulfillment of goals and softgoals. (Those measurement systems can also be used during operational tests.)

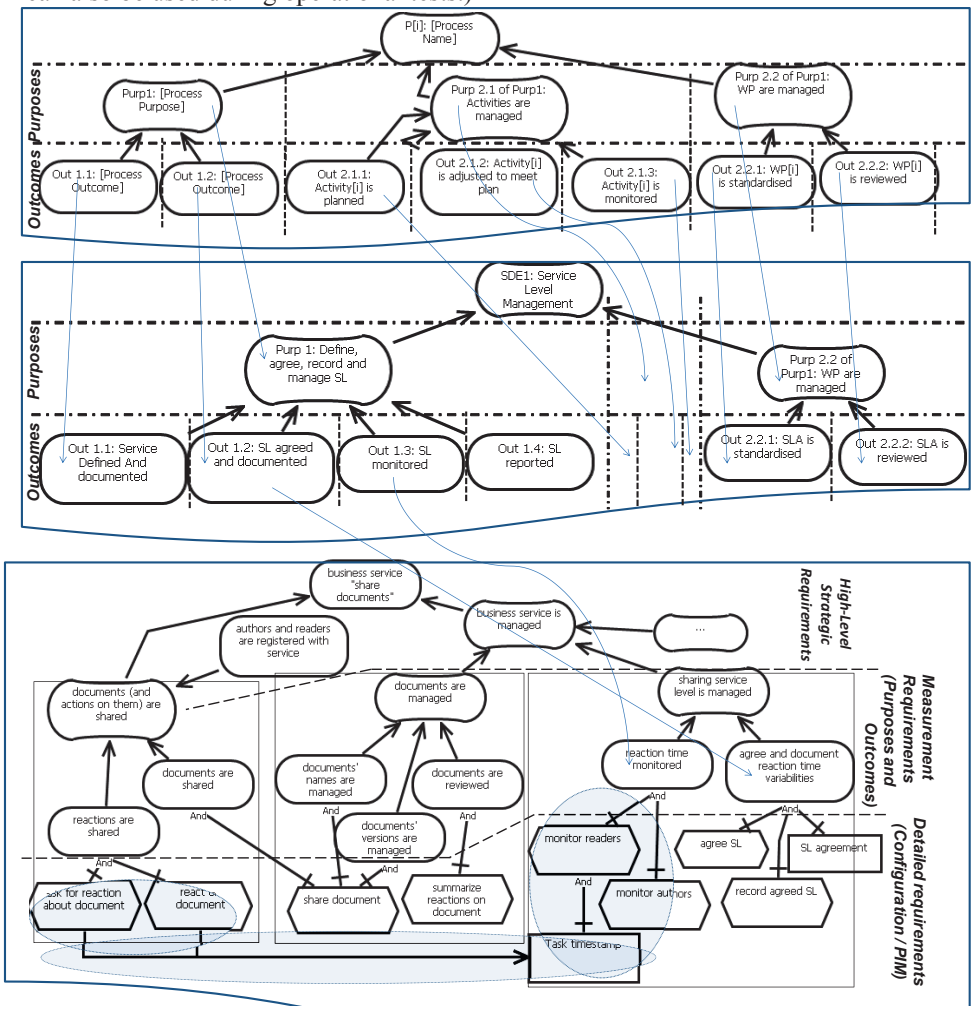



**Fig. 2.** Generic, Specific and Instantiated Measurement Framework Model.

Fourth, the same measurement specifications can be used for creating the arguments of model-based assessments of the fulfillment of goals and softgoals as explained in Sec. 3.1. (Measurements are made at run-time, whereas model-based

assessments provide expected measurement results.) However, when carefully made the results of the assessments will be consistent with the results of the measurements. (Those model-based assessments can also be used for model-based simulations.)

As seen in the example, the design method is based on the insertion of soft-system measurements between the strategic goals and the solutions. This creates a valid relationship between the (elicited) measurement profiles and alternatives of functional (and also non-functional) requirements. For a bottom-up design process it is possible to collect arguments and evidences from the analyzed solution and derive the measurement results of the model-based evaluation defined by the MF. For a top-down elicitation, measurement results of model-based evaluations can be imposed in order to derive the requirements on the basis of refinement patterns (cfr. KAOS by A. van Lamsweerde) or strategies (cfr. MAP by C.Rolland) that generate alternative solutions (not shown). For instance, concerning monitoring activities (outcome 2.1.3), in the selected measurement profile the monitored activities can include the sharing of documents and exclude the sharing of expertise. The advantage of this design method is to benefit from the constructive insights given by the MF of the soft-systems.

### 3.3 Discussion.

In recent overviews [1, 7] of the model-based techniques for assessing GORE models, nothing is said about the empiricity and objectivity of the measurements used. Those three aspects are independent of any category of statements used in RE (functional/ non-functional; requirement/assumption; belief/desire/intention; "precise"/"vague" statements, etc.). For each statement of the requirements model, the importance level of each three aspects depends on its use during the life-cycle of the "system to-be": indeed, those aspects must be compatible with the measurements methods used in analyses or assessments needed from early requirements activities to system operation. For instance, what is the level of the three aspects in the annotations of models elements with qualitative/quantitative values and their propagation/aggregation? Soft-systems measurements [5] are built on top of indicators linked to goals/softgoals with the aim to strengthen the validity of the same links found in software engineering or business process improvement methods, or in RE models ([1, 7]).

Using the same MF for requirements analysis and for defining the specifications of the measurement system and method, and for performing the actual measurements during system operation, this provides a greater consistency between the argument structure used in the requirements model, and the argument structure used at run-time.

## 4 Conclusions, Ongoing and future work

In this paper we have shown that the recent developments of Measurement Theory adapted to GORE methods, in particular ***i****, gives a sound basis for an interesting type of goals and softgoals to guide the elicitation and analysis of functional and non-functional requirements of business services. To benefit from the scientific knowledge of a number of MF defined, improved and validated by the scientific community

(often using social sciences research techniques), the efficiency of the proposed methods should be improved by providing tools supporting (a) the import of measurements frameworks into a knowledge base, (b) the definition of refinement patterns and strategies needed by the elicitation activities, (c) the assessment of the requirements models [1] on the basis of the measurements frameworks. (This last issue is similar to other engineering domains.) Ontology-based Software Engineering and Multi-Criteria Decision Theory provide techniques that depart from searching solutions to hard problems (e.g. searching inconsistent scenarios or plans), but aim at classifying and characterizing solutions of under-constrained problems that often occur in very early requirements engineering (see e.g DDP by M.S. Feather).

**Acknowledgments.** This work is supported by the project MOTIVATE of the National Research Fund, Luxembourg.

## References

1. 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, Vol. 25, pp. 841-877 (2010)
2. Barafort et al.: ITSM Process Assessment Supporting ITIL (TIPA), van Haren Publishing, Zaltbommel, The Netherland, 2009.
3. Di Renzo B., Hillairet M., Picard M., Rifaut A., Bernard C., Hagen D., Reinard D.: Operational risk management in financial institutions - Process assessment in concordance with Basel II. Software Process: Improvement and Practice 12(4): 321-330 (2007)
4. Dubois E., Kubicki S., Ramel S., and Rifaut A.:Capturing and Aligning Assurance Requirements for Business Services Systems, International Workshop on Business System Management and Engineering. June 28, (2010). Málaga (Spain).
5. Finkelstein, L.: Widely, strongly and weakly defined measurement, Measurement 34 (2003), Elsevier, pp. 39–48
6. Grandry E., Dubois E., Picard M., Rifaut A.: Managing the Alignment between Business and Software Services Requirements from a Capability Model Perspective. ServiceWave (2008): 171-182
7. Horkoff, J., Yu, E., Analysing goal models: different approaches and how to choose among them, SAC 2011; 675-682
8. ISO, ISO/IEC 15504: Information Technology – Process Assessment: Part1 to 5, 2003.
9. Rifaut, A.: Goal-Driven Requirements Engineering for Supporting the ISO 15504 Assessment Process, LNCS 3792, Proc. Int. Conf. EUROSPI, Budapest, Hungary. (2005)
10. Rifaut A., Picard M., Di Renzo B.: ISO/IEC 15504 Process Improvement to Support Basel II Compliance of Operational Risk Management in Financial Institutions. In: Proc. Conf. SPICE 2006, Luxembourg (2006).
11. Rifaut A and Dubois E.: Using Goal-Oriented Requirements Engineering for Improving the Quality of ISO/IEC 15504 based Compliance Assessment Frameworks: In Proc. IEEE Intl. Conf. On Requirements Engineering, Barcelona, IEEE Press (2008).
12. Rifaut A., Di Renzo B., Picard M.: ISO/IEC 15504, a Basis for Generally Accepted Sound Process Models in Financial Institutions: A Case Study about Venture Capital Fund Management. In: Proc. Conf. SPICE 2008, Nuremberg, (2008).
13. Rossi, G.: On some key concepts and terms in measurement having a cross-disciplinary impact, 12th IMEKO TC1 & TC7 Joint Symposium on Man Science & Measurement, September, 3-5, 2008, Annecy, France

# iStar 2011

# TOOL FAIR

# Preface

Many tools have been created to facilitate modeling and analysis with *i** and related frameworks. New to the *i** Workshop, the iStar Tool fair aims to update community knowledge about the current offering of *i** tools. The Fair occurs as part of the 5th International *i** Workshop (iStar'11), collocated with RE'11 in Trento, Italy.

We received up to ten tool submissions that were reviewed by the Tool Fair Chairs, providing feedback and suggestions. All the proposals were considered interesting to the community, and were accepted in the form of a three-page description. The authors were also requested to either create or update the description of their tool available in the *i** wiki.

The Tool Fair is organized as a plenary session in the *i** workshop. It includes a short overview and summary of tool submissions made by the Tool Fair chairs, then two-minute "lightening presentations" by each demo presenter, and finally the floor is opened for individual tool demos.

The contents of first iStar Tool fair is representative of many of the existing *i** modeling and analysis tools. The creation and analysis of GRL models in the jUCMNav Tool is described by Amyot et al. The features and history of the OpenOME Tool is summarized by Horkoff et al. Morandini et al. provide a summary of the Taom4E tool for modeling and generating code from Tropos models.

New developments for existing tools are described in separate submissions. Colomer and Franch describe the implementation of *i** models in jUCMNav. Laue and Storch use the Eclipse Modeling Toolkit to allow for the easy addition of new functionality to OpenOME. Hilts and Yu introduce the GO-DKL browser in order to support a repository of goal-oriented design knowledge, producing goal models which can be opened in OpenOME.

The Tool Fair includes tools new to the *i** community. I*-Prefer is introduced by Li et al. to facilitate model creation considering preferences, facilitating decision making. The Measufier tool is proposed by Colomer and Franch to allow analysis of structural measures over *i** models. Malta et al. introduce the iStarTool, allowing users to learn to improve the quality of their models via syntax checks and guidelines.

Finally, Cares et al. aim to support tool operability by summarizing advances in the existing iStarML representation language.

We thank the authors for their valuable contributions, and look forward to seeing all of the tools in Trento!

Jennifer Horkoff, *University of Toronto, Canada*
Xavier Franch, *Universitat Politècnica de Catalunya, Spain*

iStar'11 Tool Fair Chairs

# *I**-Prefer: A Tool for Preference Model-Driven Decision Making

Tianying Li[1,1], Tong Li[1], Haihua Xie[1], Lin Liu[1],

[1] Key Lab for Information System Security, MOE
Tsinghua National Lab for Information Science and Technology
School of Software, Tsinghua University
{ litianying10@mails, litong08@mails, xiehh06@mails, linliu@}tsinghua.edu.cn

**Abstract.** Multi-criteria decision making problems exist everywhere. In a complex system, for all involved players, it is inevitable to face task/service selection situations where multiple qualities of tasks/services criteria need to be taken into account. In addition, complex interrelationships between different impact factors and actors need to be understood and traded off. In this paper, we present a tool called *I**-Prefer which is represented with annotated NFR/*i** framework. This framework uses goal and agent-based preference models to drive these decision making activities. Particularly, we describe the purpose and main features of the tool, and give a brief introduction of the extended *i** framework which supports *I**-Prefer analysis.

**Keywords:** Preference, *i** framework, Optimal strategy

## 1 Introduction

An increasing number of researchers have been working on multi-motive decision models and methods. In order to model and analyze multi-criteria decisions in a systematic way, we have already done some work on how to make decisions based on an annotated NFR/*i** framework [1, 2]. In this paper, we present a tool named *I**-Prefer. Within this tool, we adopt graphical notations of the NFR modeling methods to model the interrelations among different criterion. Then we decide preferences of decision makers by appending numerical annotations to the nodes in the model. Strategy dependency models in *i** can be used to represent and evaluate alternative services networking decisions. Algorithms for identifying optimal solutions of the given decision problem are also integrated into the tool.

The structure of this paper is organized as follows: Section 2 presents the original purpose of the tool. Section 3 lists the main features of the tool and its availability and status. Section 4 discusses both the limitations and the future plans.

## 2 Main Purpose of the Tool

The main purpose of the tool is to model the interrelations among different criterions. Then we can decide preferences of decision makers by appending numerical notations to the nodes in the model. *I** framework is a widely used strategic intentional modeling method. Actors' goals and tasks could be modeled intuitively. Services selection always involves at least two types of actors, with mutual dependencies between them. As a result, *i** model is a natural fit for modeling services selection. In order to fulfill the needs to make decision by the preference, which is the input from domain experts and end users, the tool intends to makes some extensions to *i**: (1) Actors' preferences are concerned in the modeling process. Actors' preferences are represented as their demands of system's performance, such as system's soft-goals and weights of soft-goals. (2) The analysis of state of soft-goals is introduced based on the effect of system's design on soft-goals' satisfaction degree. The execution of a system task may increase or decrease soft-goals' satisfaction degrees, and the extent of effect is decided by task to soft-goal impact value. (3) A method of analyzing system's state is introduced, which supports the optimal system's strategy selection. Utility value is adopted to express the quantified value of system's state, which actually specifies the performance of the system [2].

## 3 Main features and current status of *I**-Prefer

*I**-Prefer is based on JavaScript, and can be applied in IE5.0 (or higher version). The tool is developed in 2009 which is the version 1.0, the next version is under development.

People can download both its source code and user manual (in Chinese) in the URL[3]: https://sourceforge.net/projects/i-prefer and can also try the tool on line.

The tool can support the methods raised in [2], specifically the main features of the current version are as follows.

(1) The tool is web-based, while most *i** tools are not. It can support graphical modeling within web page, and running without downloading any packages. The annotation of the model is the same as NFR/*i** framework. And some additional marks like utility values can also be added to the model. Some of the properties can also be changed by the user, like the name, utility value, and which actor the object belongs to.

(2) The built preference model can be saved and loaded with XML file. The user can save or load a model whenever it is needed.

(3) Automatically calculate the optimal strategies. The computing is based on the established algorithms presented in [1]. In the algorithm, each soft-goal is related to two weights, one is to specify the degree of the completion of the soft-goal, and the other is to provide user's emphasis on that soft-goal. Besides, each Task-Soft-goal has a parameter to describe the impact on the soft-goal after fulfilling the task. The user can use the calculated results to do the most reasonable choice about the tasks/services. The final decision would be presented in highlight in the result.

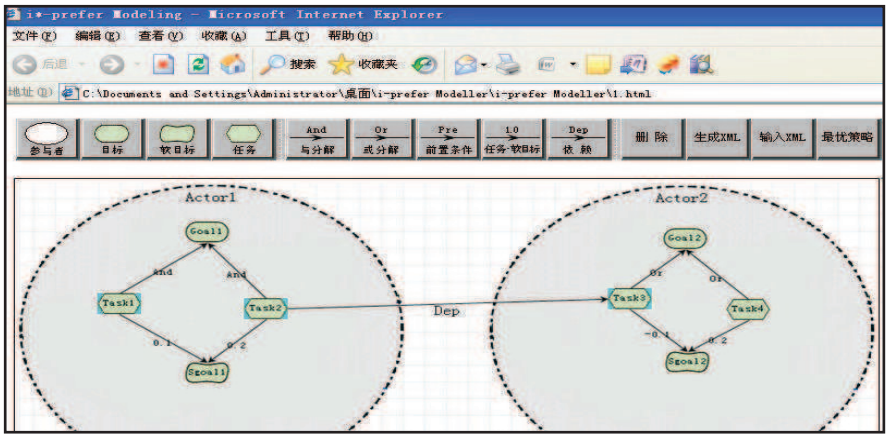



Fig. 1. A screenshot of the tool

With all these features, the tool *I**-Prefer could model the problem scenario, and further conduct the decision making and optimal strategy computing. Fig.1 has shown the screenshot of the tool, which contains 17 buttons about the creation or delete of objects, lines, as well as the buttons to generate and open XML files. Besides, there is also an Optimal Strategy button, which applies background algorithm and finally gives the optimal choice of about the tasks/services in the system.

## 4 Limitation and Future work

The tool *I**-Prefer supports the quantitative reasoning of actor's preferences on soft-goals, and the evaluation of alternative ways to achieve system goals. It can be used to facilitate automated preference tasks/services selection process and can help optimize decision making. However, there are also some limitations in the tool: (1) The tool is not mature in its algorithms, which can only calculate the optimal strategy for some simple cases; (2) Some parameters of the tool are not customized, so the user cannot change the properties like the color and size of the Actors.

In future, we would extend the algorithms to deal with more complex cases, and try to apply the tool into an industrial case. Besides, we would also make the tool more customized and user friendly.

## References

1. Haihua Xie, Lin Liu, Jingwei Yang, i*-Prefer Optimizing Requirements Elicitation Process Based on Actor Preferences, 24th ACM Symposium on Applied Computing in Hawaii in March 2009.
2. Wenting Ma, Lin Liu, Haihua Xie, Hongyu Zhang, Preference Model Driven Services Decision Making, 21st Conference on Advanced Information Systems Engineering (CAiSE 2009) in Amsterdam.
3. SourceForge download page of *I**-Prefer, https://sourceforge.net/projects/i-prefer

# OpenOME: An Open-source Goal and Agent-Oriented Model Drawing and Analysis Tool

Jennifer Horkoff[1], Yijun Yu[2], Eric Yu[3]

[1]Department of Computer Science, [3]Faculty of Information, University of Toronto
[2]Department of Computing, Faculty of Maths, Computing & Technology,
The Open University
jenhork@cs.utoronto.ca, y.yu@open.ac.uk, eric.yu@utoronto.ca

**Abstract.** OpenOME is an Eclipse-based open-source tool supporting the construction and analysis of i* models. The tool is in a stable state and available freely for download. Recently added features include support for forward and backward interactive, qualitative i* analysis.

**Keywords:** Goal-and Agent-Oriented Models, Model Analysis, Interactive Analysis, Model Views, Tool Operability.

## 1 Introduction and Features

OpenOME is a tool for the creation and analysis of goal and agent-oriented models as part of a systems analysis process. The tool supports modeling of the social and intentional viewpoint of a system, allowing users to capture the motivations behind system development in a graphical form. Creation and analysis of agent-goal models supports requirements elicitation, exploration, communication and trade-off analysis. OpenOME is especially useful to support elicitation and analysis in early requirement phases, where important non-functional and social information is made explicit.

The tool allows users to draw models using the i* Framework syntax described in [1]. The tool allows users to create i* models graphically using a palette of shapes. Standard features such as saving, zoom, cut, copy, and paste are provided. Models are grouped under user-created projects, shown in a folder view. OpenOME imports and exports models in the GMF .ood and .oom format, as well the Q7 textual modeling language [2]. See Fig. 1 for a screenshot of the OpenOME interface.

OpenOME supports the forward and backward interactive, qualitative i* analysis procedures described in [3,4]. Users can assign qualitative labels to intentions which represent their initial analysis question and then propagate these labels in a forward (direction of the link) or backward direction. The procedures are interactive, asking for user input to resolve conflicting or partial information. Results from multiple evaluations, including judgments made are stored, and can be viewed via the Alternatives Tab. Visualizations have been added to highlight model leaves and roots (potential starting points for analysis), areas of human judgment, and the intentions involved in a conflict in backward analysis.

Use of OpenOME, including the forward and backward evaluation procedures and current visualizations, has been tested in a series of user studies, reported in [5,6].

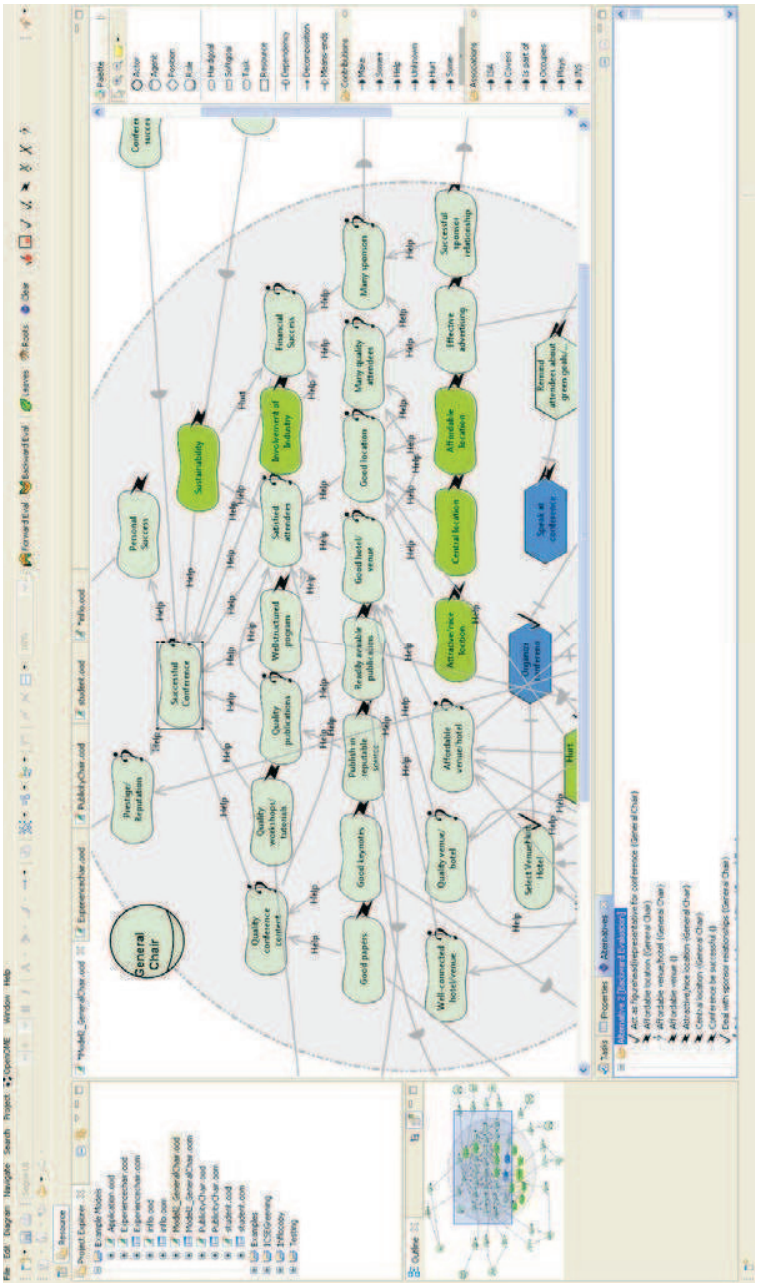



**Fig. 1.** Screenshot of the OpenOME Tool

## 2 Development, Availability and Future Work

OpenOME development originated in OME, a desktop Java application developed at the University of Toronto to support i* modeling [7]. In 2004, development of OME ceased and the source code was ported to the Eclipse platform, creating an open source version taking advantage of the Eclipse Modeling and Graphical Modeling Frameworks (EMF & GMF). Use of these frameworks allows us to automatically generate model editing code from an i* metamodel. This code has been customized and expanded to support features specific to i* modeling. OpenOME architecture takes advantage of the Eclipse package development, allowing for extension or customization with the addition of a new development package. After several rounds of development, the current architecture no longer bears similarity to that of the OME tool. Current development is at version 3.4.1.

Windows, Linux and Mac releases of OpenOME can be downloaded from Sourceforge [8]. User documentation and tutorials are available under the User Links section here: https://se.cs.toronto.edu/trac/ome/. After several rounds of users studies, and through use of the tool in several systems analysis courses, OpenOME has reached a relatively mature and stable state. Bug reports and suggestions for new features can be sent to openome-support@cs.toronto.edu. More information can be found on the user web site [9] and on the developer wiki [10].

Development on general tool functionality and bug fixes is ongoing. We are currently working on expanding the tool in several directions, including import/export into iStarML [11], customizable syntax checking, tabular views of i* models, and features encouraging model iteration such as conflict checks amongst human judgments.

**Acknowledgments.** We thank many current and former graduate and undergraduate students for their work on the OpenOME tool.

## References

[1] E. Yu, "Towards modelling and reasoning support for early-phase requirements engineering," Proceedings of ISRE 97 3rd IEEE International Symposium on Requirements Engineering, vol. 97, 1997, pp. 226-235.

[2] D.P. Leite, J.C. Sampaio, Y. Yu, L. Liu, E.S.K. Yu, and J. Mylopoulos, "Quality-Based Software Reuse," CAISE, vol. 3520, 2005, pp. 535-550.

[3] J. Horkoff and E. Yu, "Finding Solutions in Goal Models: An Interactive Backward Reasoning Approach," Proc. 29th Int. Conference on Conceptual Modeling (ER 2010) Vancouver BC Canada November 2010, Springer-Verlag, 2010, p. 59.

[4] J. Horkoff and E. Yu, "A Qualitative, Interactive Evaluation Procedure for Goal- and Agent-Oriented Models," CAiSE'09 Forum, Vol-453, CEUR-WS.org, 2009, pp. 19-24.

[5] J. Horkoff and E. Yu, "Interactive Goal Model Analysis Applied - Systematic Procedures versus Ad hoc Analysis," The Practice of Enterprise Modeling, 3rd IFIP WG8.1 (PoEM'10), 2010.

[6] J. Horkoff and E. Yu, "Visualizations to support interactive goal model analysis," Requirements Engineering Visualization REV 2010 Fifth International Workshop on, IEEE, 2010, p. 1–10.

[7] "OME, Organization Modelling Environment," 2008.

[8] "Sourceforge: OpenOME," http://sourceforge.net/projects/openome, 2011.

[9] "OpenOME, an open-source requirements engineering tool," http://www.cs.toronto.edu/km/openome/, 2010.

[10] "OpenOME Trac Wiki." https://se.cs.toronto.edu/trac/ome/, 2011.

[11] "iStarML." http://www.upc.edu/gessi/istarml/, 2011.

# Implementing Structural Measures over *i** Diagrams

Daniel Colomer, Xavier Franch

Software Engineering for Information Systems Research Group (GESSI)
Universitat Politècnica de Catalunya (UPC)
c/ Jordi Girona 1-3, 08034, Barcelona, Spain
{dcolomer, franch}@essi.upc.edu
http://www.essi.upc.edu/~gessi

**Abstract.** Measuring is a key issue in any software-related activity. In the context of the *i** framework, we are implementing Measufier, a prototype for measuring *i** diagrams in terms of properties that may be derived from their structure (structural measures). The prototype works over *i** diagrams represented by the iStarML interchange format, and provides some facilities for managing measures' catalogues, customizing the measures to the analyst needs, and computing the measure over particular diagrams.

**Keywords:** *i**, iStar, structural measures, Measufier, iStarML.

## 1 Introduction

Measuring is a fundamental activity for assessing the quality of conceptual models of any kind ("you can't control what you can't measure"). *i** models are not an exception to this rule. Some theoretical works have been proposed in the *i** community for defining measures over *i** diagrams [1]. However, there are not tools in the *i** marketplace offering the capability of defining and applying those measures. Our proposed Measufier tool is a first step to bridge this gap. This is the first version of Measufier (1.1) presenting a set of basic functionalities, to be enlarged in future versions. Its status thus may be considered quite preliminary. The tool may be downloaded from http://www.essi.upc.edu/~gessi/Measufier/resources.html, where some basic tutorial may be also found.

## 2 General Description

Measufier offers three main functionalities:

- Measure definition. It allows defining structural measures over i* diagrams according to the principles presented in [1]. At the current prototype all the measures are kept in a single catalogue.
- Model management. Several *i** diagrams represented in the iStarML interchange format [2] can be loaded in the context of a user session.

– Measure evaluation. Measufier supports the evaluation of measures selected from the catalogue over *i** diagrams loaded in a session.

Given the use of iStarML, Measufier may be easily interconnected with tools that have the ability to export models into this format. Also, the openness of this interchange format makes it possible to apply Measufier to different variants of *i** supporting thus most *i** modeling frameworks.

Figure 1, left, shows two example *i** diagrams. They represent two alternative social systems for a Pediatrics Hospital. We want to analyse them quantitatively as a way to support informed decision-making. We decide to use concepts from social networks. Figure 1, right, shows the name of four measures from this field and select just one. This measure, InDegreeMeasure, provides an estimation of the strategic importance of one element to its environment. It is defined as the number of incoming connections that an element has.

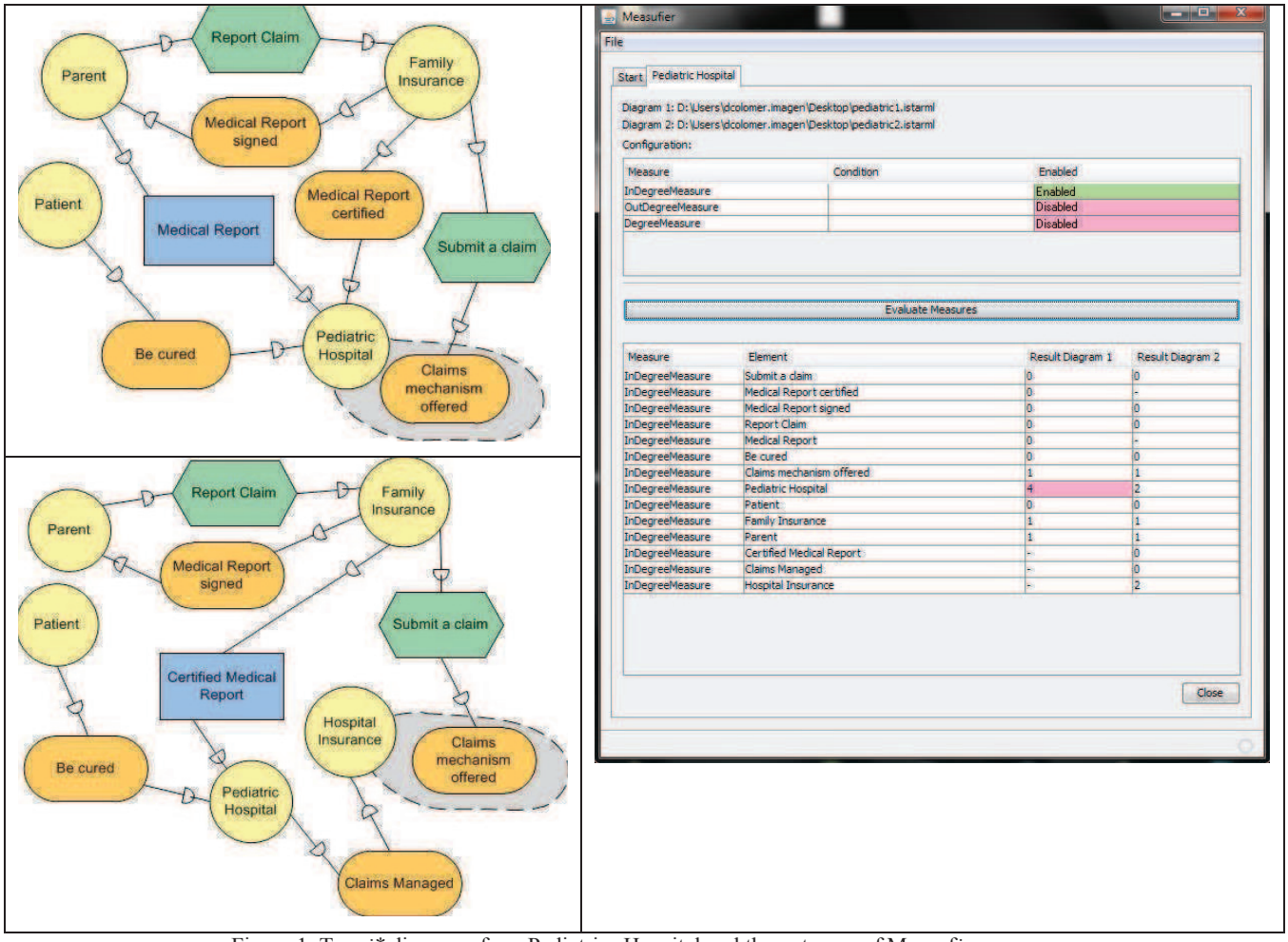



Figure 1. Two *i** diagrams for a Pediatrics Hospital and the outcome of Measufier.

## 3 Conclusions

Measufier is covering a gap in the current *i** tools landscape. Given its capabilities, it may complement other analysis tools and may complement existing techniques that require some quantitative analysis of *i** diagrams, e.g., analysing the adequacy of the model as starting point of an MDD process [3].

Because it is a first version, the tool has still many limitations, being the most important not having the ability to deal with arbitrary measures.

Our future work includes:

- Building a comprehensive catalog of measures. This catalogue will be indexed by concept or intended use of the model: e.g., social measures, measures for software architectures represented by *i** diagrams 4; etc.
- Implementing the connection with several *i** tools like OME, jUCMNav, TAOM4E, REDEPEND, etc. We will explore two non-exclusive ways. First, implementing an export facility in this tool for generating iStarML (already done for OME [5] and jUCMNav [6]). Second, offering Measufier as a service.

## Acknowledgements

This work has been partially supported by the Spanish project TIN2010-19130-C02-01. We would like to thank Carlos Cares and Lidia López for their help.

## References

1. X. Franch, G. Grau, C. Quer: "A Framework for the Definition of Metrics for Actor-Dependency Models," RE 2004.
2. C. Cares, X. Franch, A. Perini, A. Susi: "Towards Interoperability of *i** Models using iStarML," Computer Standards & Interfaces, 33(1), 2011.
3. F. Alencar, B. Marín, G. Giachetti, O. Pastor, J. Castro, J. Pimentel: "From *i** Requirements Models to Conceptual Models of a Model Driven Development Process," PoEM 2009.
4. G. Grau, X. Franch, "On the Adequacy of *i** Models for Representing and Analyzing Software Architectures," RIGiM 2007.
5. C. Cares, X. Franch, "A Metamodelling Approach for *i** Model Translations," CAiSE 2011.
6. D. Colomer, L. López, C. Cares, X. Franch, "Model Interchange and Tool Interoperability in the i* Framework: A Proof of Concept," WER 2011.

# GRL Modeling and Analysis with jUCMNav

Daniel Amyot[1], Gunter Mussbacher[2], Sepideh Ghanavati[1], and Jason Kealey[3]

[1]School of Electrical Engineering and Computer Science, University of Ottawa, Canada
damyot@eecs.uottawa.ca, sghanava@eecs.uottawa.ca
[2]Dept. of Systems and Computer Engineering, Carleton University, Ottawa, Canada
gunter@sce.carleton.ca
[3]LavaBlast Software Inc., Gatineau, Canada
jkealey@lavablast.com

**Abstract.** The Goal-oriented Requirement Language (GRL), part of the User Requirements Notation (URN) standard, is used to model and analyze stakeholder goals and requirements. jUCMNav is a free Eclipse-based modeling, analysis, and transformation tool for URN that was first released in 2005. This paper reviews the status of this tool (version 4.4, the 20th official release), highlights its main features, and briefly discusses future development plans.

**Keywords.** Analysis, Eclipse, Goal Modeling, Goal-oriented Requirement Language, jUCMNav, User Requirements Notation.

## 1 Introduction

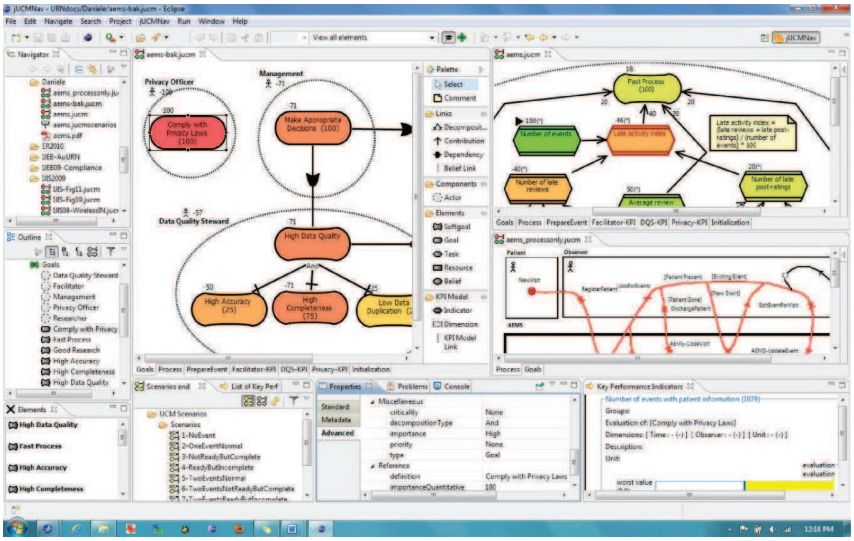
The User Requirements Notation (URN) is a graphical requirements modeling language that combines two complementary views: the Goal-oriented Requirement Language (GRL) for stakeholder objectives and decision rationales, and the Use Case Map (UCM) notation for scenarios and business processes combined with architectural components [4]. In November 2008, URN was officially standardized by the International Telecommunications Union as Recommendation Z.151. The design of URN itself started more than a decade ago, and this notation is now used for requirements engineering activities in a multitude of different contexts all around the world [3].

In 2005, students at the University of Ottawa started developing *jUCMNav*, an Eclipse-based tool for UCM modeling. Roy extended it to support GRL modeling and analysis [9], and Kealey later added UCM analysis [6]. This free, open source project has since benefited from many contributions. Now at its 20th official release, jUCMNav (version 4.4) [5] has become the most comprehensive URN tool available.

## 2 jUCMNav Modeling and Analysis Features for GRL

jUCMNav supports the standard GRL notation, with several extensions. The main editor view provides the list of GRL/UCM diagrams in the model, together with palettes and context-sensitive pop-up menus. This editor has the particularity of prevent-

ing the construction of syntactically incorrect GRL models. Also, different diagrams in a URN model can reference the same element definitions (actor, intentional element, link, etc.), hence improving consistency and allowing for complex models to be split into multiple diagrams. jUCMNav includes standard views such as a navigator for projects and model files, various outlines, element properties (with tabs), and problems. Typical Eclipse features such as zooming are supported, in addition to unlimited undo/redo and copy/pasting of model elements in the same model, across models, or to the clipboard (e.g., to paste a diagram into a word processor document).



Interesting extensions to URN include: i) *Key Performance Indicators* (KPIs) in GRL, to monitor external sources of data and convert them to the conventional GRL [-100, 100] evaluation scale [8], and ii) aspect-oriented extensions with an aspect composition mechanism, currently only supported for UCMs [7].

For analysis, jUCMNav supports GRL *strategies* (i.e., initial satisfaction values for several intentional elements), and six evaluation algorithms [2]: quantitative, qualitative, two hybrid ones, quantitative with KPI functions/aggregation, and constraint-oriented. While the first five support bottom-up propagation only, the last one uses a constraint solver and is hence more generic than bottom-up and top-down propagation algorithms, at the cost of a lower performance. These algorithms are independent, but qualitative ones are often used when little knowledge about the goals is available, whereas quantitative ones become useful as deeper knowledge is available. The Scenarios and Strategy view enables modelers to create, manage, and execute strategies. Color feedback is provided in real-time during evaluations (from *denied* in red to *satisfied* in green). Moreover, UCM scenarios can be defined and simulated. In jUCMNav, GRL satisfaction values can be used as variables in conditions attached to UCM elements (e.g., in forks or timers) and be updated in UCM responsibilities. The UCM/GRL views of a URN model can hence influence each other during analysis.

URN supports model element grouping (*concerns*), traceability (*URN links*), and annotating/stereotyping (*metadata*). In addition, OCL rules applied to the URN metamodel can be defined/selected by users to assess additional static semantic or stylistic constraints (violations are reported to the Problems view). When combined, these mechanisms can support lightweight profiles, both by extending and restricting the usage of URN elements while remaining compliant with the Z.151 standard. The tool comes with predefined, user-selectable constraints, including rules supporting a GRL profile for i* [1]. OCL is also used to compute user-defined metrics on models.

jUCMNav supports many model transformations through importing (Z.151 XML format, GRL catalogues) and exporting (Z.151 XML format, GRL catalogues, Message Sequence Charts and Core Scenario Model from UCM, and .CSV for GRL strategy evaluations) mechanisms. Reports can also be generated in PDF, RTF, and HTML, and diagrams can be exported in various bitmap formats. An integration with IBM/Telelogic DOORS for full requirements management is also available.

## 3 Conclusions and Future Plans

Since 2006, jUCMNav has been used successfully in software/requirements courses in more than a dozen universities, and it is used in dozens of labs and companies for research and industrial projects. The tool and its user/developer documentation (help, metamodel, examples, tutorials, etc.) are freely available online at [5]. Future plans include usability improvements, a textual syntax for GRL and UCM, improved URN link management, interoperability with *i**, and the support of aspect-oriented GRL.

## References

1. Amyot, D., Horkoff, J., Gross, D., Mussbacher, G.: A Lightweight GRL Profile for i* Modeling. 3rd Int. Workshop on Requirements, Intentions and Goals in Conceptual Modeling (RIGiM 2009), ER Workshops. Springer, LNCS vol. 5833:254-264 (2009)
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 (August 2010)
3. Amyot, D., Mussbacher, G.: User Requirements Notation: The First Ten Years, The Next Ten Years. Invited paper, Journal of Software (JSW), Vol. 6, No. 5, 747-768 (May 2011)
4. ITU-T: User Requirements Notation (URN) – Language definition, ITU-T Recommendation Z.151 (11/08). Geneva, Switzerland (2008); http://www.itu.int/rec/T-REC-Z.151/en
5. jUCMNav 4.4.0 (2011); http://softwareengineering.ca/jucmnav
6. Kealey, J., Amyot, D.: Enhanced Use Case Map Traversal Semantics. SDL 2007: Design for Dependable Systems. Springer, LNCS vol. 4745:133-149 (2007)
7. Mussbacher, G.: Aspect-oriented User Requirements Notation. PhD thesis, School of Information Technology and Engineering, University of Ottawa, Canada (2010)
8. Pourshahid, A., Chen, P., Amyot, D., Forster, A.J., Ghanavati, S., Peyton, L., Weiss, M.: Business Process Management with the User Requirements Notation. Electronic Commerce Research, 9(4), Springer, 269-316 (December 2009)
9. Roy, J.-F., Kealey, J., Amyot, D.: Towards Integrated Tool Support for the User Requirements Notation. SAM 2006: Language Profiles. Springer, LNCS 4320:183-197 (2006)

# iStarTool: Modeling requirements using the i* framework

Átila Malta[1], Monique Soares[1], Emanuel Santos[1], Josias Paes[1], Fernanda Alencar[2], Jaelson Castro[1]

[1] Universidade Federal de Pernambuco – UFPE, Centro de Informática, Recife, Brazil
{avmm, mcs4, ebs , jpsj2, jbc}@cin.ufpe.br
[2] Universidade Federal de Pernambuco - UFPE, Departamento de Eletrônica e Sistemas, Recife, Brazil,
fernandaalenc@gmail.com

**Abstract.** The iStarTool supports the graphical modeling of i* Framework. With a view to decrease the learning curve of i* models as well as to improve their quality, we provided the syntax checking feature. The tool allows the construction of valid models according to constraints and good practices guidelines. It is been developed using the open-source Eclipse platform and model-driven technologies, such as the Graphical Modeling Framework (GMF).

**Keywords:** iStarTool; Modeling tool, i* (iStar) Framework

## 1 Introduction

IStarTool is a graphical editor for the i* Framework models, built using the GMF framework. It supports the creation of both the SD and SR models, through an intuitive user interface built over the well-known Eclipse platform.

**Download Information/availability.** The iStarTool is free for download, and it is available at the project Web page: http://portal.cin.ufpe.br/ler/Projects/IStarTool.aspx.

**Web page and documentation.** The web page of the project is the same of the download page. The main documentation available in [1], is written in Portuguese but currently under translation to English.

**Main purpose.** The main purpose of the iStarTool is to facilitate the learning of i* language and improve the quality of i* models, being especially aimed at beginners. In several occasions during the teaching of i*, we identified the need to have a tool to help beginner users to reduce their number of mistakes well as to make it clear what kind of constructions were possible (or not). Several tools can handle some level of constraints over the i* models. In some cases also allowing, if necessary, the definition of extra modeling constraints, e.g., jUCMNav. However, we preferred to develop our own solution instead of customizing somebody else tool. The reasons are many fold. It served as an exercise to master the meta-modelling technologies and implementation environment. Furthermore, it allowed us to improve its usability, i.e., as we managed to provide more friendly warnings. In the future it will also incorporate model-driven features (see future works).

**Status and Maturity.** The current version of the iStarTool (Fig. 1) is 0.3. Some bugs still need to be fixed before the release of version 1.0. However, at this early stage, the tool presents a good degree of maturity and stability.

## 2 iStarTool

The iStarTool supports two different versions of i* Framework: the Yu's PhD Thesis and the iStarWiki [2]. Fig. 1 shows a screenshot of the tool interface. Panel 1 is where we can create and edit the diagrams. Panel 2 is the tool palette that offers the elements and links used when drawing diagrams. Panel 3 shows an overview of the diagram that allows the manipulation of large and complex diagrams. Panel 4 enabled the change of the properties of a selected element. Panel 5 highlights the button used to check (syntactically) the model.

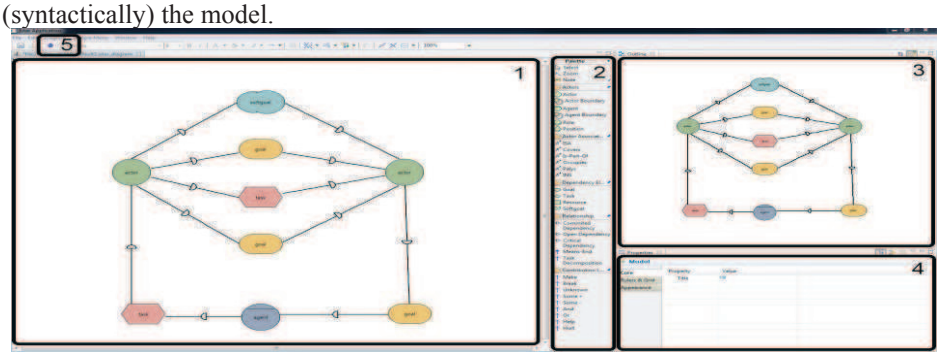



**Fig.1.** Screenshot of iStarTool

**Primary Features.** One of the main features of the iStarTool is the syntax checker. Due to the fact that we have used GMF, which supports the Object Constraint Language (OCL), it was possible to handle many common errors and constraints present in the i* models. This syntax checker feature consists of two independent systems integrated to the iStarTool, namely the the Syntax Warnings and the Syntax Checker System, described below.

The Syntax Warnings is an online system that enable pop-ups that fires warnings when the modeler makes a mistake during the modeling process. Instead of forbidding errors in the i* models, we prefer to alert users when some error is detected. For example, if a link is used in the wrong place, the Syntax Warning System shows a pop-up with an explanation, indicating to the user what is inappropriate. Then the user can correct it and learn in this process of trial and errors.

This system was deveopled because i* language allows some conditions that are difficult to be expressed and handled using OCL. Our systems can detect some of the most common problems and adequately treat them without upsetting the user. For example, if the user tries to link two actors without determining the intentional element an alert is fired. Many common mistakes are documented in the literature, see for example the i* good practices guidelines available at [2]. The errors detected include:

- Actors linked by dependency link without an intentional element (SD Model);

- Dependency links used inside an actor boundary (SR Model) linking internal elements;
- Contribution, Means-end or decomposition links used between actors in SD Models;

Unlike the Syntax Warning System, the Syntax Checker runs offline. The reason it that some i* steps could be considered wrong if analyzed in real time. In order to execute it, the user only needs to click a button (see Fig 1, panel 5). The Syntax Checker detects the same problems of the Syntax Warning and many others such as:

- The same intentional element being targeted more than once;
- The same intentional element being the source more than once;
- Dependency links without an intentional element in a SR model.

The iStarTool also allows users to work on multiple models at the same time. It saves all models in XML format, by default, and if necessary, they can be exported as an image. In a previous work we have identified the common and variable modeling elements of several i* based languages [4]. Based on SPL principles, we produced a core metamodel, which enabled the definition of specific i* dialects. Moreover, the current features of the iStarTool can be reused to support families of graphical editors for i* based languages. Since the AGILE approach was based on the iStarTool, similar families of tools could be generated in minutes.

## 3 Limitations and Future plans

We are planning to make several extensions. For example we want to support the measurement of i* models (based on some well pre-defined metrics) as well as to provide some semantic checkers present. We plan to integrate model transformations to produce other artifacts such as architectural models and scenario descriptions [5].

We intend to improve the interface of the iStarTool as well as the shape of the i* elements. Moreover, we wish to provide the interoperability with other i* modeling tools. Currently, the iStarTool just runs on Windows. However we are already working to support other operational systems.

## References

1. Santos, B. S. IStarTool – A proposal of tool for modeling i*. (in portuguese, Uma proposta de ferramenta para modelagem de i*). Master's thesis - Universidade Federal de Pernambuco, Centro de Informática, Brasil, 2008.
2. i* Wiki: i* Guides < http://istar.rwth-aachen.de/tiki-index.php?page_ref_id=200 >. Last Access in May of 2011.
3. Graphical Modelling Framework < http://www.eclipse.org/modeling/gmp/ >. Last Access in May of 2011.
4. Paes, J., Lima, , Santos, E., Silva, C., and Castro, J.: AGILE : Automatic Generation of i * Languages. In: Proceedings of the 24th Ibero-American Conference on Software Engineering - CIbSE 2011, Rio de Janeiro, Brasil: 2011.
5. Lucena, M., Silva, C., Santos, E., Alencar, F., and Castro, J.: Applying Transformation Rules to Improve i* Models. In: Proceedings of the 21st International Conference on Software Engineering and Knowledge Engineering (SEKE 2009), Boston, USA: 2009, pp. 43-48.

# Tool Interoperability using iStarML

Carlos Cares[1,2], Xavier Franch[2], Daniel Colomer[2], Lidia López[2]

[1] Universidad de La Frontera, Av. Francisco Salazar 01145, 4811230, Temuco, Chile,
[2] Universitat Politècnica de Catalunya, c/ Jordi Girona 1-3, 08034, Barcelona, Spain,
{ccares, franch, dcolomer}@essi.upc.edu, llopez@lsi.upc.edu
http://www.essi.upc.edu/~gessi/

**Abstract.** iStarML is an XML-based format for enabling interoperability among *i** tools. Its main design focus was to support data interchange even when involved tools implement different *i** variants. In this paper, we present a summary of the format, we briefly describe the `ccistarml` Java library, and we show an application of it. We finally summarize the requirements for representing new *i** concepts in order to generate a revised version of iStarML.

**Keywords:** *i** Framework, iStar, iStarML, interoperability.

## 1 iStarML 1.0: Basics and Structure

As an effect of the past and even current proliferation of different *i** variants, interoperability has become a non-functional requirement hard to accomplish by *i** tools. iStarML [1] is an XML-based proposal that has been conceived to deal with the existence of different *i** variants. It follows a concentric ring structure (see Fig. 1) having a rigid centre and a flexible periphery. Core *i** concepts are in the rigid part of the internal ring whilst flexibility is added going to the periphery up to 4 rings: actor and intentional elements are core concepts; types of core concepts (e.g., softgoal, belief) are in the second ring; particular values for decompositions are in third ring; strong variations of core concepts, e.g. "norm" as intentional element is in the fourth ring.

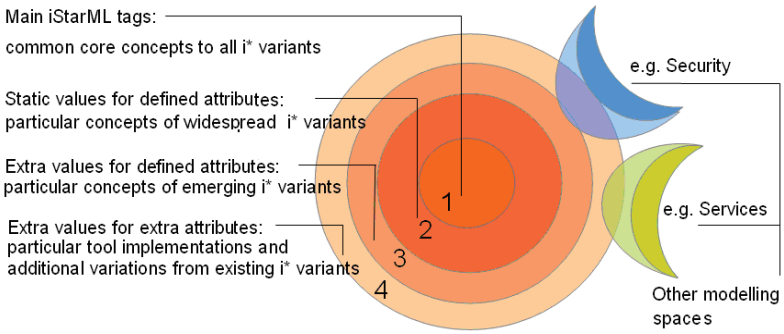



**Fig. 1.** Concentric ring structure of iStarML.

## 2 iStarML Computational Support

In [2] we offer a set of basic XML specifications. First, we provide an iStarML specification using an XSD definition. This specification is the most used specification types for XML files. A clear advantage is that XML processors can parse XML files using XSD definitions which means that iStarML parsers can be implemented taking public XML processors and this XSD definition. However, this definition has some limitations derived from the XSD grammar, e.g., a dependency between attributes cannot be represented on XSD specifications. Therefore, although less popular but more accurate, we have provided a Schematron [3] set of rules for iStarML syntax verification.

In addition to these parsing proposals, we have developed a Java package for handling iStarML files. We have called it `ccistarml` package. The several classes inside this package allow creating, reading and modifying iStarML files. Each of these classes is explained and illustrated using simple examples in the `ccistarml` tutorial [4]. The code in Fig. 2 illustrates how an iStarML file can be loaded, parsed, modified by adding a new actor, and saved.

```
ccistarmlFile f = new ccistarmlFile();
f.loadFile("sample01.istarml.xml");
f.istarmlParser();
if (!f.hasErrors()) {
   ccistarmlContent content = f.mainTagStructure();
   content.add_actor("Tutor");
   f.saveFile();
}
```

**Fig. 2.** Sample code for managing iStarML representations of *i** diagrams.

## 3 iStarML Tests and Scenarios of Use

Since iStarML was proposed, we have developed different interoperability scenarios and applications: (1) storing diagrams in the H*i*ME hierarchical *i** tool [5]; (2) exporting and importing iStarML in the jUCMNav GRL tool including interoperability proofs [6]; (3) translating files from OME3 tool to iStarML file by using a Java applet [7]. About taking advantage of XML representation of iStarML we have (4) exemplified the use of XPath for metric calculations [1]; (5) used XSLT for transforming iStarML files into Prolog clauses [8]; and (6) used XQuery for quality assurance of *i** models [9]. We also mention the use of iStarML as the basis for formulating a supermetamodel coordinating model interoperability in a semantic-aware scenario [10].

## 4 Conclusions and Future Work

In this paper we have presented iStarML 1.0 as an interoperability facility for *i** tools. Our proposal considers the existence of *i** variants and hence, a polysemantic scenario. We have shown its structure, some computational support and different

scenarios illustrating both the feasibility of reaching interoperability and the advantages of having a XML representation for *i** models.

It is our position that iStarML is a solid proposal, but its dissemination and usage depends on including most of the current and future representational requirements of the *i** community, which can be reached only by constituting a wide working group motivated to generate iStarML 2.0.

Next steps on iStarML include interoperability tests of the two existing possibilities of graphic representations: iStarML's graphic elements and the nested structures proposed in iStarML for including SVG tags [11]. Here for example we have the challenge of handling the graphical proposal for representing inherited intentional elements from [12]. Finally, we are also planning to improve the current representational capabilities of iStarML extending it to handle concepts such as modules [13] and similar tags enabling Computer Supporting Collaborative Engineering.

## Acknowledgments

This work has been supported by the Spanish project TIN2010-19130-c02-01.

## References

1. Cares, C., Franch, X., Perini, A., Susi, A.: Towards Interoperability of *i** Models Using iStarML. *Computer Standards & Interfaces*, 33(1), 69-79 (2011)
2. iStarML site, http://www.essi.upc.edu/~gessi/iStarML
3. Jelliffe,R.: The Schematron Assertion Language 1.6, http://xml.ascc.net/resource/schematron/Schematron2000.html (2002)
4. Cares, C.: ccistarml: A Java Package for Handling iStarML files, http://www.essi.upc.edu/~ccares/papers/ccistarml_v0.6.pdf
5. HiME site, http://www.upc.edu/gessi/HIME/
6. Colomer, D., López, L., Cares, C., Franch, X.: Model Interchange and Tool Interoperability in the *i** Framework: A Proof of Concept. In *Proc. of the 14th Workshop on Requirements Engineering* (WER11), 27-29 April, Rio de Janeiro, Brasil, pp.369-381 (2011)
7. OME3 to iStarML Converter, http://www.essi.upc.edu/~gessi/iStarML/ometoistarml-remoto/online_tools.html
8. Cares, C., Franch, X.: 3MSF: A Framework to Select Mobile Office Devices. *Int. Journal of Computer Science and Applications*, 6(5), 121-144 (2009)
9. Cares, C., Franch, X.: Towards a Framework for Improving Goal-Oriented Requirements Models Quality. In *Proc. of the 12th Workshop on Requirements Engineering* (WER09), July 16-17, Valparaiso, Chile. pp. 3-14 (2009)
10. Cares, C. and Franch, X.: A Metamodelling Approach for i* Model Translations. In Proc. of *the 23rd Int. Conference on Advanced Information Systems Engineering* (CAiSE), June 20-24, London, United Kingdom, pp. 337-351 (2011).
11. Scalable Vector Graphics (SVG) site, http://www.w3.org/Graphics/SVG/
12. López, L., Franch, X., Marco, J.: Defining Inheritance in i* at the Level of SR Intentional Elements. In *Proc. of the 3rd International i* Workshop* (iSTAR08), February 11-12, Recife, Brazil, pp. 71-74 (2008).
13. Franch, X.: Incorporating Modules into the i* Framework, LNCS, 6051, pp 439-545 (2007)

# Adding Functionality to *openOME* for Everyone

Ralf Laue, Arian Storch

Chair of Applied Telematics / e-Business, University of Leipzig, Germany
laue@ebus.informatik.uni-leipzig.de

**Abstract.** Adding new functionality to graphical editors like *openOME* usually requires to become familiar with the programming environment of the underlying framework.
We present an interface for Eclipse EMF/GMF-based modelling tools that allows to add new functionality very quickly - without the need to be familiar with Eclipse development.

## 1 Primary Features

The Eclipse Modeling Framework (EMF) and the Eclipse GMF (Graphical Modeling Framework) are well-established frameworks for developing modelling tools. An example for an EMF/GMF-based $i^*$ modelling tool is *openOME* [1].

We know from our own experience that often a lot of knowledge is required before someone can actually start to add functionality to EMF/GMF-based editors. The aim of our Eclipse plugins - called Eclipse Modeling Toolbox - is to make the task of extending editors like *openOME* as easy as possible. In this section, we present the possibilities offered by our plugins.

### 1.1 User-Defined Attributes

An additional view for user-defined attributes allows to add own attributes to model elements or to a model as a whole. Fig. 1 shows how user defined attributes have been added to a task in an $i^*$ model. If a URL referring to a local or remote file is used as an attribute value, this file can be opened by clicking on the URL. This way, it is possible to associate additional files to a model element (for example business process descriptions that are related to a task).

### 1.2 Export and Import

By using the Eclipse Modeling Toolbox, it is possible to create export and import functionality for other file formats. This export and import is done by means of XSLT transformations. So far, we have created transformations for exporting an *openOME* model to a set of Prolog facts. We are currently working on an export to the interchange format iStarML [2].

A new format can be supported by simply adding a new XSLT file and inserting information about the export/import format to a configuration file.

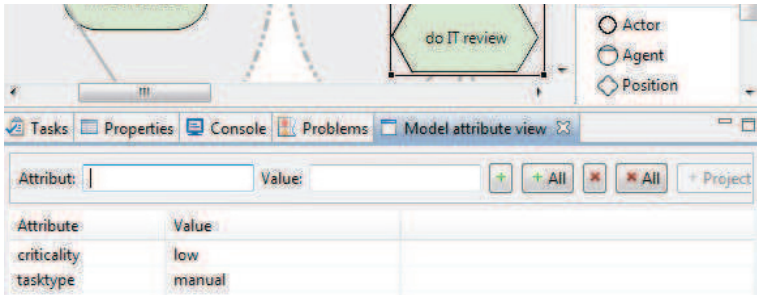



**Fig. 1.** User-defined attributes added to a task

### 1.3 Integrating Add-ons

We tried to make the integration of third-party programs into *openOME* as easy as possible. We know from our own experience that often a lot of knowledge is necessary before someone can actually do such integration. At least, the answers to the following questions have to be known:

- How can we access the model data and transform them into the data format expected by the third-party program?
- How can we start the third-party program from within the modeling tool?
- How can we transfer the answers given by the third-party program back into the user interface of the modeling tool?

With our plugins , we provide easy-to-understand interfaces for dealing with the above questions. The already mentioned export scripts can be used for accessing and transforming the model (including its user-defined attributes as described in Sect. 1.2).

The information on how to start the external program can be added to a configuration table at runtime, either manually or by importing an XML file containing the necessary information. This creates a new menu item from which the external program can be started.

Finally, we have to make sure that the results computed by the third-party tool are transferred back into the *openOME* user interface. For this purpose, we provide several interfaces. They abstract away Eclipse implementation details and allow the external program

- to print information into the Eclipse console view,
- to add information about an error, warning or information to the Eclipse problem view
- to add a visual marker to the graphical model element,
- to add, delete or change attributes of the modelling elements (which includes existing attributes such as “name”, graphical attributes such as “element size” and user-defined attributes)

With the described features, it is possible to integrate new functionality into *openOME* without having to learn about Eclipse development. In the most cases, new features can be added even without having to compile the sources.

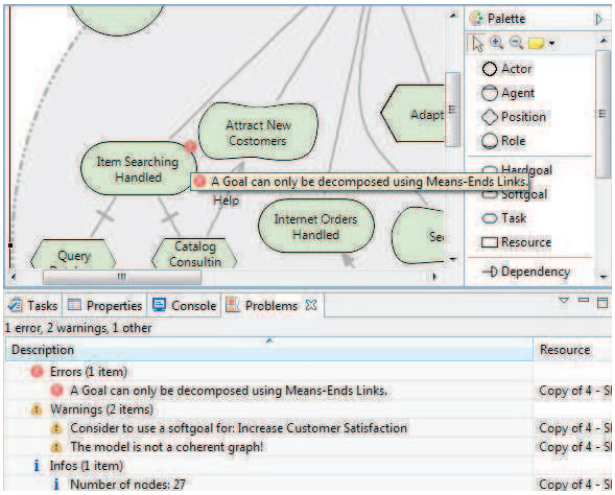



**Fig. 2.** Results from the add-on "Extended Model Validation" in *openOME*

### 1.4 Validation

As an example for a useful *openOME* add-on, we have developed an add-on called "Extended Model Validation". It exports the information that is contained in the model into a set of Prolog facts. Afterwards, SWI-Prolog is called for locating modelling problems such as syntactical errors (like "dependency link without dependum"), layout problems and (to some extent) problems with the labels (like labels including the phrase "...to be..." within a task instead of a goal). More information about the model validation approach can be found in [3].

## 2 Status and Future Plans

It is important to mention that the Eclipse Modeling Toolbox can be integrated within any EMF/GMF-based modelling tool. It has been used successfully within the business process modelling tool *bflow** for the development of some useful add-ons. Future plans related to *openOME* include to provide iStarML import and to add more functinality to the add-on-mechanism. Everyone is invited to use and improve the Eclipse Modeling Toolbox which is available at `http://sourceforge.net/projects/eclipsemodeling/`.

## References

1. www.cs.toronto.edu/km/openome/: (Openome, an requirements engineering tool)
2. Cares, C., Franch, X., Perini, A., Susi, A.: Towards interoperability of i* models using iStarML. Computer Standards & Interfaces **33** (2011) 69 – 79
3. Laue, R., Storch, A.: A flexible approach for validating i* models. In: Proceedings of the 5th International i* Workshop, Trento, Italy. (2011)

# Tropos modeling, code generation and testing with the Taom4E tool

Mirko Morandini, Cu Duy Nguyen, Loris Penserini, Anna Perini, and Angelo Susi

FBK-CIT, Trento, Italy,
{morandini,cunduy,penserini,perini,susi}@fbk.eu,

**Abstract.** We present Taom4E, a tool for the development of software following the agent-oriented software engineering methodology Tropos. The Eclipse plug-in Taom4E supports visual goal modelling and includes functionalities for code generation and testing for goal-oriented agent platforms.

**Key words:** Goal-oriented development, modelling tools, code generation.

## Introduction

The agent-oriented software engineering methodology Tropos, which roots in *i** for requirements modelling, becomes increasingly popular. Modelling tools are essential for such a methodology. We present Taom4E[1] (Tool for Agent Oriented visual Modelling for the Eclipse platform) [1], a tool which supports goal-oriented modelling, code generation and testing for goal-directed systems, following the Tropos methodology and its extension Tropos4AS. Taom4E supports the Tropos modelling activities for early and late requirements analysis and architectural design and contains extensions for an automated agent-oriented implementation and testing.

## The Taom4E Tool

The Taom4E tool provides a graphical model editor, based on the Tropos metamodel, as defined in [2], and allows various views on this model. Combining Tropos actor- and goal diagrams, representing strategic dependency (SD) and strategic rationale (SR) diagrams in *i**, the *Mixed Diagram* is the main graphical representation in Taom4E, throughout the supported development phases. In the Tropos early and late requirements analysis phases, actor diagrams, displaying the dependencies between actors, can be graphically created. Actors are

[1] Taom4E is developed by the Software Engineering unit at Fondazione Bruno Kessler (FBK), Trento. The current version 0.6.3 is downloadable under GPL license from the tool homepage http://selab.fbk.eu/taom.

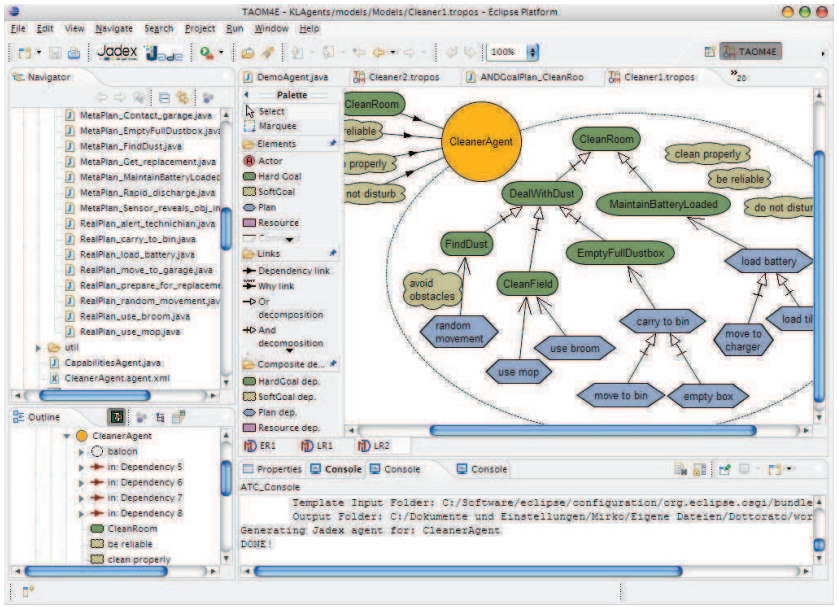



**Fig. 1.** Interface of the TAOM4E Eclipse plug-in.

detailed in a goal diagram, which is graphically shown in a "balloon" associated to the actor. In this balloon, delegated goals are visualized and can be decomposed to more detailed goals, operationalized by plans or delegated to other actors. An actor representing the system-to-be can be furthermore detailed to a multi-agent system, in an *architectural design diagram*. These development phases provide different views on a single TAOM4E model, thus ensuring traceability, but not yet preserving the model history. Figure 1 shows the principal view of the tool front-end. Models are edited according to the Tropos graphical notation, provided by the *Palette* window. Tabs are used to switch between model views of the different phases (ER, LR,...). Visualized diagrams are views on the whole model, whose components are displayed in the *Outline* window. In the *Navigator* in the left-hand side window, the folders with the output of the ***t2x*** code generation tool are actually shown, for each actor in the system.

### Architecture and Extensions

Various extensions to the TAOM4E model editor are provided. The ***t2x*** code generation tool maps goal models to a goal-directed implementation on the Jadex BDI agent platform, explicitly preserving goal models at run-time and providing the proper middleware for navigating this model and acting according to it. Agent code can be generated from the graphical interface, and the implemented prototypes are executable directly from the Eclipse user interface. Moreover, extensions for environment and condition modelling according to the Tropos4AS

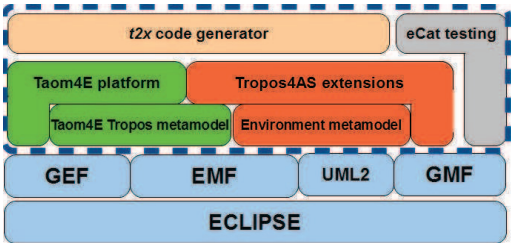



**Fig. 2.** Tool architecture: Taom4E and its extensions.

framework [3], are available. Figure 2 summarizes the architecture of the toolset, which consists of various plug-ins for the Eclipse development platform, built on the Eclipse EMF and GEF frameworks for model-driven development. They are installable through the Eclipse installation manager by adding the link provided on the tool homepage. The models are saved in XMI format and can be accessed by other applications through an EMF-based interface.

Goal-Oriented testing has been proposed as a complementary activity to goal-oriented modelling and code generation, with the aim to support testing and validation along the process phases [4]. The *eCAT* tool derives test cases directly from Tropos goal models and uses them to test implemented agents, using FIPA-standard messaging services.

## Conclusion

The Taom4E tool was used in several projects that involved Tropos, and in various university courses on requirements engineering and on goal-oriented development. As a design limitation, the tool can only handle a single model per file, therefore changes to one view of the model are projected to the whole model. Moreover, no automatic diagram repositioning is implemented. For addressing this and other issues, future work concerns the reimplementation of the graphical front-end with state-of-the-art technology. Also, we are working on the extendibility of the graphical model editor with additional concepts.

## References

1. Morandini, M., Nguyen, D.C., Perini, A., Siena, A., Susi, A.: Tool-supported development with tropos: The conference management system case study. In: Agent Oriented Software Engineering VIII. Volume 4951 of LNCS. (2008) 182–196
2. Susi, A., Perini, A., Giorgini, P., Mylopoulos, J.: The Tropos Metamodel and its Use. Informatica (Slovenia) **29**(4) (2005) 401–408
3. Morandini, M., Penserini, L., Perini, A.: Towards goal-oriented development of self-adaptive systems. In: SEAMS '08: Workshop on Software engineering for adaptive and self-managing systems, ACM (2008) 9–16
4. Nguyen, D.C., Perini, A., Tonella, P.: ecat: a tool for automating test cases generation and execution in testing multi-agent systems. In: AAMAS. (2008) 1669–1670

# Analysing a repository of design knowledge with the GO-DKL browser

Andrew Hilts and Eric Yu

Faculty of Information
University of Toronto
140 St. George Street, Toronto, ON
andrew.hilts@utoronto.ca
eric.yu@utoronto.ca

**Abstract.** This paper presents the GO-DKL browser, a web-based interface designed to support the contextualization and analysis of items within *Goal-oriented Design Knowledge Libraries*. It is still at a very early stage of development, though a small sample of information systems designers evaluated a demonstration of the tool and provided generally positive feedback.

**Keywords:** Design Knowledge, User Interfaces, Knowledge reuse, Open OME

## 1 Purpose of the GO-DKL browser

We define a *goal-oriented design knowledge library* as a repository of relational data linking design features with stakeholder objectives. These relationships, as in *i** and related approaches, indicate the contribution type that a child (design feature / task) passes on to a parent (soft goal). Such a repository might be populated with codified empirical data gathered from a review of a domain's literature, as in [1]. The repository database can be queried to retrieve all design features that have an impact on a focal goal, or on all goals in the database, among other possibilities.

The GO-DKL browser is intended to enable system designers to associate situational goals they have elicited and defined with library items, and to subsequently analyze the impact of retrieved design alternatives on those goals. This association is an act of contextualizing the library items; designers can select a subset of knowledge base records that they feel are related to their own project. From there, the system can retrieve design features that contribute to the selected goals.

The GO-DKL browser is intended to facilitate a focused analysis of certain relationships within a goal graph, in an attempt to mitigate the daunting task of assessing complex goal graphs [4]. Specifically, designers can select focal goals, and drill down to reveal their contributing factors (eg: sub goals or design features). Analysts can assess the effects of not including certain design features on high-level goals, or modify the library's relationships based on trusted contextual knowledge.

## 2 Primary Features

Essentially, the GO-DKL browser is a web-based interface to a MYSQL database. This interface is structured to support the analytic processes discussed above, through goal association, browsing, selection, and model exploration and reconfiguration phases.

The model exploration and reconfiguration phases are centered around a tree-list interface generated using JavaScript. The tree-list structure allows relationships to be expanded and collapsed at will so that certain components can be isolated. HTML form elements permit the reconfiguration of contribution relationships. Parent goals are automatically coloured based on their received contributions from child elements, using simplified *i** evaluation concepts, based largely on the propagation rules defined in [2].

The list is populated by the designer's selected goals and any other knowledge base items that contribute to those goals (his or her 'project model'). The actual list presentation is generated by a script that converts tabular records from the knowledge base into hierarchical HTML elements, with high-level goals as parent elements, and design features as child elements[1]. Often the same design feature or relationship will appear numerous times throughout the model, in which case the interface treats all instances of a it as one data point.

Therefore, if one instance of a design feature is deselected, all other instances of that feature will be deselected throughout the tree-list. In this manner, if an analyst selects a design feature because of its contribution to a focal goal, and similarly deselects another, he or she may then examine other goals to see the effects of those choice in other parts of the model.

## 3 Limitations, Evaluation and Future Work

While the tree-list view can help an analyst to focus intently on the contributions to a single goal (which can be difficult to do with complex models), the 'bigger picture' may be harder to assess. To mitigate this limitation, each high-level goal in the tree-list features an option to export a *model slice*[3], which retrieves all design features contributing to that goal, *in addition to* all other goals affected by those design features. In this manner, an analyst may examine the larger context of design decisions within his or her project model. The model slice is dynamically exported as a *Q7 file*, which is interoperable with Open OME[2].

Several system design practitioners were shown a demonstration of the GO-DKL browser. They generally deemed it to be a valuable way of retrieving and informally assessing the findings of related projects in comparison to their own endeavours, though several usability critiques emerged. Future work would address these criticisms, as well as attempt to better interoperate with other goal modeling tools.

[1] A maximum of 5 levels of decomposition is currently supported.
[2] Importing Q7 files is currently unsupported.

## 4 Availability

Source code[3] and a high-level description[4] are currently available; the community is invited to contribute and expand on this work.

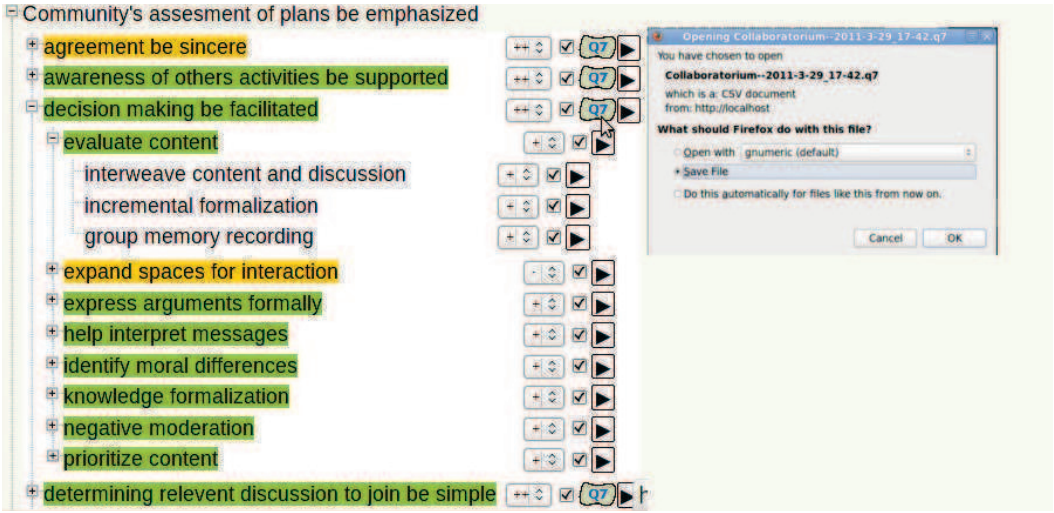



**Fig. 1.** A 'project model' of selected relationships may be browsed and reconfigured through GO-DKL browser's tree-list interface. Model slices for each high-level goal may be downloaded and imported into Open OME for further analysis.

## References

1. Esfahani, H., Yu, E.: A Repository of Agile Method Fragments. In: Münch, J., Yang, Y., Schäfer, W. (eds.) LNCS 6195: New Modeling Concepts for Todays Software Processes. pp. 163–174. Springer, Berlin / Heidelberg (2010)
2. Horkoff, J., Yu, E.: A Qualitative , Interactive Evaluation Procedure for Goal- and Agent-Oriented Models. In: CAISE Forum. CEUR-WS.org, vol. 453. pp. 19–24 (2009)
3. Leica, M.F.: Scalability concepts for i* modelling and analysis. Master's thesis, University of Toronto (2005)
4. Pastor, O., Estrada, H., Martínez, A.: Strengths and Weaknesses of the i* framework. In: Yu, E., Giorgini, P., Maiden, N., Mylopoulos, J. (eds.) Social Modeling for Requirements Engineering. pp. 607–643. MIT Press, Cambridge, MA (2011)

[3] https://github.com/andrewhilts/Go-DKL also features a detailed README
[4] http://www.designknowledge.org

# *i** Modules: a jUCMNav Implementation[†]

Daniel Colomer, Xavier Franch

Software Engineering for Information Systems Research Group (GESSI)
Universitat Politècnica de Catalunya (UPC)
c/ Jordi Girona 1-3, 08034, Barcelona, Spain
{dcolomer, franch}@essi.upc.edu
http://www.essi.upc.edu/~gessi

**Abstract.** When building large-scale goal-oriented models using the *i** framework, the problem of scalability arises. Modules have been proposed to structure *i** models into reusable and combinable fragments. In this work we present an implementation of the module concept over the jUCMNav tool.

**Keywords:** *i**, iStar, modules, jUCMNav.

## 1 Introduction

One research challenge for the *i** community is to make *i** models more manageable and scalable. In [1] we defined a theoretical approach for adding modularity facilities to the *i** metamodel in a loosely coupled way, also tailored to a particular domain, namely the modularization of goal models for data warehouse schemata [2]. In this work, we present an implementation of the general concept of module as an extension of the jUCMNav 4.2.1 plug-in. The tool may be downloaded from http://www.essi.upc.edu/~gessi/mod_extension/resources.html where a basic tutorial in the form of user's manual may be found, as well as details on the metamodel used.

jUCMNav is a graphical editor and an analysis and transformation tool for the *User Requirements Notation* (URN). URN is intended for the elicitation, analysis, specification, and validation of requirements. It combines modeling concepts and notations for *goals and intentions* (with GRL) and *scenarios* (with UCM). We will focus on the GRL notation because of its *i**-based nature. It is a graphical language for supporting goal-oriented modelling and reasoning about requirements, especially non-functional requirements and quality attributes. It provides constructs for expressing various types of concepts that appear during the requirement process. GRL has its roots in two widespread goal-oriented modeling languages: *i** and the NFR Framework. Major benefits of GRL over other popular notations include its integration with a scenario notation and a clear separation of model elements from their graphical representation, enabling a scalable and consistent representation of multiple views/diagrams of the same goal model.

[†] This work has been partially supported by the Spanish project TIN2010-19130-C02-01.

## 2 Module Implementation

We extended the last jUCMNav metamodel available (`URN_23.mdl`), see Fig. 1. In order to guarantee later graphical and usability efficiency we made some decisions that differ from the model presented in [1]. A `State` pattern was implemented in order to allow dynamic state (i.e., type) changes during module definition. Then a new attribute was added to the existing `IntentionalElement` definition representing the notion of root (for graphical purposes) so the relationship `root` introduced in [1] was no longer needed. Constraints such as multiplicities were assigned to integrity constraints due to modeling software limitations. The implemented structure also facilitates later extensions such as new module definitions.

Fig. 2 shows a snapshot of module in jUCMNav. In the left-hand side we may find module references. They have two different functionalities: to inform the user about the nature of the module that is currently being edited and about the different sources from which the current module was obtained (they are only shown if the module was obtained as a result of one or more module operations) for traceability purposes. This second type of references is shown in green background.

In [1], constraints are proposed for ensuring the structural correctness of the different types of modules. Both general and particular constrains over SR and SD Modules have been implemented as Static Semantics checking rules (see Fig. 3).

A crucial point of the approach in [1] is that of module operations. *Combination* and *Application* are somehow similar, so we decided to implement both of them as a single abstract operation. When this abstract operation is applied to an undefined module, *Module Application* will be executed and then a list of dependency matches is needed. When applied to any type of module (different from a undefined module) *Module Combination* will be executed. In this case a simple merge is carried out and the resulting module is created. Both operations were implemented as part of the set of Eclipse navigator view functionalities (see Fig. 4). A simple merge algorithm is used and so some limitations appear (see Section 3).

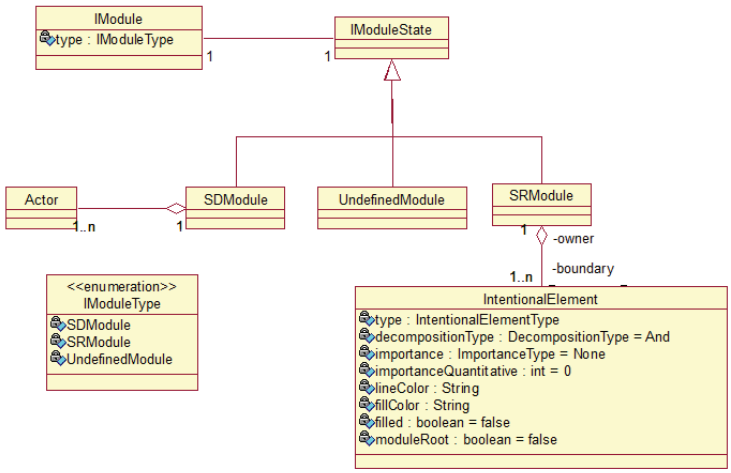



Figure 1. The metamodel part related to modules as implemented in the jUCMNav extension.

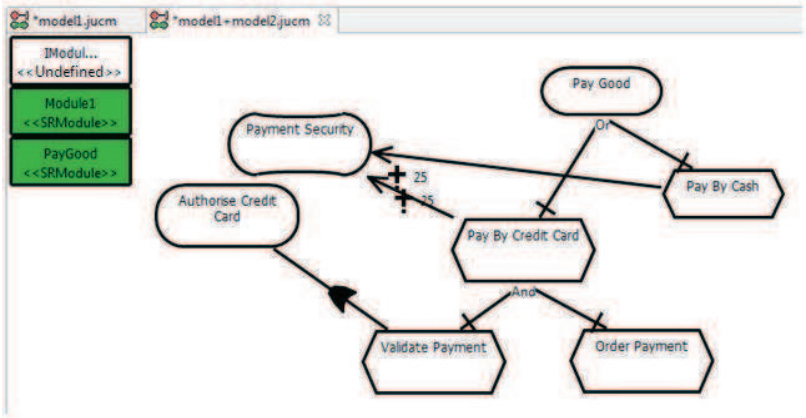



Figure 2. Module definition in jUCMNav extension.

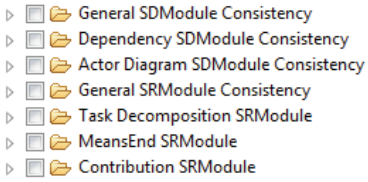



Figure 3. Static Semantics checking rules.

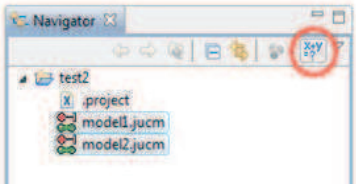



Figure 4. Module operations in Eclipse.

## 3 Limitations and Future Work

jUCMNav makes a clear separation of model elements from their graphical representation, enabling a consistent representation of multiple diagrams of the same goal model. This multiple-diagram representation is not covered in [1] and although the metamodel extension was made taking this into account, the current solution only supports files with a single diagram. Future work aims at solving this limitation.

Extensibility has been a goal. New module specializations can be easily added by extending the current implemented hierarchy. Functionalities for collapsing and expanding are yet to be implemented. Module operation constraints can also be easily added through the `ModuleCombinationAction` class. Last, there are two different ways of extending module restrictions: 1) jUCMNav offers the possibility to add, remove and edit current integrity constraints through Eclipse's preferences view; 2) new OCL constraint packages could be easily added to the plug-in by incorporating their XML description and extending the default integrity constraint loader.

## References

1. X. Franch: "Incorporating Modules into the *i** Framework". CAiSE 2010.
2. A. Maté, J. Trujillo, X. Franch. "A Modularization Proposal for Goal-Oriented Analysis of Data Warehouses using *i**". ER 2011.


**a CEUR Workshop Proceedings, ISSN 1613-0073**