iStar 2010 – Proceedings of the 4th International *i** Workshop

7-8th June, 2010

Hammamet, Tunisia

http://www.cin.ufpe.br/~istar10/

EDITORS:
Jaelson Castro
Xavier Franch
John Mylopoulos
Eric Yu

a CEUR Workshop Proceedings, ISSN 1613-0073, Volume 586

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), and Recife, Brazil (2008), the 4th International *i** Workshop is being held in Hammamet, Tunisia June 7-8, 2010. 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 22nd Conference for Advanced Information Systems Engineering (CAiSE 2010), benefiting from the common interests shared by the workshop and the conference. Following past workshops, the format is a small and informal gathering aimed at maximizing interaction. As initiated in the 2008 edition, participants submit papers in advance of the workshop. This year, however, the submitted papers are reviewed by at least two members of a programme committee, providing useful feedback for authors. Revised versions of the papers are included in these proceedings. We thank authors and reviewers for their valuable contributions. We also thank the organizers of the CAiSE conference for their support.

We look forward to lively conversations and debates with old and new friends at the workshop, in the refreshing surroundings of Hammamet, Tunisia!

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*.
Carlos Cares, *Universidad de la Frontera, Chile.*
Luiz Marcio Cysneiros, *York University, Canada*.
Hugo Estrada, *CENIDET, Mexico*.
Daniel Ghose, *University of Toronto, Canada*.
Paolo Giorgini, *University of Trento, Italy*.
Renata S.S. Guizzardi, *Federal University of 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 Universidade Catolica do Rio de Janeiro, Brazil*.
Yves Lespérance, *York University, Canada*.
Sotirios Liaskos, *York University, Canada*.
Lin Liu, *Tsinghua University, China.*
Lidia López, *Universitat Politècnica de Catalunya, Spain*.
Neil Maiden, *City University London. UK*.
Oscar Pastor, *Universidad Politécnica de Valencia, Spain*.
Anna Perini, *Fondazione Bruno Kessler, Italy*.
Pete Sawyer, *Lancaster University, UK*.
Dominik Schmitz, *RWTH Aachen University, Germany*.
Alberto Siena, *Fondazione Bruno Kessler, Italy*.
Angelo Susi, *Fondazione Bruno Kessler, Italy*.
Jelena Zdravkovic, *Stockholm University, Sweden*.

Proceedings Edition Support

David Ameller, *Universitat Politècnica de Catalunya, Spain*.
Lidia López, *Universitat Politècnica de Catalunya, Spain*.
Marc Oriol, *Universitat Politècnica de Catalunya, Spain*.

Website Support

Carlos Diego Quirino Lima, *Universidade Federal de Pernambuco, Brazil*.

Table of Contents

Short Papers

On difficulties of forming opinions on what you don't know that you don't know - in Information Systems Engineering

Arne Sølvberg

Norwegian University of Science and Technology, Trondheim, Norway

Abstract. Information systems engineering is increasingly dealing with non-routine problem solving. Information support systems must be built to adapt to continuous changes in the ways of the supported organizations. Problem solving organizations learn as they operate. As more knowledge is gained about a particular task the initial approach to the task may change. New subproblems are identified and new needs for information support will surface. This is in contrast to the more common routine processing of predefined tasks and the associated workflow design. Straightforward requirements engineering is not longer sufficient. The talk will discuss the phenomenon of uncovering the unknown in an information systems engineering setting

Improving the Modularity of *i** Models

Fernanda Alencar[1], Márcia Lucena[2,], Carla Silva[3],
Emanuel Santos[4], Jaelson Castro[4]

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

Abstract. i* offers expressive models to capture social and intentional characteristics of a system organizational context, and explicitly captures stakeholders' motivations and rationale in a requirements model. Thus, the more detailed i* models are, the more complex they become. Hence, i* models can become unnecessarily hard to read, understand, maintain and reuse. In the past years we have been investigating how to tame the complexity of the models, with a view to improve their modularity. This paper presents two of our strategies. The first one relies on aspect-orientation principles whereas the second one is based on model transformations.

Keywords: i*, modularization, Aspects, Model Transformations.

1 Introduction

Modularity measures the degree to which the modeling language offers well-defined building blocks for building model. Although i* incorporates a decomposition mechanism based on strategic actors, which could be used to improve modularization of i* models, the way in which this mechanism is used is often not suitable to produce models that are easy to maintain and reuse. Current modeling methods represent the rationale of an actor in a monolithic way [3],[6]. Besides, sometimes several refinements are described in a scattered and tangled form (also known as crosscutting), making it hard to visualize the boundaries of sub-graphs related to specific domains. This poor modularity compromise the management of the complexity of the models, an important pre-requisite for the adoption of i* in industrial settings [4]. In order to reduce the complexity of i* models and increase their modularity we proposed two strategies; the use of aspect oriented principles [1] or the adoption of a model transformation strategy [8]. The paper is organised as follows. Section 2 describes a strategy to improve the modularity of i* models using

aspect oriented principles. Section 3 presents an approach which relies on the definition of transformation rules to re-structure the models. Section 4 discusses results obtained and Section 5 points out ongoing and future research.

2 Modularizing i* with Aspects

The modularity of i* models can be improved by removing tangled and scattered information into aspectual actors together with some weaving mechanisms [1]. Our aspectual approach consists of (i) a set of guidelines to identify crosscutting concerns in i* models; and (ii) an extension of the i* modeling language [11] by adding aspectual constructors to modularize crosscutting concerns and to allow its graphical composition with other system modules (Fig 1).

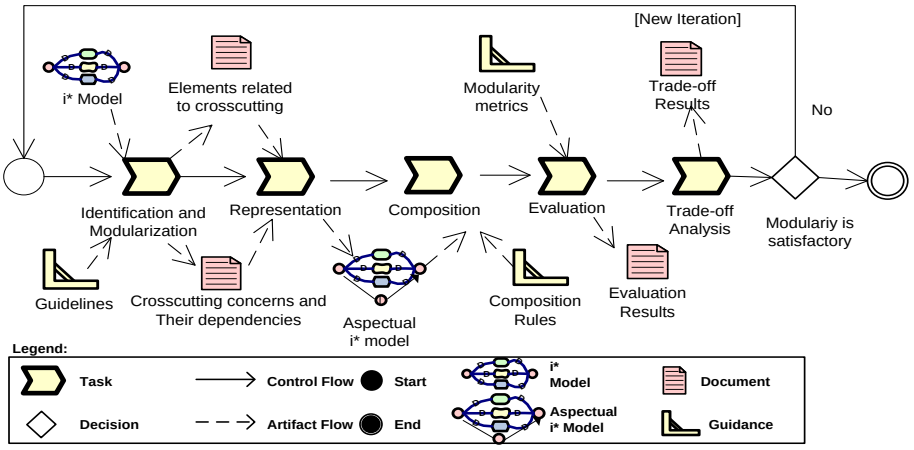

Fig. 1 – The modular i* with aspects strategy.

In this approach the crosscutting concerns are extracted into modules, called aspects, which are later composed back to the base model. Hence, we claim that Aspect-Oriented Software Development (AOSD) mechanisms [5] can contribute to increase the modularity of i* models. Four guidelines were proposed to deal with the identification, separation and modularity of the crosscutting concerns. Once identified, the crosscutting concerns are removed from the original actors, and placed in a new type of model element, the so called *Aspectual Element*. This element will have a specific graphical representation. Later it will be composed (weaved) with an actor or another aspectual element using a *Crosscut Relationship*. This relationship specifies how an i* element, located inside an aspectual element, is related to another i* element located inside an actor or another aspectual element. The composition step can be performed by graphical transformations. The evaluation of the resulting i* models is based on a suite of metrics adapted from the literature. Finally, a trade-off analysis will be performed and if the results are not appropriate (modularity is still poor) a new interaction may be executed.

3 Modularizing i* by means of Model Transformations

Another approach to improve the modularity of i* models is to restructure the models in order to extract the information that are not fully related to the application domain. To balance the responsibilities of the system actor, this information could be delegated to new system actors. Hence, we could transform the original model into a more modularized one.

Our model transformation approach consists of three activities (Fig 2): (i) Analyze Internal Elements, where Internal Elements can be factored out from software actor are identified; (ii) Apply Transformation Rules, which relies on model transformation rules to systematically move (delegate) the identified internal elements from software actor to new actors; (iii) Evaluate i* Models, used to evaluate the modularization of the models. The process is semi-automatic since the activities (ii) and (iii) can be automated, while the analysis of internal elements activity (i) depends on requirements engineers and domain experts. In this case, it is necessary to use: (i) heuristics to guide the decomposition of the software actor; (ii) a set of rules to transform i* models in modular i* models; (iii) metrics to evaluate the degree of modularity of both initial and final models. Further details can be found in [8].

Some measurement is required to check the improvement of the modularity. If the modularization still is inappropriate, new iterations may be necessary. These modular i* models are used as the starting point to generate architectural descriptions from requirements models [9].

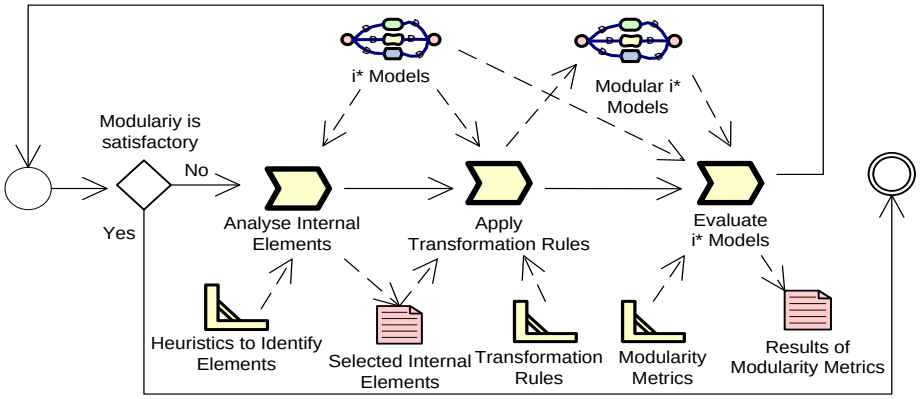

Fig. 2 – The modular i* with model transformation.

In order to illustrate the techniques used in this work, we review the *Media Shop* example [3]. *Media Shop* is a store that sells and ships different kinds of media items. To increase market share, *Media Shop* has decided to use the *Medi@* system, a business to customer retail sales front-end on the Internet.

Often i* models are overloaded with information capturing features of both the system organizational environment and the software system itself. However, the more detailed i* models are, the more complex they become (Fig. 3). This rich ontology aligned with the common misuse of the decomposition mechanisms provided by the i*, can head to models unnecessarily hard to read, understand, maintain and reuse.

The proposed approach allows delegation of different issues of a problem, initially concentrated into a single system actor, to new actors, which allows dealing with each actor separately. Details on an earlier version of this activity can be found in [8]. We have added a new rule, to deal with a special situation that may arise when independent sub-graphs, i.e., sub-graphs from different domains, have the same root goal. These sub-graphs are alternatives to satisfy this root goal. In this case, an actor is created for each alternative. Later, each of them will be considered as a different architectural solution.

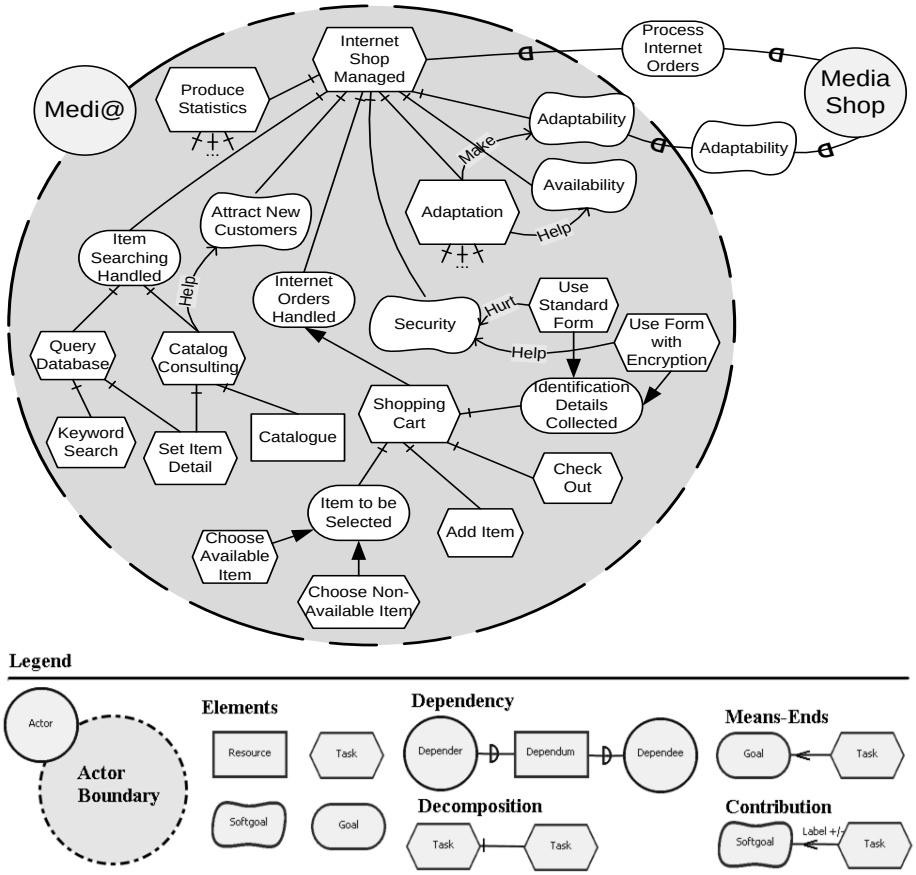

Fig. 3 – SR model for Medi@ system.

After carrying out the *Prepare Requirements Models* activity, the resulting model is decomposed into more modularized software actors (see Fig. 4). In our example there are two alternatives to achieve the *Identification Details Collected* goal (see Fig. 3). One relies on the use of standard forms, while a second alternative is to use encrypted forms. If we apply horizontal rules (those that transform an initial i* model into a more modular one [8]) each alternative previously identified is moved to a different actor (see A1 and A2 dependencies in Fig. 4). Thus, in our *Medi@* example, we will have two SR i* models representing different configurations of system and to be considered in the next activity. For the sake of space, here we present both

alternatives in the same model. In fact, different SR models should have been used to represent each alternative. But an interested reader can easily extract them.

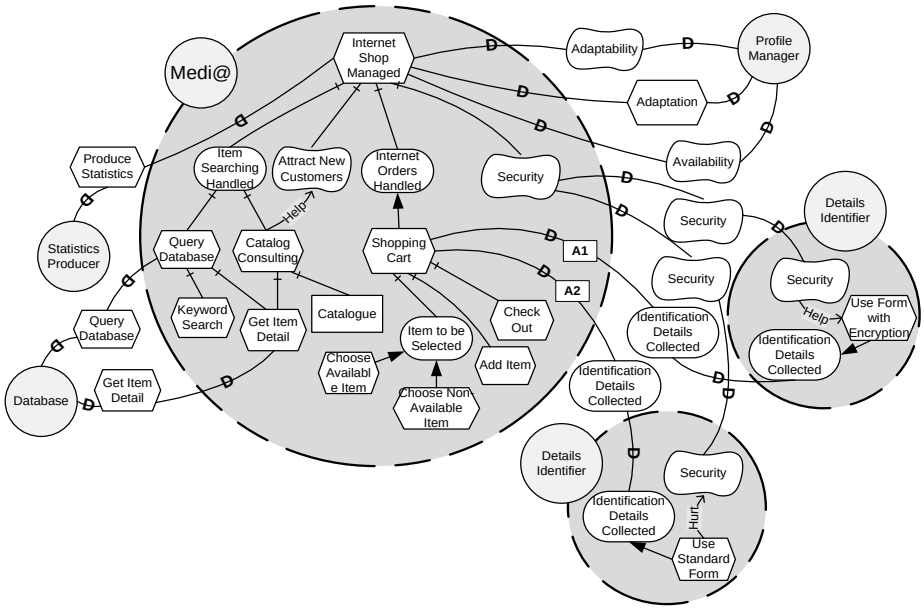

Fig. 4 – Modular i* model after model transformations.

4 Discussion

The aspectual approach contributes to increase modularity of i* models and, as demonstrated by the application of the metrics in [1], the number of concerns in a single module was reduced. Also, the models' visual complexity decreased, which may improve model understandability. This approach was applied to two case studies: the meeting scheduler problem [2] and a web-based information system [1].

However, the approach relies on aspect oriented principles. The big disadvantages of this strategy is the need to introduce new elements (namely aspects) in the original i* semantics. If the reader is familiarized with the aspect oriented principles this is not a cognitive burden. Otherwise some learning curve is required.

The second modularization approach relies on model transformations. The evaluation results demonstrated that it also promotes reduction of complexity in i* models. Besides, the proper definition of rules (for example in OCL, QVT or ATL) enables the semi-automatic derivation of modular i* specifications as well as can contribute to keep traceability among software artifacts. Note that since it does not introduce new elements to the i* syntax/semantics it is of easier adoption. This approach was applied to two case studies: a web-based recommendation system [10] and a web-based information system [9].

Both approaches can be used in a complementary way. The second approach could be used to decompose a system actor overloaded of responsibilities into several new

system actors, whereas the first approach could be used to identify the crosscutting concerns present in the i* models and separate them into aspectual elements.

5 Ongoing and Future Work

Currently we are evolving the Istar Tool [7] to support our modularity approaches. As future work, we intend to unify our approaches to decrease complexity, and to increase modularity and separation of concerns in i* models.

The identification of suitable metrics for goal modeling is also advancing, as other case studies are performed in an experimental setting. We also need to validate the metrics. We plan to define a trade-off analysis method to complement the aspectual i* process and to investigate the use of modularized i* models to support early architectural design. We aim at the decrease of coupling and improvement of separation of concerns, issues which are critical when dealing with large and complex projects. We also plan to evaluate and improve the quality of i* models [10].

References

1. Alencar, F., Castro, J., Lucena, M., Santos, E., Silva, C., Araújo, J., Moreira, A.: Towards Modular i* Models. Requirement Engineering Track at 25th ACM symposium on Applied Computing, SAC 2010. pp. 292-297 , Sierre, Switzerland (2010).
2. Alencar, F., Moreira, A., Araújo, J., Castro, J., Silva, C., Mylopoulos, J.: Towards an Approach to Integrate i* with Aspects. 8th Intl. Bi-Conference Ws. on Agent-Oriented Information Systems, AOIS'06 at CAISE'06. pp. 183-201, Luxembourg, June (2006).
3. Castro, J., Kolp, M. and Mylopoulos, J.: Towards Requirements-Driven Information Systems Engineering: The Tropos Project. Information Systems Journal, Elsevier, Vol 27: 365--89 (2002).
4. Estrada, H., Rebollar, A., Pastor, O. and Mylopoulos, J.: An Empirical Evaluation of the i* Framework in a Model-Based Software Generation Environment. In: CAiSE'06, LNCS 4001, Springer, pp. 513--527 (2006).
5. Filman, R., et al..: Aspect- Oriented Software Development. Addison-Wesley (2005)
6. Grau, G., Franch, X., Maiden, N. A. M.: PRiM: An i*-based process reengineering method for information systems specification. In: Information and Software Technology, v. 50, pp. 76-100 (2008).
7. IstarTool Project: A Model Driven Tool for Modeling i* models. Available at http://portal.cin.ufpe.br/ler/Projects/IstarTool.aspx, Mar (2010) .
8. Lucena, M., Silva, C., Santos, E., Alencar, F., Castro, J.: Applying Transformation Rules to Improve i* Models. In Software Engineering and Knowledge Engineering (SEKE 2009). pp. 43-48 , Boston, USA (2009).
9. Lucena, M., Castro, J., Silva, C., Alencar, F., Santos, E., and Pimentel, J.: A Model Transformation Approach to Derive Architectural Models from Goal-Oriented Requirements Models. 8th Int. Ws. On System/Software Architectures (IWSSA'09) LNCS. Berlin, Heidelberg: Springer, vol. 5872, pp. 370--380 (2009).
10. Ramos, R., et al.: AIRDoc-Approach to Improve Requirement Documents. In: XXII Brazilian Symposium on Software Engineering (SBES'08), pp. 1--16 (2008) .
11. Yu, E.: Modeling Strategic Relationships for Process Reengineering. Ph.D. thesis. Department of Computer Science, University of Toronto, Canada (1995).

From *i** to OO-Method: Problems and Solutions

Fernanda Alencar[1], Beatriz Marín[2], Giovanni Giachetti[2], Emanuel Santos[1], Oscar Pastor[2], Jaelson Castro[1], Xavier Franch[3]

[1]Universidade Federal de Pernambuco, Av. Prof. Luiz Freire s/n, 50740-540, Recife, Brazil
fernanda.ralencar@ufpe.br, {ebs, jbc}@cin.ufpe.br
[2]Universidad Politécnica de Valencia, Camino de Vera s/n, CP:46022, Valencia, Spain
{bmarin, ggiachetti, opastor}@dsic.upv.es,
[3]Universitat Politècnica de Catalunya, Omega-122, CP: 08034, Barcelona, Spain
franch@lsi.upc.edu

Abstract. Nowadays, the successful development of software products depends on a good understanding of the system requirements. The *i** framework offers expressive models to capture social and intentional characteristics in an organizational context. However, there is a well-known gap between intentional *i** models and other conceptual models used for software development. In order to reduce this gap, we have developed a transformation process to obtain from *i** models an appropriate input for the OO-Method Model Driven approach. In this paper, we present the problems detected from the application of this transformation process and the possible solutions, which are oriented to improve the alignment of *i** and OO-Method conceptual models.

Keywords: Goal-Oriented Requirement Engineering, i*, Requirement transformations, OO-Method, Model-Driven Development.

1 Introduction

Currently, an appropriate requirement specification is a key aspect for the correct development of software systems [9]. Requirements specification should include not only software specifications, but also multiple complementary views: intentional, structural, behavioral, functional, presentational, etc.

Goal-Oriented Requirements Engineering (GORE) stood out because it is mainly concerned with the stakeholder intentions and their rationales. Among the several GORE works, we have chosen the *i** framework [17] because it is a consolidated modeling technique with good tool support [7], and an abstract syntax formalized by a metamodel specification [10].

Nonetheless, it is still an open question the relationship between the intentional models described in terms of i* and the remaining conceptual models (e.g. structural, behavioral, functional, presentational views) used in other well-known model driven approaches.

In this paper we report on lessons learnt with a collaborative project[1], which aims at relating i* and the OO-Method approaches. The *OO-Method* is used as a reference MDD technology because it has been successfully applied to industrial software development [14] by means of the *OlivaNova* suite [3].

This rest of this paper is organized as follows: Section 2 presents our approach. Section 3 presents some problems that have arisen in the application of this approach and the solutions proposed for these issues. Finally, section 4 presents our conclusions and further work.

2 Relating i* and OO-Method Approaches

We propose a transformation process presented with the Business Process Modeling Notation (BPMN [13]) and composed by two sub-process, i* Models Analysis and Transformation Guidelines (further details in [1] and [2]), to obtain an OO-Method class model from an *i** model (see Fig. 1).

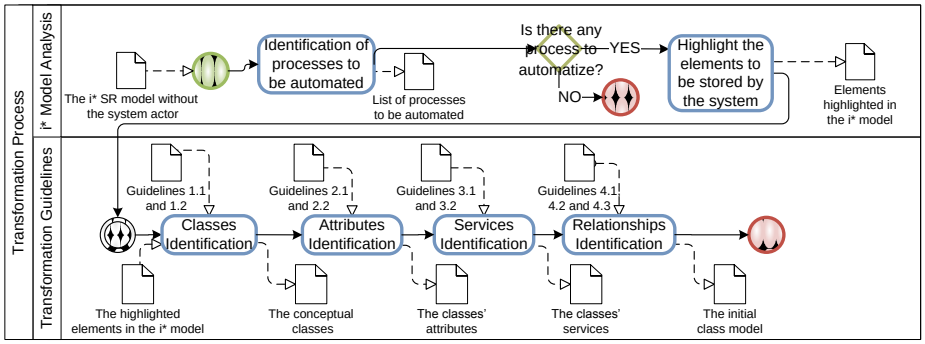

Fig. 1. The transformation process modeled with BPMN [13]

Initially, we analyze the goals defined in the Early SR model (see Fig.1, first activity: Identification of processes to be automated) to capture the organizational processes that we want to automate. Then, if there is any process to be automated, we highlight the intentional elements that are related to these processes (goals and tasks in the *i** model). Those elements will be related to the information and/or entities to be implemented by the intended system. From the list of identified intentional elements we obtain an initial skeleton of OO-Method conceptual model through the application of a set of transformation guidelines (second sub-process, see Fig.1).

Table 1 depicts a summary of the transformation guidelines that are used to explain the problems presented in this paper, which is a subset of the guidelines presented in [2]. This table shows the *i** constructs involved in the transformation, the additional

[1] CAPES-DGU: Integration of Organizational Modelling Techniques to Software Automatic Generation: OO-Method Case (in Portuguese). 2nd partial report. Ministério da Educação, Coordenação Geral de Cooperação Internacional Programa Brasil-Espanha da CAPES/DGU. Processo Nº 167/08, Brazil, 2010

information that must be considered to perform the transformation, and the target constructs of the OO-Method class model.

Table 1. Guidelines for the transformation of *i** models into OO-Method class models.

***i** Construct*	**Additional Information**	**Class Model Construct**
Actor		Class
Resource	Physical entity	Class
	Informational entity related to a physical resource or an actor	An attribute that represents information of the class generated from the actor or physical resource
	Resource in a decomposition tree	Input arguments for the service generated from the related task
	Dependum resource	Input argument of the depender task
	Physical entity inside of an actor boundary	An association between the classes generated from the physical resource and the owner actor
Task	Participating in a resource dependency as depender or dependee	A service of the class generated from the dependum resource
	If generates a resource	A creation service of the class generated from the resource
Dependency link	Where the *dependum* resource and the *depender* and *dependee* actors are transformed in classes	Associations are automatically defined among the generated classes

In order to illustrate, we present a brief example i* model (see Fig. 2) that is defined from the OO-Method case study presented in [11], which is related to the operation of a Photography Agency. This case study is also used in [1, 2]. In particular, the presented i* model shows the reception of work requests (i.e. job applications) from photographers that want to be hired. Due to space constraints, only a simplified version of the complete case study is presented. It is important to mention that, in the complete i* model, not all the i* elements are involved in the transformation process. Only those elements that are related to the intended system are considered (i.e. the involved actors).

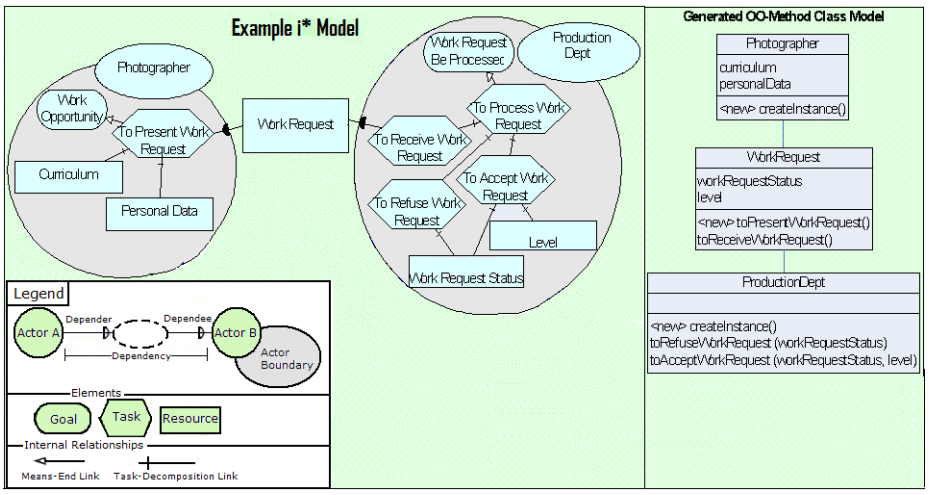

Fig. 2. A illustrate example

3 Some Problems and Solutions

In this section, we show some of the most relevant problems identified to perform an automatic transformation of *i** models into OO-Method Class Diagram, as suggested by the previously guidelines. For each issue a particular solution is proposed.

Problem 1. *It is not possible to automatically infer if a resource corresponds to a physical or an informational entity*. Since a physical entity is transformed into a class and an informational entity is transformed into an attribute, this distinction must be established. As a solution, we propose to extend resources with an attribute which defines the its type because we pretend.

Problem 2. *Differences in the Abstraction levels of i* and OO-Method.* The *i** requirements technique is oriented to capture aspects of the strategies and intentions involved in the relationships among actors (stakeholders), while the OO-Method is concerned with the representation of the functionality of the intended software system. Note that there is some abstraction gap. Furthermore, the transformation guidelines should only consider the subset of *i** elements that are required for the generation of an initial OO-Method class model. However, it is very important to keep the traceability information between *i** and OO-Method models. One possibility is to define an auxiliary model to record the traceability data. This intermediate model could be used specially for those *i** elements that do have direct representation in the OO-Method class model, e.g. goals.

Problem 3. *Two or more kind of elements of the i* model can be transformed into the same kind of element of the OO-Method class model.* As Table 1 shows that both actors and resources may be transformed into classes. Therefore, if we examine only the Class Diagram it is not possible to determine if it has been generated from an i* actor or resource. In other words, the traceability between the conceptual representation of the system and the corresponding requirement element is lost. This problem could also be solved by the intermediate model introduced as solution for the problem 2.

Problem 4. *Some relevant information of the i* model may be lost in the transformation process.* After the application of the transformation guidelines, it is not possible to identify from the generated Class Models: (i) which elements are related to the *depender*, *dependeee*, and *dependum* in the dependency links; (ii) the involved tasks decompositions; (iii) the services that are representing a *means* at the *i** models to preserve the means-end-links. The intermediate model presented as solution for problems 2 and 3 can also store the mapping required to identify these elements from the generated class model.

Problem 5. *It is not possible to directly specify which elements of the i* model must be automated.* According to the proposed transformation process (see Section 2), the transformation guidelines are only applied to those *i** elements that must be automated into the software system. Thus, to capture this information, we propose to use a metamodel extension mechanism to label the corresponding *i** model, for instance, such a UML profile [5]. In addition, the metamodel extension mechanism can also be

used to add the additional properties that are required to automate the transformation guidelines, such as the additional property that is required to solve Problem 1.

Problem 6. *The cardinalities of the associations between classes cannot be automatically inferred.* This problem is due to the difference in the abstraction level of *i** and OO-Method models. As a solution, we propose the introduction of a new property in the *i** model that allows the cardinality of the association among the generated classes to be automatically inferred. In fact in the context of Software Product Line development we have already proposed an i* extension that deals with cardinality (the so called i*-c) [16].

4 Conclusions and Further Work

In this paper we outline our attempt to relate intentional information described in terms of i* models and OO-Method conceptual models. Moreover, we highlight some shortfalls and discuss possible solutions for some of the identified problems.

Our proposal defines guidelines which be automated as well as some procedures which are semi-automatic or even manual, i.e. require human intervention [2]. The solutions presented in this paper are oriented towards the fully automation of the process. Thus, we want to minimize the dependency on highly experienced analysts and designers to manually transform the requirements models into appropriate OO-Method models.

Initial results of our approach are presented in [6]. However, it is important to note that the quality of the GORE (i*) models directly affects the quality of OO-Method conceptual models. In our proposal, we assume that the *i** models are of high standard, i.e. do not present defects (omissions, inconsistency, erroneous facts, ambiguous, etc.). However, this assumption may be unrealistic. Thus, we are also working in proposal to evaluate the quality of requirements models [4, 15].

As future work, we plan to apply the transformation guidelines to different case studies in order to evaluate the correctness and completeness of our proposal. In addition, we plan to formalize and automate the guidelines using metamodeling standards (such as MOF [12]) and model-to-model transformations technologies (such as ATL [8]). Finally, we also consider the definition of metamodel extensions for the *i** framework in order to improve the modeling facilities for MDD environments and to completely automate the transformation of GORE models since we intend to preserve the automate trace between rationales and the data design.

References

1. Alencar, F., Pastor, O., Marín, B., Giachetti, G., Castro, J.: Aligning Goal-Oriented Requirements Engineering and Model-Driven Development. Poster in the 11th International Conference on Enterprise Information Systems (ICEIS´09), May, Milan, Italy (2009)
2. Alencar, F., Pastor, O., Marín, B., Giachetti, G., Castro, J., Pimentel, J.: From i* Requirements Models to Conceptual Models of a Model Driven Development Process. 2nd W. Conf. on the Practice of Enterprise Modeling (POEM´09), Stockholm, Sweden (2009)
3. Care Technologies Company. Available at www.care-t.com. Last access: March (2010).

4. Franch, X.: A Method for the Definition of Metrics over i* Models. In: 21st Int. Conf. on Advanced Information Systems (CAiSE 2009), pp. 201--215a. Springer-Verlag LNCS (2009)
5. Giachetti, G., Marin, B., Pastor, O.: Integration of Domain-Specific Modeling Languages and UML through UML Profile Extension Mechanism International Journal of Computer Science and Applications, vol. 6 nº 5, 145--174 (2009)
6. Giachetti ,G., Alencar, F., Marín, B., Pastor, O., Castro J.: Beyond Requirements: An Approach to Integrate i* and Model-Driven Development. In: XIII Ibero-American Conference on Software Engineering (CIbSE2010), April, Cuenca, Ecuador (2010)
7. Grau, G., Franch, X., Ávila, S.: J-PRiM: A Java Tool for a Process Reengineering i* Methodology. In: RE 2006: p.352--353 (2006)
8. Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I.: ATL: A model transformation tool. Science of Computer Programming, vol. 72 nº 1-2, 31--39 (2008)
9. Lamsweerde, A.v.: Systematic Requirements Engineering - From System Goals to UML Models to Software Specifications. Wiley, (2008).
10. Lucena, M., Santos, E.,Silva, M. J. , Silva, C., Alencar, F. , Castro, J.: Towards a Unified Metamodel for i*. In: 2nd IEEE Int. Conference on Research Challenges in Information Science (RCIS'08), Marrakech. Proceedings of the RCIS'08, pp. 237--246 (2008)
11. Marín, B., Giachetti, G., Pastor, O.: The Photography Agency: A case study of the OO-Method Approach. Technical Report DSIC-II/13/08, Universidad Politécnica de Valencia, Valencia, España (2008)
12. OMG: MOF 2.0 Core Specification (2006)
13. OMG: Business Process Modeling Notation version 1.1 (2008)
14. Pastor, O. and Molina, J. C.: Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling, Springer-Verlag 1st ed., Springer, New York, New York (2007)
15. Ramos, R.A.: AIRDoc - An Approach to Improve the Quality of Requirements Documents: Dealing with Use Case Models. PhD Thesis. Federal University of Pernambuco, (2009)
16. Silva, C., Borba, C., Castro, J.: G2SPL: A Goal Oriented Requirements Engineering Process for Software Product Line (In Portuguese: G2SPL: Um Processo de Engenharia de Requisitos Orientada a Objetivos para Linhas de Produtos de Software). In: Proceedings of 13th Workshop on Requirements Engineering (WER'10) (2010)
17. Yu, E.: Modelling Strategic Relationships for Process Reengineering, PhD Thesis, University of Toronto, Toronto, Canada (1995).

The Evolution of Tropos: Contexts, Commitments and Adaptivity

Raian Ali, Amit K. Chopra, Fabiano Dalpiaz,
Paolo Giorgini, John Mylopoulos, and Vitor E. Silva Souza
Department of Information Engineering and Computer Science
University of Trento – Italy
{ali, chopra, dalpiaz, paolo.giorgini, jm, vitorsouza}@disi.unitn.it

Abstract. Software evolution is the main research focus of the Tropos group at University of Trento (UniTN): how do we build systems that are aware of their requirements, and are able to dynamically reconfigure themselves in response to changes in context (the environment within which they operate) and requirements. The purpose of this report is to offer an overview of ongoing work at UniTN. In particular, the report presents ideas and results of four lines of research: contextual requirements modeling and reasoning, commitments and goal models, developing self-reconfigurable systems, and requirements awareness.

1 Introduction

At the University of Trento (UniTN), research on Tropos is conducted within the Software Engineering and Formal Methods research program[1]. Currently, our main research challenge is facilitating software evolution so that systems may be able to evolve in response to changes in their operational environment and, more pertinently, in their requirements themselves. We are addressing this challenge by formalizing high-level concepts, and developing tools, techniques, and methodologies around these concepts. Our approach is to support evolution via design-time models that are made available at runtime. These models capture stakeholder intentions and commitments, social interactions, business processes, and organizational goals.

Evolution can be automatic (self-adaptation), or manual, or something in between. When evolution is automatic, design-time models determine what is to be monitored, what are the possible ways to adapt the behavior of the system when it deviates from its intended purposes, and how to evolve the system at runtime. When evolution is manual, these models are used as support for human activities. They offer a comprehensive view of the requirements and traceability links between elements of these models and the software code.

The rest of the report describes our current research objectives and activities, our latest results, and future work.

[1] http://www.troposproject.org

2 Objectives and Scientific Contributions

Our activities in the area of software evolution may be broadly divided into the following topics: contextual requirements modeling and reasoning, commitments and goal models, architectures for self-reconfigurable systems, and requirements awareness. The following elaborates on each.

Modeling and reasoning about contextual requirements. Advances in computing, sensors, and communication technology have given rise to new computing paradigms such as ambient, ubiquitous and pervasive computing. These paradigms weave computing systems with human living environments to transparently meet human needs. Context, a core element of these paradigms, can be defined as the reification of the environment, and includes whatever provides a surrounding within which the system operates [11]. Before influencing the behavior of software, context influences the behavior of users. It influences user goals and their choices in determining how to reach these goals. Capturing this latest influence is an essential step towards software developed to meet user requirements in different contexts.

In our research, we are interested in modeling and reasoning techniques for developing software systems expected to operate in varying contexts. We extend the Tropos goal modeling framework [1, 2] with context and allow the designer to capture the relation between the space of variants of a goal model and the context in which each variant is applicable. The framework defines a set of modeling constructs to analyze and discover relevant information the system needs at runtime to identify and characterize the context in which it is operating. We also propose various reasoning techniques to support the analysis. Particularly, we are interested in (i) checking the consistency of contextual goal models, (ii) detecting harmful interplays between tasks of a goal model originating from conflicting changes over the context, (iii) deriving goal model variants that comply with certain context and users' priorities, and (iv) deriving a subset of executable tasks that can satisfy at the minimal cost users' goals in all analyzed contexts. A prototype tool has been implemented to support reasoning about contextual goal models. The modeling and reasoning framework has been applied on two systems scenarios: a smart home for people with dementia, and a museum-guide to support museum visitors.

Social commitments and goal models. Requirements modeling for open settings such as for service-oriented and sociotechnical systems pose new challenges due to the autonomy and heterogeneity of the participants, that is, *agents*. Such settings are also highly dynamic—agents may not even know each others' identities before runtime [3].

The *i** approach was influential in emphasizing the social nature of requirements fulfillment—agents often *depend* on others to achieve their goals. An *i** dependency involves one actor wanting something, and another being able and committed to delivering that something. However, *i** does not does not achieve a clean separation between an agent's internals and its social relationships with others. For example, the formalization of dependencies refers to the ability of the dependee, that is, its internal routines. As a result, *i** has limited applications in open settings.

Our recent work on modeling agents and social relations among them replaces dependencies with interaction protocols and social commitments [3, 4]. Social commitments are brought about and manipulated solely by interaction among agents [7]. The protocols serve as specifications of convention. An agent's social commitments cleanly capture an agent's external relationships with others without referring to any agent's internals. Given an agent's goal model and capabilities—the specification of its internals—one can reason if a particular protocol *supports* the agent's goals. Specifically, support for an agent's goal may be determined objectively without referring to the agent's beliefs about others. By contrast, in *i**, an agent's belief about the workability of dependencies must be justified.

An agent's beliefs about another's ability or intentions with respect to a certain goal may be important in arriving at certain decisions. However, it is also important to systematically understand and separate the internal from the external—this enables us to build agent reasoning in a modular fashion. For example, an agent may first determine if a protocol is suitable for its goals, and then select with whom to interact within that protocol based on its internal model of others.

Social commitments are more expressive than dependencies in *i**. Social commitments are conditional, thus enabling capturing reciprocity among agents—that if one agent brings about some condition, then the other bring about another condition. Moreover, social commitments also refer to the contextual setting—these are often important in contractual settings. Formal reasoning for social commitments is also well-developed [12].

Architectures for self-reconfigurable sociotechnical systems. A sociotechnical ystem (STS) is an interplay of humans, organizations and technical systems. STSs are distributed systems where a number of autonomous and intentional actors interact in order to achieve their respective objectives. STSs are characterized by dynamism, unpredictability and weak controllability. The operational environment is subject to sudden and unexpected changes, actors may join and leave the system at will, social dependencies between actors are at risk because of actors' autonomy, and actors may fail in achieving their goals. The interests of the actors can be supported technologically by a software architecture that (i) monitors the actors' behavior, (ii) diagnoses failures against correct behavioral models, and (iii) reacts to failures via compensation actions. We have proposed an architecture based on this cycle in [5].
Our architecture becomes an integral component of an STS. The correct behavior of actors is specified by their respective goal model. The architecture observes the actions performed by participating actors, compares the monitored data against goal models, and enacts reconfigurations in response to failures. The implemented algorithms are based on the Belief-Desire-Intention paradigm [6]. Indeed, an actor participating in an STS behaves correctly if, whenever a goal is activated, it selects and executes a plan that eventually will lead to the achievement of that goal. Failures occur if the actor does not carry out the plan correctly, doesn't perform any action, or if a dependee does not bring about the dependum for the depender. Reconfiguration actions take into account the autonomy and uncontrollability of the participants: the architecture can (i) perform real actions by controlling actuators; (ii) remind or suggest the actors what to do; and (iii) assign some responsibilities to external agents.

The architecture has been applied to a smart-home case study, where the mission of the system is to support a patient in his everyday activities.

Requirements awareness. Lately there has been growing interest in systems that can adapt to changes in their environment or requirements during run-time. This kind of adaptive system generally uses some kind of feedback loop to monitor, diagnose and compensate these adverse situations. We're interested in studying the requirements that lead to this feedback loop functionality and we propose a new class of requirements, called Awareness Requirements (AwReqs). AwReqs are requirements that refer to other requirements, quality constraints or domain assumptions, and their success or failure. As a simple example, consider the requirements for a meeting scheduler. To schedule a meeting, one should know about the agenda of the participants of the meeting, arrange the meeting (set date/time, book room), and finally notify all participants about it. As a requirement for adaptation, we may want to say that the goal of notifying the participants should never fail, or that booking a room should succeed 90% of the times over any given month. To these AwReqs, the requirements engineer can attach compensation actions that would get the system back to normal operation. AwReqs can also refer to quality constraints (QCs) and domain assumptions (DAs). If there was a QC stating that meetings should have 75% attendance, an AwReq could say that this quality constraint should succeed 90% over every week. AwReqs for DAs are analogous. And since AwReqs are requirements themselves, one could create an AwReq that refers to the success of another AwReq (a meta-AwReq). Our research on this topic is detailed in [9], where we also propose: (a) a formalization using OCL; (b) elicitation techniques for AwReqs; (c) patterns for AwReqs; (d) graphical notation; and (e) a systematic process to go from AwReqs to feedback loops.

3 Future work

Future work on contextual requirements includes applying the framework developed so far to security requirements. The main idea is that contexts can influence security requirements and then security has to be analyzed and handled according to the context where the system operates. For example, in an emergency situation (such as fire), a person would allow the rescue team to access his personal data such as his location and his health status, while in a normal situation the same person would have more restricted security concerns. Our interest here is to extend the goal-oriented requirements engineering for security to cope with contextual security requirements introducing new constructs and different forms of reasoning specific for security.

Concerning commitments and self-reconfigurable systems, we are currently analyzing how a monitor-diagnose-compensate loop changes when we consider commitments together with goals. We will develop runtime agent reasoning for actors specified as goals, qualities and commitments. A correctness property, from an actor's perspective, would take the form of policies: achieve so and so goals but without violating so and so commitments. The key here is to formalize the notion of a variant in terms of both goals and commitments, and then understand adaptation as switching between variants — similar to the development in [10].

With respect to awareness requirements, the research is at its beginnings and there is much to be done. First and foremost, we intend to conduct case studies to assess our proposal. For that matter, we plan on developing a prototype framework that implements feedback loops from requirement models, most likely using previous experience in diagnosing frameworks [8]. Other challenges that lie ahead include analyzing the role of contexts with respect to AwReqs, implementing consistency checking for the model, and studying predictive and evolutionary features that could improve adaptability.

Acknowledgments This work has been partially funded by the EU Commission, through projects SecureChange, COMPAS, NESSOS and ANIKETOS.

References

1. R. Ali, F. Dalpiaz, and P. Giorgini. Location-Based Software Modeling and Analysis: Tropos-Based Approach. Proceedings of the 27th International Conference on Conceptual Modeling (ER'08), pages 169–182, 2008.
2. R. Ali, F. Dalpiaz, and P. Giorgini. A Goal Modeling Framework for Self-Contextualizable Software. Proceedings 14th International Conference on Exploring Modeling Methods in Systems Analysis and Design (EMMSAD'09), LNBIP 29-0326, pages 326–338. Springer, 2009.
3. A. K. Chopra, F. Dalpiaz, P. Giorgini, and J. Mylopoulos. Modeling and Reasoning about Service-Oriented Applications via Goals and Commitments. Proceedings 22nd International Conference on Advanced Information Systems Engineering (CAiSE'10), 2010. to appear.
4. A. K. Chopra, F. Dalpiaz, P. Giorgini, and J. Mylopoulos. Reasoning about Agents and Protocols via Goals and Commitments. Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'10), 2010. to appear.
5. F. Dalpiaz, P. Giorgini, and J. Mylopoulos. An Architecture for Requirements-Driven Self-Reconfiguration. Proceedings 21st International Conference on Advanced Information Systems Engineering (CAiSE'09), LNCS 5565, pages 246–260. Springer, 2009.
6. F. Dalpiaz, P. Giorgini, and J. Mylopoulos. Software Self-Reconfiguration: a BDI-Based Approach. Proceedings 8th International Conference on Autonomous Agents and Multiagent Systems, pages 1159–1160. IFAAMAS, 2009.
7. M. P. Singh. Agent communication languages: Rethinking the principles. IEEE Computer, 31(12):40–47, Dec. 1998.
8. V. E. S. Souza and J. Mylopoulos. Monitoring and Diagnosing Malicious Attacks with Autonomic Software. Proceedings 28th International Conference on Conceptual Modeling (ER'09), pages 84–98, Gramado, Brazil, 2009. Springer.
9. A. Lapouchnian, V. E. S. Souza, and J. Mylopoulos. Awareness Requirements for Adaptive Systems. Submitted for review, 2010.
10. Ji Zhang and B. H. C. Cheng. Model-Based Development of Dynamically Adaptive Software. Proceedings 28th International Conference on Software Engineering (ICSE), pages 371–380, 2006.
11. A. Finkelstein, A. Savigni. A Framework for Requirements Engineering for Context-Aware Services. Proceedings of STRAW'01, 2001.
12. A. K. Chopra and M. P. Singh. Multiagent Commitment Alignment. Proceedings of the Eighth International Conference on Autonomous Agents and Multiagent Systems, 2009, pages 937—944

Definition and Uses of the *i** Metamodel[1]

Carlos Cares[1,2], Xavier Franch[1], Lidia López[1], Jordi Marco[1]

[1]Universitat Politècnica de Catalunya, Omega-122, 08034 Barcelona, Spain
{ccares, franch}@essi.upc.edu, {llopez, jmarco}@lsi.upc.edu
[2]Universidad de La Frontera, Av. Francisco Salazar, 01145 Temuco, Chile

Abstract. The clear definition of a metamodel can be considered helpful for any conceptual modeling approach, and the *i** framework is not an exception. Agreeing on a metamodel for *i** can be considered even more convenient than ever when we are aware of the different dialects and variations that the community proposed, and keep proposing, over the seminal *i** definition. In this paper we present the revised version of the *i** metamodel proposed by the GESSI research group at 2005 and we report some current contexts of use: 1) definition of a data interchange format; 2) definition of the inheritance construct; 3) definition of a modularity construct; and 4) definition of a metrics framework.

Keywords: Goal-Oriented Requirements Engineering, *i**, Metamodel, iStarML.

1 Introduction

Since it was first released, the *i** framework has been adapted to the needs of specific research groups that wanted to represent concepts specific of their software engineering problem, like security, law compliance, trust modelling, architectural design, model-driven development and agent-orientation, among others. Even, the *i** framework itself has experienced a natural evolution that has led to a slightly modified version available in the *i** wiki. This set of main *i** variations have been object of our study which we have described and genealogically analysed in [1].

This diversity, although not necessarily pernicious, has some consequences. When reading a work around the *i** framework, it is necessary first to understand what concrete version of *i** is being used. If the contribution is based on the original framework, sometimes the authors declare which version are using (lately, it is happening to be the wiki version), but sometimes there is no explicit mention, which often makes the reader a bit hesitant about details of the proposal being presented. On the other hand, if the work is proposing some new variation, the semantics is sometimes given informally or by using a formalism which is not easy to align with the available descriptions of *i**. In order to deal with this problem we proposed at 2005 a reference metamodel for *i** [2] where particular metamodels of *i** variations can be obtained by applying UML refactoring operations. Since then, we have revised this metamodel upon which we have based our research work on: *i** inheritance, requirements interoperability, and metrics frameworks among others.

[1] This work has been partially supported by the Spanish project TIN2007-64753.

2 Objectives of the Research

In this report-of-progress paper, we review our proposal of metamodel for the *i** framework, we show our updated version and outline several contexts in which it has been used to formalize our *i**-related proposals. In particular, our main objective has been to define a metamodel able to express most of the current variations and extensions of the *i** framework and to use this metamodel as a reference model for our lines of research. More precisely, we have worked with four specific objectives in mind: (i) To define a metamodel for the *i** framework not bound to any particular technology, (ii) To use this metamodel as the underlying baseline for defining a *i** diagram interchange format, iStarML, (iii) To use this metamodel as a reference framework over which formulating our own extensions and variations of *i**, namely a full definition of inheritance and the concept of module, and (iv) To use this metamodel as the syntactic baseline over which formulating a framework for the definition of metrics on *i**.

3 Scientific Contributions

As a first tangible contribution, we are proposing an *i** metamodel compliant with the objectives stated in Section 2 (generality, flexibility, technological independence; see Figure 1). It has been built by consolidating several main versions of the framework, as thoroughly described in [1]. It presents some superclasses of interest (*Node* as the most general one, and also *DependableNode* and *IntentionalElement*) and then the most relevant *i** concepts: *Actor*, *SR-Element* and *SD-Dependum* as classes, and *Relationship* and *Link* as association classes, all of them with the appropriate subclasses.

The iStarML interchange format [3, 4] has being designed starting from the metamodel. The format implements the metamodel as an XML grammar and also supports the possibility of extension with new constructs. The ccistarml v0.6 Java package (http://www.essi.upc.edu/~ccares/ccsoftware/ccistarml_v0.6.1.zip) allows creating, importing and handling iStarML-compliant files. Several tools have been and are being customized to support importing and exporting iStarML. Remarkably, the H*i*ME tool (http://www.lsi.upc.edu/~llopez/hime/) supports both export and import, whilst we have developed an online translator from OME .tel files into iStarML (http://www.essi.upc.edu/~ccares/index.php?section=ometranslator). It is planned for adoption in a next release of TAOM4E (http://sra.itc.it/tools/taom4e/).

The inheritance proposal presented in [5, 6] and the module construct as defined in [7] have been related to the metamodel. This makes the definition of both concepts easy to integrate into the *i** framework. In the case of inheritance, the integration is very tight since inheritance appears in the metamodel itself, therefore we are just providing a more detailed definition of a core concept. The concept of inheritance has been implemented in the H*i*ME tool presented above, supporting the three operations identified in [5]: extension, refinement and redefinition. As for modules, the option has been to integrate in a loosely coupled way, meaning that modules are linked to the metamodel but the metamodel is not modified.

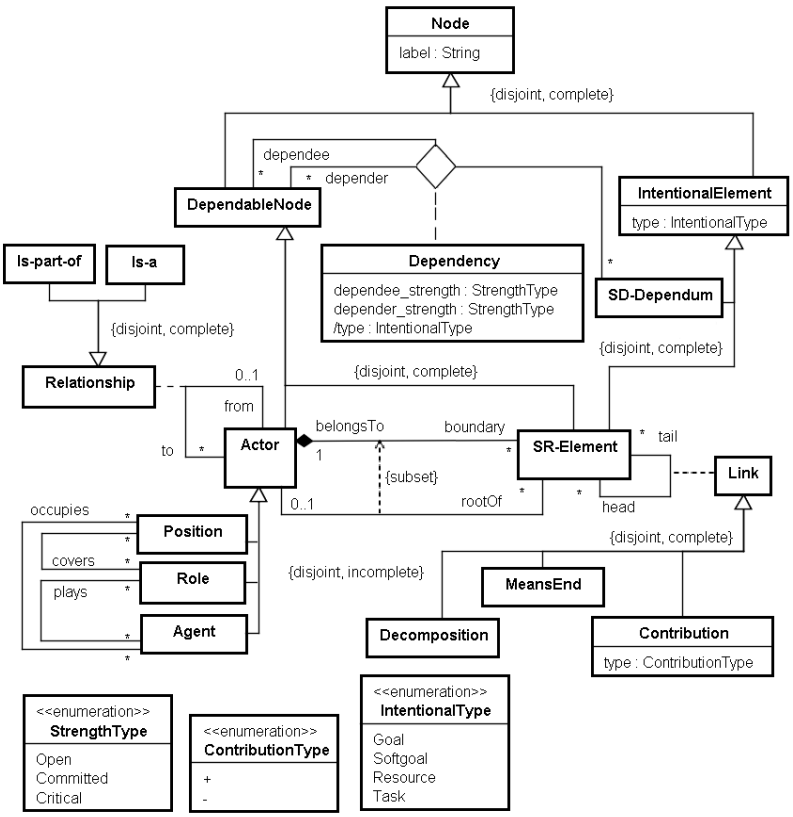

Fig. 1. The *i** metamodel.

Figure 2 provides an excerpt of both modifications. We may observe that in addition to actor inheritance, we are allowing the refinement of dependencies and intentional elements in general, and extension and redefinition only for SR-elements. Concerning modules, in addition to the general concept of module, we identify SD-modules and SR-modules. SR-modules may be of different types too, e.g. for storing means-end decompositions. New classes are introduced for these concepts, coupled to their counterpart elements in the metamodel.

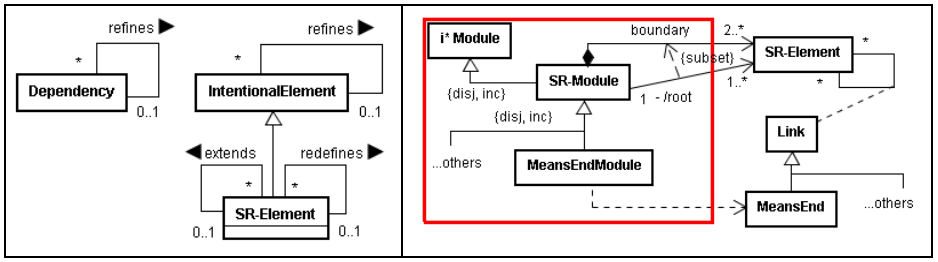

Fig. 2. Extending the *i** metamodel, excerpts: inheritance (left) and modules (right).

As a last remarkable use of the *i** metamodel, we have used it as the baseline for formulating an *i** metrics definition framework. In [8], we have provided a catalogue of metric definition patterns in which their form is expressed as an OCL template involving metamodel elements (see Figure 3 for an example). In [9], we have illustrated a particular case of application, the definition of a metric suite for business process modeling. In this exemplar we may observe the general procedure in which an extension of the *i** metamodel for capturing the essential concepts of the domain of interest (business processes in this case) is needed. Then, the metric suites that exist in this domain are mapped into their counterparts using a metamodel mapping.

Name	**Dependency-Based (Metrics Definition -> Quantitative -> Structural)**
Context	Some metrics have sense when applied to dependency links
Problem	The metrics will depend not just on the characteristics of the dependency link itself, but also on the two actors that act as depender and dependee
Solution	Identify three different factors that influence the metrics: one bound to the dependency link itself (probably related with the type of its dependum), and the others to the two actors, depender and dependee
Required knowledge	The effect of the depender, the dependee and the dependum in the metric, represented by three functions: filter: Dependum → Float correctionFactorDepender: Actor → Float, correctionFactorDependee: Actor → Float
Form	**context** Dependency::metric(): Type **let** ownerActor(x: DependableNode): Actor = **if** x.oclIsTypeOf(Actor) **then** x **else** x.owner **in**: **post**: result = self. dependum.filter() * ownerActor(self. depender).correctionFactorDepender() * ownerActor(self.dependee).correctionFactorDependee()

Fig. 3. Defining metric patterns by means of OCL templates over the *i** metamodel.

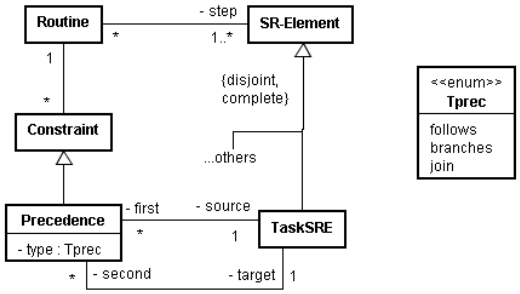

Fig. 4. Extending the *i** metamodel for defining metrics over business process models.

4 Conclusions

This main purpose of this paper has been twofold. On the one hand, illustrating the form that the *i** reference metamodel takes by including a particular proposal. On the second hand, providing an overview of the different uses of such a metamodel in different contexts that may be of general interest for the *i** community: for model interchange, for definition of new concepts and for definition of metrics.

Several authors agree on our belief that the existence of an *i** metamodel could bring some benefits (shared understanding, tool interoperability, etc.) to the *i** community [10, 11], although the statement could in fact be a matter of discussion, and in fact some other researchers advocate for more focused metamodels like the Tropos metamodel [12] and the GRL metamodel [13]. Our view is that the differences among the several existing approaches concerning the core concepts of *i** are not so severe as to prevent the proposed agreement, whilst the potential benefits seem attractive enough.

5 Ongoing and Future Work

We think that the most important future work is a community work: agreeing on a metamodel as the "official" *i** framework metamodel (being the one presented here or other), making it available in the *i** wiki for reference. Its existence shall provide a shared context to *i** researchers and practitioners, and shall serve as reference for: new extensions and variations, semantic and pragmatic agreements, tool support, etc.

Concerning our particular future work, we plan to advance in the following research lines: (1) using iStarML as the technological infrastructure to connect as many available tools as possible. This also means coping with the mapping problem where a construct that is used in some source tool is not supported in some destination tool. An example of how to deal with this case has been presented in [4]; (2) completing the definition of inheritance, providing the necessary restrictions on the use of the identified operations (extension, refinement and redefinition [5]) in the form of OCL constraints over the metamodel elements; (3) creating a comprehensive catalogue of metrics suite based on the use of the patterns identified in our previous work; (4) implementing the concepts presented here (inheritance, modularity and metrics) using our H*i*ME tool, and (5) providing semantics to the metamodel (i.e., how the different concepts proposed in the metamodel should be interpreted).

References

1. Cares, C., Franch, X., Mayol, E., Quer, C. "A Reference Model for *i**". In: *Social Modelling for Requirements Engineering*, The MIT Press, 2010 (in press).
2. Ayala, C.P., Cares, C., Carvallo, J.P., Grau, G., Haya, M., Salazar, G., Franch, X., Mayol, E., and Quer, C. "A Comparative Analisys of *i**-Based Goal-Oriented Modelling Languages". AOSDM 2005.
3. Cares, C., Franch, X., Perini, A., Susi, A. "iStarML: An XML-based Model Interchange Format for *i**". In: *Procs. of the 3rd International i* Workshop*, CEUR-WS 322, 2008.
4. Cares, C., Franch, X., Perini, A., Susi, A. "Towards Interoperability of *i** Models using iStarML". *Computer Standards & Interfaces*, Elsevier (in press).
5. Clotet, R., Franch, X., López, L., Marco, J., Seyff, N., Grünbacher, P. "The Meaning of Inheritance in *i**". AOIS 2007.
6. López, L., Franch, X., Marco, J. "Defining Inheritance in *i** at the Level of SR Intentional Elements". In: *Procs. of the 3rd International i* Workshop*, CEUR-WS 322, 2008.
7. Franch, X. "Incorporating Modules into the *i** Framework". CAiSE 2010.

8. Franch, X., Grau, G. "Towards a Catalogue of Patterns for Defining Metrics over *i** Models". CAiSE 2008.
9. Franch, X. "A Method for the Definition of Metrics over *i** Models". CAiSE 2009.
10. Cabot, J., Yu, E. "Improving Requirements Specifications in Model-Driven Development Process". ChaMDE 2008.
11. Moody, D.L., Heymans, P., Matulevicius, R. "Improving the Effectiveness of Visual Representations in Requirements Engineering: An Evaluation of *i** Visual Syntax". RE 2009.
12. Susi, A., Perini, A., Mylopoulos, J., Giorgini, P. "The Tropos Metamodel and its Use". Informatica, 2005.
13. Amyot, D., Horkoff, J., Gross, D., Mussbacher, G. "A Lightweight GRL Profile for *i** Modeling". RIGiM 2009.

Itemized Strategic Dependency: a Variant of the *i** SD Model to Facilitate Knowledge Elicitation

Hesam Chiniforooshan Esfahani[1], Eric Yu[2], Maria Carmela Annosi[3]

[1]Department of Computer Science, University of Toronto
[2]Faculty of Information, University of Toronto
[3]Ericsson Software Research, Ericsson Telecomunicazioni S.p.a, Italy
[1]hesam@cs.toronto.edu, [2]yu@ischool.utoronto.ca, [3]mariacarmela.annosi@ericsson.com

Abstract. This paper introduces a variant of the *i** Strategic Dependency (SD) model, called Itemized Strategic Dependency (ISD). The goal of introducing ISDs was to use a simplified version of SD diagrams to model actor dependencies in requirements and process engineering. We used ISD models during the early stages of a software process improvement initiative in one of R&D organizations at Ericsson Italy. In this paper, we explain how ISDs helped us to interact with people who were not familiar with the *i** notation; to elicit their knowledge of organization; and to reveal the hidden problems of their software development process.

1 Introduction

It has been commonly accepted that software development, in many of its aspects, is a human-based activity. The reliance of software companies on the collaboration of project stakeholders (including analysts, designers, developers, customers, etc.) often causes a network of interactions, which its complexity rapidly grows as the size of organizations or projects increase. This phenomenon usually coincides with the emergence of inefficiency symptoms in the process of software development, such as miscommunication of people, loss of knowledge, rework, excessive documentation, and ambiguity of software architecture.

The *i** Strategic Dependency (SD) modeling [1] has been introduced as a way of conceptualizing the collaboration complexities that exist as dependency relations among organizational or system actors. SD models have been used in Requirements Engineering (RE) and Process Engineering (PE). While *i** modeling (including SD and SR) can be used by specially-trained analysts, more effective modeling and knowledge elicitation can be achieved when domain stakeholders are able to directly contribute to the construction and analysis of such models. When following the original format of SD models [1], as the number of actors and dependencies increases, the diagram becomes cluttered and cumbersome to extend or modify. In this paper we introduce a variant of the SD model, called *Itemized Strategic Dependency* (ISD), in order to promote the process of knowledge elicitation during the early stages of RE and PE, and to facilitate the understandability of models by those who are not familiar with the *i** notation. ISD models have been successfully used in a Software Process

Improvement initiative that we are currently involved in one of the R&D units at Ericsson Italy.

2 Objectives of the Research

As mentioned before the main objective of this research is to promote the understandability of the *i** SD models, by simplifying the representation style of strategic dependencies. To achieve this objective, we need to answer the following two questions:

1. What needs to be retained in the simplified SD, and what can be omitted?
2. What are the tradeoffs of the new approach (e.g. what info can be lost)?
3. How will the new approach work out in a real life project?

3 Itemized Strategic Dependency Models

The complexity of *i** SD models happens when the number of dependums between actors increases. The ISD models are introduced to simplify the visual representation of SD models, while conveying the same information. In such models, a single dependency link is used to represent all the dependencies from one actor to another in one direction, with the dependums written as an itemized list associated with the link. Figure 1 and 2 provide two examples. The dependency link is a continuous curved line from the depender actor to the dependee actor with a single "D" near the middle. Unlike the original SD, the dependums are not enclosed in different shapes according to dependum types.

To further simplify modeling and to support incremental elicitation, we define two submodels of the ISD: *Functional ISD*, for representing functional dependencies; and *Quality ISD*, for representing the quality attributes of dependency relations. We also define a concept of the *Viewpoint Actor* (VA), the actor from whose viewpoint the model is constructed. The viewpoint actor is denoted by a thicker circle. For an ISD with a viewpoint actor, relationships among other actors are likely to be sketchy and incomplete. This form of the ISD is intended especially for interacting with stakeholders during individual interviews. When no viewpoint actor is indicated in an ISD model, the model represents the understanding of the analyst/modeler, typically gained by integrating the viewpoints of all actors.

3.1 Functional Itemized Strategic Dependency (FISD)

Each FISD shows all the functional dependencies that a Viewpoint Actor has with other organization actors. Here, by *functional dependency* we mean all dependency relations that are not related to any particular quality attribute. Such dependency relations would be Resource, Task, or Goal dependencies in an original SD representation format. As shown in Figure 1 each dependency relation contains a list of dependency items, for which the SA depends on other organizational actors, or vice

versa. For instance, in our case study in Ericsson, Designers were depending on System Manager for the Node Requirements Specification (NRS), Feature Specification, and Pre-study documents; also for setting up meetings to clarify these documents requirements.

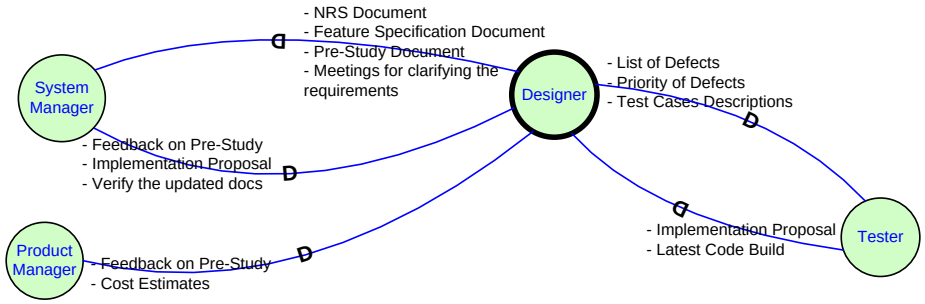

Figure 1: Sample FISD, representing the functional dependencies of Designer (the Viewpoint Actor) and System Manager

If we wanted to represent these dependency relations in original SD models we had to depict 14 dependency links. For instance, just for representing the SD relations of Designers to System Manager we had to draw three resource dependencies for three documents, and one task dependency relations for setting up the meeting. In should be mentioned that since the sample FISD in Figure 1 is developed from the viewpoint of designers, the represented dependencies of the System Manager (or other actors) to Designer is the perception of designers, not necessarily in agreement with System Manager's perception. The complete FISDs of an organization can be developed by aggregating the VP-based FISDs, developed for each organizational actor.

3.2 Quality Itemized Strategic Dependency (QISD)

QISDs represent more delicate aspects of dependency relations, by listing the quality attributes of the functional dependencies represented in FISDs. In other words, QISDs represent the Softgoal dependency relations, which are related to the functional dependencies identified in FISDs. Every FISD can be transformed to a number of QISDs, each representing the pair-wise dependency relations of the VA and a subset of other actors. Figure 2 shows a QISD we developed for our case study, and reveals some of the quality attributes that are expected from dependums of dependency relations between Designer and System Manager. For instance, it shows that designers expect that the NRS documents be sufficiently detailed, technical, and regularly updated. On the other hands, System Manager expects that the designers' feedback on Pre-Study document to be reliable and accurate.

As shown in the Figure 2 for every functional dependency (represented in FISD models) there is a corresponding entry in the QISD models. If the quality attributes of that functional dependency were already extracted, they were written on the dependum list, otherwise, a number of question marks represent the fact that the quality attributes of the corresponding functional dependency have not been yet

identified. Since the process of knowledge extraction both in requirement and process engineering is usually iterative, this approach can guide modelers in better clarification of the complexities of a subject domain.

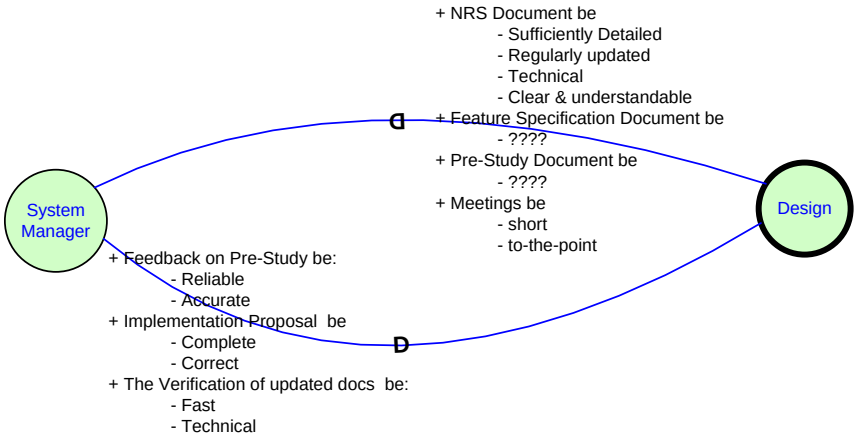

Figure 2: Sample QISD, representing the quality attributes that are important on dependency relations of SA:Designer and System Manager

4 Conclusions

This visual design of ISDs greatly reduces the number of lines on the dependency diagram. The itemized text block of dependums can be easily edited or added to. The main drawback is that the block of dependums is now visualized as a single unit, while semantically each item should be treated as independent. It is therefore harder to visualize redirecting one of the dependums to a different actor, e.g., in cases where a mistake was made, or when considering alternative configurations during process redesign. Further, it is no longer possible to interleave dependency links going in opposite directions to group related links together. One possible solution to this limitation is the use of tables, instead of text blocks on dependency links. In this way further information about dependums (e.g. their types) can be expressed.

We used the ISD models for the early phase on a Software Process Improvement (SPI) initiative in one of the R&D units of Ericsson in Italy. The primary objective of that phase of the SPI initiative was to identify the problematic issues of the current process, in order to come up with proper solutions in the later phases of the initiative. To achieve this goal, we conducted two rounds of interviews with operative and managerial personnel of that R&D unit. In the first round of interviews, we asked the interviewees to describe their role and responsibilities in the organization, and explain different kinds of collaborations that they had with other organization role. After the first round of interviews we gained an initial understanding of the unit, thus we started to develop FISD and QISD models. During the second round of interviews, we used

the initial models and asked interviewees to complete the initial models. Using the ISD models for structuring the interviews we could guide the interviewees to:

1. Visually observe their collaborations with other organizational actors, in terms of the mutual dependency relations.
2. Validate our initial understanding of dependency relations in that R&D unit.
3. Express the functional or quality dependencies that they did not expressed during the first round of interviewees.
4. Identify process problems, which were due to the quality attributes associated with functional dependencies, and were not expressed at the first meetings.
5. Identify process problems, which were due to missing dependencies (i.e. dependencies that should have been exit for facilitating the work).
6. Identify process problems, which were due to unnecessary collaborations and dependencies.

5 Ongoing and future work

As the ongoing project we are still involved in the SPI initiative. We have almost completed the preliminary stage of this initiative, and with the help of ISD models gained a good understanding of the collaboration complexities of the R&D unit. We are going to integrate the information we collected from different Viewpoint Actors, and build a set of comprehensive dependency models that represent the as-is dependency structure of the R&D unit. These models will be used to explicitly represent the hidden or unnecessary complexities, which reduced the productivity of that unit.

As of a future work, we are going to integrate this modeling approach as part of a method engineering framework introduced in [2]. Besides, we are working on a comprehensive SPI framework, which is based on the intentional aspects of development processes, and works with regard to the functional and quality goals of software processes. We planned to use ISD models to represent new dependency relations that will be proposed as the to-be process in an SPI initiative. The models in this paper were developed using Microsoft Visio. We hope to extend the OpenOME in near future to support ISDs as well.

Acknowledgment. We would like to greatly appreciate the constructive comments of reviewers of this paper.

References

1. E. Yu. *Towards modelling and reasoning support for early-phase requirements engineering*. In Proceedings of the Third IEEE International Symposium on Requirements Engineering: (received "Most Influential Paper After 10 Years Award" at RE'07) (1997)
2. H. Chiniforooshan Esfahani and E. Yu. *Situational Evaluation of Method Fragments: an Evidence-Based Goal-Oriented Approach*. In 22nd International Conference of Advances Information Systems Engineering (CAiSE'10). Tunisia (2010)

From Business Services to Web Services: an MDA Approach

Hugo Estrada [1], Itzel Morales-Ramírez [2], Alicia Martínez [1], Oscar Pastor [3]
[1] CENIDET, Cuernavaca, Mor. México
{hestrada, amartinez}@cenidet.edu.mx
[2] Universidad Veracruzana, Xalapa, Ver. México
itmorales@uv.mx
[3] Technical University of Valencia, Spain
opastor@dsic.upv.es

Abstract. At present, enterprises need sophisticated software applications to sell and promote their products or services in order to maintain their leadership in the business world. One of the most promising trends is the Web services technology as the appropriate mechanism to implement e-business, which uses Internet to replicate services offered by an enterprise. However, despite the clear advantages of Web services, there are problems in determining the initial functionalities required by them. Currently, Web services functionalities are not obtained in a systematic manner from the organizational environment. Therefore, it is complicated to ensure that Web services fit the business user's needs. In this paper we face this problem by defining a methodological approach to generate Web services from organizational descriptions. The Model-Driven Architecture has been used in this work in order to ensure the systematic translation of modeling primitives of the organizational model into their corresponding WSDL services descriptions.

Keywords: business service, Web service, Model-Driven Architecture.

1 Introduction

Internet has become an essential tool in current enterprises' activities [1], where it has been used as a successful strategy to guarantee their competitiveness in order to maintain their leadership. In this context, e-business has been implemented as a mechanism that allows the enterprises to offer their products and services remotely, so they can expand their scope previously limited by the location. E-business approach focuses on determining the processes needed by a company to offer their services through the Web. In this context, the Web services are a key technology for an effective operation of e-business systems.

Web services are very useful in the enterprise context because they allow interoperability between software applications. The Web services can afford the evolution of processes that use the same data while merely changing the implementation. This flexibility feature allows diversity in "how" something is done.

However, not all problems involved in developing Web services have been properly solved. One of the main issues in defining Web services is the difficulty to determine what should be the expected functionality of such services. This problem arises from the following factors: 1) the current technology in Web service modeling focuses on defining its functionality without considering, in a systematic manner, the main needs of the organizational context; 2) the lack of reliable sources that allow the designers to implement a Web service reflecting the business tasks as well as the user's requirements; 3) the need of some mechanisms to establish the correspondence between the business functionalities and those which have been implemented in Web services.

We consider the most important of the current difficulties to specify the correct functionality of Web services to be the following: a) business models are not properly adapted to support service-oriented specifications and the result is the incompatibility between these models and those that implement the Web services; and b) there is a lack of methodological approaches to automatically generate services from the business's features which results in this process often needing to be accomplished manually. Therefore, we can conclude that Web services require the establishment of a good specification of the processes that are involved in the enterprise context.

In this research work, a methodological approach is proposed to generate WSDL (Web Services Description Language) descriptions, which are obtained from service-oriented business models. The source model [3] is at a high level and deploys the services offered by an enterprise to customers. It is important to point out that service-oriented approach has been developed over the *i** Framework by adding new rules and properties to get an organizational vision of services. This is the reason why its notation uses the concept of dependency to indicate that a service helps to satisfy the goals of a user. Also the more detailed view of services is represented using an extension of *i** primitives.

As a contribution, this work established an approach of generation of services where the business model is correctly adapted to the service technology, and the Web services are obtained in a systematic way using the MDA (Model Driven Architecture) approach [6].

2 Objectives of the research

The main objective of this work is to obtain the WSDL specifications of Web services from service-oriented organizational models [3]. It is important to point out that several research works exist which discuss the creation of Web services from organizational models [4] [5] [8], however, the research work presented in this paper represents the first approach that proposes the use of the MDA standards to translate service-oriented business models into Web Service descriptions. It is also important to comment that in most of the related works [4] [5] [8] [9] we found the following issues: 1) the Web service modeling is not considered within the organizational environment; 2) some of them have not applied the MDA approach; or 3) the works have not carried out a generation of the WSDL specification.

The proposed methodological approach consists of three phases which are summarized below and depicted in Figure 1:
a) The MDA approach imposes a restriction to transform models; it consists in achieving the transformation using models based on MOF (Meta-Object Facility) specifications [7], and using well-defined models to ensure a systematic generation when applying the transformation rules from a PIM (Platform Independent Model) source model to a PSM (Platform Specific Model) destination model. In this phase we have defined a metamodel for the business service model proposed in [3], the MOF-Ecore specifications were applied in order to create this metamodel.
b) Once the metamodel was defined, the standard of the Object Management Group (OMG), MOFScript, is used to establish the transformation rules M2T (Model to Text), to allow the definition of transformations between a business environment and the implementation of Web services.
c) Finally a software tool, called MOS Tool, uses the rules in MOFScript to generate the WSDL documents (PSM). First, the business service model (PIM) is created and used as a guide for designing the Web services functionalities. The service model is designed with the structure of business service metamodel (a), and then the rules (b) are applied to create the WSDL documents.

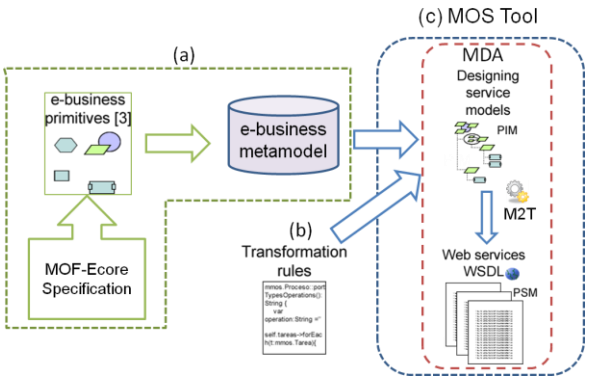

Fig. 1. Overview of the proposed approach

3 Scientific contributions

A service-oriented metamodel: One of the main contributions was the creation of an e-business metamodel (see Fig.1) which was obtained by applying the MOF-Ecore specifications [2] on the primitives of the service-oriented business model [3]. Table 1 shows the relationship among the modeling primitives, their attributes and the attributes description. It was not simple to establish the same level of correspondence between Web service and the service element of the business service model, due to the high level of abstraction in the business model. To match the Web service with the business elements, we analyzed the similar functionalities of them, and we found that a Web service corresponds to the process element; it represents correctly the Web

service while the service element represents the orchestration. The metamodel, called MOS Ecore, being based on MOF, ensures the compatibility required to be stored in MOF repositories and handled through the MOF tools. Using the model, the Web services are created inside the organizational context, and *.mos files are generated.

Table 1. Description of e-business primitives.

Business Service models [3]	e-business primitives	Attributes	Attribute description
Global model, composite service model	Aggregated service	Execution order	Participation order in business model
		Description	Details of the Service's offered
	Basic service	Execution order	Participation order in orchestration
		Description	Service's details offered
Process model	Process	Transaction	Indicates if it will be deployed as a Web service
		Execution order	Participation order in orchestration
		Description	Process's details about what is done
Protocol model	Task	Transaction	Indicates if it will be deployed as an operation
		Execution order	Participation order in orchestration
		Type	Resource's type generated
		Description	Task's details about the atom activity
	Resource	Type	Resource's type to be used by a task
		Description	Resource's details about how is used by a task

Business models to Web services transformation rules: Another contribution is the definition of the transformation rules in MOFScript which are necessary in order to obtain the mapping from a MOS model to the WSDL description of Web. The rules take the *.mos files as input models to generate WSDL specifications as output.

Service modeling tool. MOS tool was developed using NetBeans IDE 6.5 as a mechanism to validate the proposed approach. It has the following features: 1) it allows for creating and opening service models; 2) it serializes the MOS model to XML; 3) it stores the models as files with extension *.mos and; 4) it executes Eclipse V3.3.2 to apply the transformation rules using the MOFScript plug-in. Once the WSDL document is created, it just needs to be checked in well-formed XML syntax.

The only property that needs to be changed is the address where the implementation is located.

4 Conclusions

In this paper, a methodological approach is proposed to generate Web services from a business service model. We propose that by using a service-oriented model at an organizational level, it is possible to facilitate the work of defining Web services under a methodological approach such as MDA.

5 Ongoing an future work

In future work, we will be dealing with the generation of complete functionality of the Web service. We are currently working on methods to use the organizational descriptions (business service models) to generate precisely the choreography and orchestration of services, using the BPEL language. Another proposal for future work is the extension of the MOS Ecore metamodel to integrate all the modeling stages proposed by the business service architecture [3].

References

1. Bieberstein Norbert, Bose Sanjay, Walker Lance, Lynch Angela: Impact of service-oriented architecture on enterprise systems, organizational structures, and individuals. IBM System Journal, 691-708 (2005)
2. Budinsky Frank, Steinberg David, Merks Ed, Ellersick Raymond, Grose Timothy J.: Eclipse Modeling Framework. Addison-Wesley (2003)
3. Estrada Esquivel Hugo: A service-oriented approach for the *i** framework. PhD Thesis, Valencia University of Technology, Valencia, Spain (2008)
4. Grønmo Roy, Skogan David, Solheim Ida, Oldevik Jon: Model-driven Web Service Development. Idea Group Inc. International Journal of Web Services Research, Vol.1, No. 4, 1-13 (2004)
5. Lau Diana, Mylopoulos John: Designing Web Services with Tropos. In: Proceedings of the IEEE International Conference on Web Services (ICSWS'04). San Diego, California, USA (2004)
6. OMG: MDA Guide Version 1.0.1. Specification, Needham, MA, OMG (2003)
7. OMG: Meta-Object Facility (MOF) Specification Version 1.4. Needham, MA, OMG (2002)
8. Papazoglou Mike P., Yang Jian: Design Methodology for Web Services and Business Processes. In: A. Buchmann et al. (eds.) TES 2002. LNCS, vol. 2444, pp. 54-64, 2002. Springer, Heidelberg (2002)
9. Vara Juan M., De Castro Valeria, Marcos Esperanza: WSDL Automatic Generation from UML Models in a MDA Framework. International Journal of Web Services Practices, Vol. 1, No. 1-2, pp. 1-12 (2005)

A bit of "Persona", a bit of "Goal", a bit of "Process" ... a recipe for Analyzing User Intensive Software Systems

Chiara Di Francescomarino, Chiara Leonardi, Alessandro Marchetto, Cu D. Nguyen, Nauman A. Qureshi, Luca Sabatucci, Anna Perini, Angelo Susi, Paolo Tonella, and Massimo Zancanaro

Fondazione Bruno Kessler, Povo, Trento, Italy
{dfmchiara,cleonardi,marchetto,cunduy,qureshi,sabatucci,
perini,susi,tonella,zancana}@fbk.eu

Abstract. The centrality of users in the design and development of complex systems, such as service-based applications, calls for new methodologies and techniques to extract and represent user needs and to translate them into real processes.
In this short paper, we describe the integration of concepts and analysis techniques of different approaches, namely Goal-Oriented Requirements Engineering, User-Centred Design and Process-Oriented Modeling, that are being developed in the context of two projects related to Ambient Assisted Living and Internet of Services.

Key words: Goal-Oriented paradigm, User-Centred Design, Requirements Engineering, Business Process Modeling

1 Introduction

The central role of the users in the design and evolution of complex systems, such as service-based applications for the Internet of Services (IOS[1]) or Ambient Assisted Living systems, has been widely recognized in the last years [1, 2]. Thus, to stress this aspect, we refer them as "user intensive" systems.

Goal-oriented Requirements Engineering (GORE) plays a fundamental role in the development of this kind of systems, enabling reasoning about the domain features with the aim of identifying conflicts and of checking for validity of functional and non-functional requirements. This technique has been exploited in the ACube project[2], whose goal is to study technologies for monitoring complex environments that can be applied in areas such as assisted living homes to help personnel, as well as to support the independence and safety of users.

Moreover, the key for the operationalization of goal-oriented system requirements in terms of services is the definition of a set of rules for associating process modeling concepts to goal-oriented ones. This enables designers to trace business

[1] http://www.future-internet.eu
[2] ACube is funded by the Autonomous Province of Trento. http://acube.fbk.eu/

processes of service-based systems back to intentional elements of the user model (e.g, goals, preferences, roles). The experience in this area comes from the Internet of Service project (IOS[3]), whose aim is to push the "Internet of Services" for real services, rather than for software services. Studies in different areas are conducted within this project, such as the area of service usage, representation, engineering, and delivery.

Nevertheless, we experienced that the two approaches, more than sharing the language of goals, require an effective way to center the design on the users of the system. The aim of this short paper is to propose an integrated methodology in which goal-oriented analysis [3], user-center design [4] and process modeling [2] may cooperate for a continuous communication between requirements engineers, stakeholders and designers, thus reducing the risk of misunderstanding the domain, missing important requirements, and resulting in an increase of the final value of the product.

2 From users' needs to requirements

The strength of the Goal-Oriented techniques in modelling the domains can be still enhanced by coupling the engineering perspective with a creative perspective typical of User-Centred Design approaches. The *User Centered Design* exploits a series of well-defined methods and techniques coming from social sciences and psychology for analysis, design, and evaluation technologies. Contextual inquiries, personas and scenarios are widely employed for obtaining a rich picture of the context (organizational, social, physical), and easily communicating it to stakeholders in order to envision acceptable and innovative technological solutions. Additional values emerge from this collaboration at different phases of the process.

Our proposal is to integrate Tropos [3] with user-centred design techniques in order to guide requirements engineering teams toward effective collection of user requirements and the envisaging of the design of complex software system infrastructures, while helping to fill the gap between end users and developers. Basic principles of this integration are: (i) early focus on users, tasks and environment, (ii) the active involvement of users in the design process, (iii) allocation of functions between user and system, (iv) the incorporation of user-derived feedback into system design, (v) iterative design whereby a prototype is designed, tested and modified.

The result of this integration is a process that encapsulates activities from both the two methodologies, promoting a very close collaboration between teams and an easy exchange of data. The process begins with the *investigation of the domain* in order to understand the organizational setting of the domain and to derive possible needs and services that the system could provide to users. The contextual inquiry produces a rich collection of data in a narrative format. The *data interpretation* phase provides the necessary abstraction to create a

[3] A joint Research project at FBK IRST-CIT. http://se.fbk.eu/en/node/15

believable model of the domain, but avoiding to loose important details typical of a narrative analysis. Tropos early requirement plays a central role at this step by providing the semi-formal language for describing the domain. The *data consolidation* filters data to focus on relevant characteristics: activity scenarios and personas allow designers to have insight the system, providing an anchor to the real domain and the real users. Finally, the *envisioning phase* lets the analysis team to reason on the system-to-be in order to expand designers' prospective, to look at the problems from different points of view, to figure out how their ideas can work in a real context, to identify design criticalities, and finally, to generate requirements. Brainstorming and other creative techniques are used with the result to shift from the Tropos early requirement phase toward the late requirement phase, thus obtaining a list of requirements for the system. The process terminates with the validation of requirements with customers, essential for evaluating the value of the services the system will provide [1].

The key for the integration is the use of *Scenarios* and *Personas* along the whole process. Whereas the use of Scenarios in RE is pretty established as an instrument to describe instances of behavior of the system, Personas — from social science — is still going to be consolidated in RE. Their conjunct usage may increase the ability to envision the system [5], to identify requirement problems and exceptional cases [6] and to help in discovering system functionalities. In particular, Personas are powerful instruments for creating descriptive models of system-to-be users based on behavioral data, gathered from many actual users encountered in ethnographic interviews [4]. Personas' descriptions contain the empathy with users and their personal motivations within a scenario; the cognitive and emotional dimensions are important factors since they help the designer to take decisions in the design process.

In our integrated process, scenarios are stories about people (personas) performing activities; they describe a context in which personas act with the aim of summarizing, clarifying and reasoning about the collected information. The aim is the validation with stakeholders and the technical team. They are presented as narrative or visual stories easy to understand even for non technical people [6].

Among the possible limits of the approach, there is the need of keeping the huge amount of data, usually collected by contextual inquiry and generated by the scenarios-based design, always synchronized with the Tropos diagrams; this also represents a challenge from a theoretical and practical point of view. In the future work, we intend to refine the approach and to investigate the requirements of a tool that support a multi-disciplinary team in this respect.

3 Aligning Goal and Process Models for Real Services

A real service is a combination of actual services and software services that provide electronic access to and monitoring of the actual services [7]. An example of real service is the application that we use when we plan to attend a concert at "Arena di Verona": its realization involves both software services (e.g., the on-

line ticket booking), and actual services (e.g., the actual transportation service needed for reaching Verona).

In real services the user plays a key role: she expresses preferences (e.g., a cheap transportation mean); the service is adapted to her needs (e.g., if she has a meeting when her train should leave, a different actual transportation service is used); the real service is composed taking into account her perspective (e.g., the travel and the concert performance are part of the same real service, though in reality they are independent). Moreover, real services are context-aware, which may result in instant changes and timely responses of the user. Hence the user's preferences, needs and decisions guide the composition of real services, and they, in turn, impact and modify the user's assets (e.g., user's money, user's agenda). For example, buying the ticket for the concert, decreases the user's money.

Due to the "user intensive" nature of these systems, our high level purpose is to stress the centrality of the user along their life cycle. Hence, in this phase, our goal is to move from the user centred requirements (elicited as described in the previous Section) to the system design and validation, while preserving the central role of the user (and her assets). In detail, our work aims at defining a modeling framework that integrates the goal-oriented paradigm (specifically Tropos modeling methods [3]) and process modeling (in particular BPMN [8]), enabling the designers to capture the intentional elements of the user (e.g., goals, preferences, roles, assets), as well as the operational aspects of the real service (e.g., its control flow description, the effect of business activities on user assets). The modeling framework, moreover, includes an ontology for representing user assets and asset modifiers (i.e. activities characterizing the real service that can modify the value of a user-asset), thus capturing their semantics and making them available for supporting the system realization/execution (e.g., the service composition according to the user needs/preferences).

More precisely, the framework rests on the following iterative steps for designing a real service [2]:

- Ontology construction. An ontology modeling *user-assets* and *asset modifiers*is defined, thus capturing the semantics of the concepts, their relationships and constraints.
- Goal model construction. It starts with the analysis of the domain involved in the target system and the system requirements (from the previous requirement phase) and results in the definition of a goal model of the target system.
- Business process model construction. The process model of the target system is defined, deriving part of the information from the goal model. The process is described in the BPMN language and it details the process realization in terms of relevant activities and their execution control flow.
- Dynamic semantics definition of process activities. The dynamic semantics of the asset modifiers is defined, thus allowing to capture the impact of such activities on the user-assets.
- Process model annotation. The generated process model is enriched with semantic annotations [9] taken from the ontology modeling the asset modifiers.

In [2] we illustrated the proposed design methodology along an exemplar case-study. In our future work we intend to extend the modeling framework in the following directions: (i) adding scenarios to goals in the goal model in order to have a more detailed and precise description of goals, thus allowing to better reason about the domain and its concrete operationalization; (ii) further investigating how user preferences (expressed as soft goals in the goal model) affect the user assets, thus allowing to reason about "good" alternatives, to be possibly suggested to the user, during the system process execution; (iii) combining the business process semantics with the dynamic semantics of user assets, thus supporting the reasoning on feasible executions and possible user recommendations. Moreover, the design phases will be complemented with early validation and implementation phases. Early validation will be realized using a simulation system in which runs of real services, events, and operating contexts are simulated for testing purposes.

References

1. Leonardi, C., Sabatucci, L., Susi, A., Zancanaro, M.: Ahab's leg: Exploring the issues of communicating semi-formal requirement to final users. In: Proceedins of CAiSE 2010. (2010)
2. Marchetto, A., Nguyen, C.D., Francescomarino, C.D., Qureshi, N.A., Perini, A., Tonella, P.: A Design Methodology for Real Services. In: 2nd International Workshop on Principles of Engineering Service-Oriented Systems, PESOS, May 2-8, 2010, Cape Town, South Africa. (2010)
3. 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)
4. Cooper, A., Reimann, R., Cronin, D.: About face 3: the essentials of interaction design. Wiley India Pvt. Ltd. (2007)
5. Rolland, C., Salinesi, C.: Supporting Requirements Elicitation through Goal/Scenario Coupling. In: Conceptual Modeling: Foundations and Applications, Springer (2009) 416
6. Sutcliffe, A., Maiden, N., Minocha, S., Manuel, D.: Supporting scenario-based requirements engineering. IEEE Transactions on Software Engineering **24**(12) (1998) 1072–1088
7. Pistore, M., Traverso, P., Paolucci, M., Wagner, M.: From Software Services to a Future Internet of Services. In: Towards the Future Internet - A European Research Perspective. IOS Press (2009) 183–192
8. Business Process Management Initiative (BPMI): Business process modeling notation: Specification. `http://www.omg.org/spec/BPMN/1.2/` (2009)
9. Di Francescomarino, C., Ghidini, C., Rospocher, M., Serafini, L., Tonella, P.: Reasoning on semantically annotated processes. In: International Conference on Service-Oriented Computing. (2008)

Using Intentional Actor Modeling to Support the Evolution of Enterprise Software Architectures in Organizations

Daniel Gross, Eric Yu

Faculty of Information, University of Toronto, Canada

Abstract. The development and evolution of enterprise-wide software architectures is influenced by multiple stakeholders and decision makers in development organizations. Architectural design and evolution is embedded in a larger distributed network of organizational participants, who actively contribute to the identification, interpretation, delegation, reasoning, and enacting of decision processes. In this paper we argue that to adequately support architectural design and evolution in development organizations, it is necessary to deal with the inherently distributed and interrelated nature of organizational decision making in development organizations. This paper proposes treating architectural design and evolution as a distributed and interconnected decision process among organizational actors. The utility of the proposed approach is explored through a pilot study at an insurance company during an enterprise SOA evolution effort.

1 Introduction

Architectural decision making in software development organizations occurs within a larger context of organizational decision making. Multiple stakeholders, occupying various positions in the development, client and other third party organizations, are involved in decision making that directly or indirectly influences architectural development and evolution. For example, upper management scans and interprets the organizational environment, such as current customer demands, future market opportunities and the like, and decides on current and future strategies and goals of the organization. Upper management then hands off enterprise strategies to other decision making stakeholders in the organization, such as product management to exercise their know-how in turning strategy into product approaches. Similarly, upper management and product management rely on enterprise architects to develop architectural principles and guidelines for a sound enterprise-wide architecture that supports current and future strategic goals.

Decision making in development organizations is thus a distributed and interconnected phenomenon, in which upstream management decisions have influence on downstream architectural decision making, and where downstream decisions influence upstream goal achievement. An architectural modeling and analysis approach that supports architectural design reasoning and decision making

needs to support representing, capturing and analyzing the distributed and interconnected decision processes in development organizations[1].

2 Objective of Research

The objective of this research is to explore techniques for supporting enterprise architects in capturing, reasoning about and communicating the architectural design goals, principles and guidelines in the context of other organizational decision processes in the development organization. More specifically, this research proposes applying and adapting agent and goal modeling and analysis techniques to support enterprise architects in:

1. representing, capturing and analyzing design reasoning and decision making of designers and stakeholders from different contrasting viewpoints, such as from a component designer's point of view, and from the enterprise architect's point of view;
2. identifying higher level stakeholders and decision makers whose goals, priorities and choices give rise to, and influence the lower level goal and design reasoning and decision making of designers and stakeholders;
3. systematically analyzing how goals and design approaches compare and contrast when analyzed from different points of view and in relation to their links to goals, prioritizations and tradeoff making of higher level stakeholders in an organization

3 Scientific contribution

The main contribution of this research is the application and adaptation of i* [2] to represent, capture and analyze distributed architectural design and decision making in an enterprise development organization at different organizational levels such as the management level, the enterprise architecting level, the enterprise system architecting level and the component designer level; and the exploration of the utility of the proposed approach during a study at an insurance company (to simplify in this paper we use the terms agent and actor interchangeably).

While i* was originally proposed in the context of early requirements engineering, it has also been applied to representing, capturing and analyzing software architecture [3-5]. These approaches have focused on mapping agent and goal-oriented concepts to concepts in the software architecture domain. In these works, agents directly map onto components, dependencies across agents usually capture some quality constraints across interactions of components, and architectural design reasoning and alternatives are captured outside of agents, thereby assuming an anonymous and global designer of the architecture and all components. This underutilizes the i* approach given its ability to deal with distributed intentionality, autonomy and decision making in organizational settings [2, 6].

Furthermore, existing approaches do not link architectural design to relevant business and management decision making stakeholders in the development organization, or to client and third party organizations. The novelty of this work is that it views the representing, capturing and analyzing of such distributed intents and decision making across all such stakeholders in organizations as an essential part of the development and evolution of software system architectures.

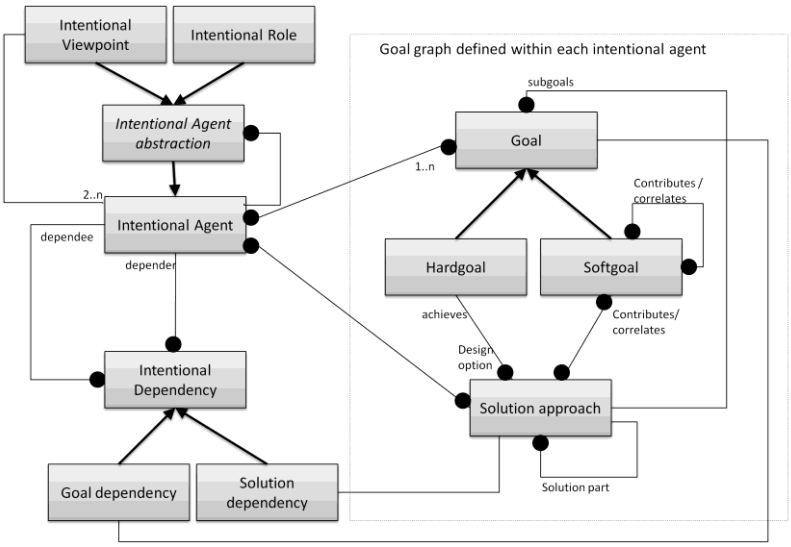

Fig 1: Modeling concept overview

This work can be seen as a significant extension of some early works of the authors [7, 8] in which stakeholders in development organizations contributed intents towards a design team involved in architectural reasoning and decision making. Here the work extends these into dealing with distributed business and architectural decision making in development organizations.

This work also extends i* with *intentional viewpoints*, a novel intentional actor type, and an adaptation of the intentional role concept. Different intentional roles support capturing the reasoning about different non-overlapping design responsibilities. Intentional viewpoints adapt the intentional role concept to reasoning about overlapping design responsibilities. This is analogous to the use of (non-intentional) viewpoints [9] in the usual non-intentional settings, where multiple model fragments are created by different modelers to describe the same phenomenon of interest.

Overlaps between intentional viewpoints are identified when they include overlapping hardgoals (see figure). Given the usual informal design discussions in organizations, is it however difficult to provide clear rules to determine when two hardgoals overlap. Instead, we propose some general guidelines. Given two hardgoals g1 and g2: we consider the goals overlapping if g1 in some way implies by g2. We

suggest looking at the following (non-exhaustive) list of criteria: if g1 is an instance of g2; if g1 is a subtype of g2, if g1 is part of g2; if g1 is an implementation of g2. In the case study we observed that architects and designers adjusted their terminology for each other during discussion. We therefore used the same hardgoal when representing each intentional viewpoint.

The intentional viewpoints concept illustrated in this research captures design reasoning, while non-intentional viewpoints present different design perspectives, which is the result of decision making. Such viewpoint models do not include the design intents and reasoning that lead to design outcome (hence non-intentional models).

4 Conclusions

The utility of the proposed approach was observed during a pilot case study at an insurance company. Intentional viewpoints were used to clearly show and contrast the reasoning of two stakeholders – the SOA architect on the one hand, and the designer of the "Consumer" component on the other. The Consumer component designer is concerned about local design simplicity and maintainability of the consumer component, whereas the SOA architect aims to achieve maintainable, evolvable and scalable enterprise systems, being responsible for the broader enterprise-wide design problem. By placing these intentional viewpoint models side by side the designers felt they were able to communicate better with each other over the design issue at hand. Furthermore, when management issues and reasoning where included (in the form of a network of "management" agents), linking them to the intentional viewpoints of the enterprise architect and the component designer, software design reasoning could be understood within the context of higher level management strategies and prioritizations. These linkages were also considered useful by stakeholders at the study site and were seen as contributing to SOA governance within the Enterprise.

5 Ongoing and Future work

The next steps in this ongoing pilot study are to further analyze design discussions and extend the modeling technique where appropriate; to identify modeling simplifications so that detailed agent and goal oriented design modeling and reasoning can be distilled to a few representative agent and goal elements specifically adapted to the explanatory needs of different types of stakeholders (maintainers, designers, reuse managers, middle and upper management, etc.); to put these simpler models to test in communicating and explaining discussion points with designers and stakeholders; and to obtain relevant feedback and, where necessary, ideas for improvements.

Technical features being explored include developing a systematic approach to compare and contrast the reasoning captured in different intentional viewpoints, while taking into account the different scopes and levels of abstraction each intentional viewpoint may present; the inclusion of different intentional agent types, such as

intentional roles and intentional positions, as well as other knowledge structuring mechanisms across agents, in particular inheritance and instantiation linking between intentional agent types [2], and how these are combined with agent types to support dealing with larger scale capturing and documenting of architectural decision discussions and decision making in development organization; the integration of intentional architectural agent modeling with non-intentional architecture modeling approaches, and the representation of relevant non-intentional knowledge in a notation neutral manner [10]; and appropriate tool support in enterprise organization settings.

References

1. Curtis, W., et al., *On building software process models under the lamppost.* Proceedings of the 9th international conference on Software Engineering, 1987: p. 96--103.
2. Yu, E., *Modeling Strategic Relationships for Process Re-Engineering*, in *Department of Computer Science*. 1994, University of Toronto: Toronto.
3. Gross, D. and E. Yu, *Dealing with system qualities during design and composition of aspects and modules: an agent and goal-oriented approach*, in *Proceedings of the 1st International Workshop on Traceability in Emerging Forms of Software Engineering*. 2002a.
4. Kolp, M. and J. Mylopoulos. *Software Architecture as Organizational Structures*. in *Proceedings ASERC Workshop on "The Role of Software Architectures in the Construction, Evolution, and Reuse of Software Systems*. 2001. Edmonton, Canada.
5. Grau, G. and X. Franch, *On the Adequacy of i* Models for Representing and Analyzing Software Architectures.* Proceedings of the First International Workshop on Requirements, Intentions and Goals in Conceptual Modeling (RIGiM'07), 2007: p. 296-305.
6. Yu, E., *Agent Orientation as a Modelling Paradigm.* Wirtschaftsinformatik, 2001. **43**(2): p. 123-132.
7. Chung, L., D. Gross, and E. Yu, *Architectural design to meet stakeholder requirements*, in *Software Architecture*, P. Donohue, Editor. 1999, Kluwer: San Antonio, Texas, USA. p. 545-564.
8. Gross, D. and E. Yu, *Evolving System Architecture to Meet Changing Business Goals: an Agent and Goal-Oriented Approach*, in *Proceedings of the First International Workshop From Software Requirements to Architectures (STRAW 2001) at the International Conference of Software Engineering*. 2001: Toronto, Canada.
9. Nuseibeh, B., J. Kramer, and A. Finkelstein, *A framework for expressing the relationship between multiple views in requirements specification.* IEEE Transactions on Software Engineering 1994. **20**(10): p. 760-773.
10. Gross, D. and E. Yu, *Resolving artifact description ambiguities during software design using semiotic agent modeling*, in *12th International Conference on Informatics and Semiotics in Organisations*. 2010: Reading, UK.

Bridging the Gap between Goals, Agents and Business Processes

Renata S.S. Guizzardi, Giancarlo Guizzardi,
João Paulo A. Almeida, Evellin C. Cardoso

Ontology and Conceptual Modeling Research Group (NEMO)
Federal University of Espírito Santo
Av. Fernando Ferrari, S/N, 29060-970, Vitória/ES, Brazil
[rguizzardi,gguizzardi,jpalmeida]@inf.ufes.br, evellinc@gmail.com

Abstract. Organizational Modeling is a discipline which tries to capture and reason about the distinct dimensions (e.g. structure, strategies and processes) involved in organizations by the means of visual models. In order to be effective, these models must represent in an abstract way, the right set of concepts composing each of the organizational dimension. Our work focuses on identifying and understanding this set of concepts through a foundational ontology. Moreover, we aim at investigating different modeling languages, identifying if (and to what extent) each of them, individually or in combination with one another, adequately covers this set of concepts. In this article, we discuss our work on the combination of *i**/Tropos (representing a goal modeling dimension) with approaches representing the agent-oriented organization and business process domains. Finally, we elaborate on case studies and computational support for the methodologies originated from the combination of these languages.

Keywords: organizational modeling, goals, agents, business processes, foundational ontologies.

1 Introduction

Mainly aiming at staying in business or seeking for higher profits, organizations today need support for fostering innovation and boosting production. This leads to efforts in different directions, promoting, for instance, organizational *reengineering*, in order to improve the way products and services are delivered, and *knowledge management* to keep a constant flow of usable knowledge throughout the organization's points of action. Both for reengineering and knowledge management, it is crucial that organizations develop a deeper understanding regarding their different dimensions, such as *structure*, *strategies* and *processes*. Such an understanding can emerge through *Organizational Modeling*, a discipline which tries to capture and reason about these distinct dimensions by the means of models. In order to be effective, these models must represent in an abstract way, the right set of concepts composing each of the organizational dimension. Our work focuses on identifying and understanding this set of concepts. Moreover, we aim at investigating different modeling languages,

identifying if (and to what extent) each of them, individually or in combination with one another, adequately covers this set of concepts.

In [10], we proposed to combine *i**/Tropos with another agent-oriented approach named AORML, so as to result in a thorough methodology to analyze and design agent-oriented knowledge management systems. The idea was to apply *i**/Tropos as an organizational modeling approach to diagnose what kind of support an organization needs to enable knowledge creation and sharing. And then, use AORML to design a system to support these processes.

However, fostering innovation does not necessarily involve a supporting system. Many times, this can be achieved by changing the practices and processes adopted by the organization. This brings us to the area of business process engineering, which focuses on a detailed understanding of the chain of activities that deliver the organization's products and services. However, the existing business process modeling languages stress the temporal order of activities, giving only marginal attention to the strategic dimension (i.e. goals) that motivates these activities to be executed. For instance, the modeling language used in ARIS, the most prominent business process modeling framework, from an industrial point of view, offers a very simple syntax for modeling goals. This syntax basically allows the identification of a few goals and subgoals, connecting them to macro-processes, without supporting in depth analysis, such as *i**'s alternative and contribution analyses. Our current work investigates how to relate goals and business processes by combining *i**/Tropos to ARIS EPC (Event-driven Process Chains), ARIS`s syntax to model processes [1,2].

It is also important to state that both for combining goals and agents and for integrating goals and business processes, we adopt an *ontological approach*, as argued for in this same event two years ago [9]. Foundational ontologies have been proven to create a safe path for (re)engineering consistent and coherent conceptual modeling languages. We hereby rely on a foundational ontology named UFO [8,7], which guides us in the alignment of *i**/Tropos with different approaches. In fact, the utmost goal of our work concerns this ontology, as our research group aims towards the investigation of "the ontological nature of the social entities underlying the agent-oriented modeling paradigm. By doing this with the help of an interdisciplinary approach, we aim at defining a **stable** and **sound formal theory** which can be used as a foundation for agent concepts" [9].

The remaining of this paper states the objectives of our research (section 2), the main scientific results achieved by this work (section 3), conclusions (section 4) and future work (section 5).

2 Objectives of the Research

Our research objectives comprise:

1. Evolving the theoretical foundation for agent-oriented, process-oriented and goal-oriented paradigms and applying this theoretical foundation to analyze, evaluate and integrate conceptual modeling languages.

2. Investigating the relations between the goal domain, the business process domain and the (agentive) organizational structure domain with the purpose of improving the modeling of the organizational strategic dimension.
3. Developing model-driven methodologies, which relies on the combination of existing works and on the evolution of existing solutions for automated support.
4. Applying the resulting methodologies in case studies with the purpose of validating them in practice.

3 Scientific Contributions

The subsections in the sequel bear a correspondence (in a reverse order) to the objectives enumerated in section 2. Due to lack of space, we have decided not to include here a discussion regarding objective 1, namely, the ontological theories providing foundations for our work. Aside from space limitation, the ontological theories themselves as well as their applications are more general than the scope of the workshop. Recent publications related to these theories as well as their applications can be found, for example, in [5,6,7] and [3,4], respectively. However, because these theories crosscut and support the remaining objectives, their role w.r.t. to each of these objectives is discussed in the corresponding sections below.

3.1 Case Studies

With the purpose of investigating the potential relationships between goals and business processes in a real world organization, we have conducted an exploratory study in a Rheumatology Department of a hospital in Brazil. The result of this case study comprehends a set of goal models in *i**/Tropos, each one directly associated with a business process, also fully modeled in ARIS EPC. Such goal and business process models focus on the organization as it is today (*i** early requirements or AS-IS model, in business process modeling jargon). From the point of view of the department where the study was conducted, this result opens up many possibilities for re-engineering and process automation.

Developing the exploratory study in a real organization has given us the opportunity to test and question many of the techniques generally associated to goal elicitation, such as interviews and active observation. After applying these techniques, we noted that most of the goals had a process-like nature, instead of capturing the intentions behind the tasks of the stakeholders. Moreover, some of the business processes were unrelated to strategic goals, which suggested that a large number of goals had remained unidentified. The solution to this problem involved the application of Non-Functional Requirements (NFR) catalogues. In our case, NFR catalogues are not used in the scope of system development, as in its original proposal. Conversely, it is applied to elicit goals that directly impact the organization's business processes. The application of the catalogues has shown to be very interesting because it enables reasoning about the organization from a more

strategic point of view. This can be confirmed by the elicitation of goals which referred to quality attributes either for the business processes or for the organization as a whole. In that respect, the catalogues employed in this case study provided guidelines for identifying these attributes in a systematic way. The main scientific contribution resulting from this case study is a methodology to elicit goals and business processes [1,2].

We have also conducted a second case study exploring the mutual interaction between goal models in *i**/TROPOS and business process models in ARIS EPC. This second case study took place in the context of a Brazilian (multi-national) large organizational of the energy (petroleum and gas) sector. As discussed in our previous paper [9], it is important that the same business process and its composing activities can be seen at different levels of granularity in different phases of the process, from conceptual modeling to implementation. An example of this situation took place in the aforementioned project. In that case, it was required that a workflow specification should be derived from a large business process model. However, the requirement was to implement a more abstract version of the initial conceptual model, i.e., a version of the latter model captured in a higher level of abstraction. In order to do that, one is required to construct a more abstract version of a process in a bottom-up fashion, i.e., by (among other things) creating macro-activities which will be composed of a number of the original ones. Now, a question begging issue here is: how do decide which activity will be part of which macro-activity? The solution found in that project was to elicit *i**/TROPOS goal models that were decomposed into a level so that each activity in the original process could be associated to a goal. By doing that, we could construct the macro-activities in the more abstract process model by creating a systematic alignment between the goals decomposition structure and the process composition one.

3.2 Relating Goals and Business Processes

As a result of the hospital's case study (section 3.1), we observed that establishing the relations between goals and business process is far from straightforward. This can be accounted by the fact that goals may be formulated at various levels of abstraction and precision. To solve that, we propose using a *Goal Taxonomy* [2] to deepen our understanding about the goal domain, before establishing the relationships between goals and business processes. Goal taxonomies have been applied in system requirements elicitation to guide the discovery of goals and requirements, and their subsequent implementation in the target system. In the scope of BPM, a goal taxonomy is important because the different types of goals impact on the structures of business processes which support them. For example, some goal can be associated with one sole business process in order to be satisfied. Alternatively, another goal requires several business processes to execute simultaneously in order to be satisfied. Our major reason for proposing such classification is to reflect the different ways goals can be satisfied according to their participation in relations with business processes. This was crucial to enable the alignment of goals and business processes.

Moreover, besides understanding the goal domain, other concepts are important to help us align goals and business processes. Concepts such as agents, intentionality,

commitments, among others, also have an impact on how goals and business processes are related. The semantics of these concepts can be well understood with the use of UFO [8,7]. UFO provided us with a common ontological foundation for goals and other enterprise elements, enabling us to understand how these elements relate. The resulting alignment between goals and business processes was only possible due to this understanding.

3.3 Relating Goals and Agents

In [10], we proposed ARKnowD (read "Arnold"), a methodology which combines *i**/Tropos and AORML to develop knowledge management systems. ARKnowD's life cycle is composed of four activities, namely requirements elicitation, requirements analysis, architectural design and detailed design. These activities may be iteratively executed up to the point that the solution is modeled in enough detail to enable implementation. *i**/Tropos is applied in the first three activities while AORML covers the forth one.

Inspired by the Model Driven Architecture (MDA) initiative and guided by the UFO ontology [8], we developed some transformation rules which map *i**/Tropos into AORML. This guarantees a smooth transition from architectural to detailed design, guiding the developer on the use of the methodology, and facilitating automatic model transformation from one activity to the other [11].

Preliminary work has been done on delivering automated support to ARKnowD [10]. By applying metamodel transformation, using our transformation rules, we started to integrate AROML into an *i**/Tropos modeling tool named TAOM4E (http://sra.itc.it/tools/taom4e/). This work allowed an *i**/Tropos actor diagram to be transformed into an AORML agent diagram. We are currently busy to provide transformations from *i**/Tropos's diagrams to the remaining AORML models, so as to deliver a modeling tool which enables full design using ARKnowD. This will also allow code generation using the JADE framework, thus also supporting system implementation. In this context, we are also investigating how to generate, from the AORML model, a database model which can be later transformed into SQL, hence also delivering a database to support the agent-oriented system under implementation.

4 Conclusions

Distinct modeling approaches have been designed over the years and by different communities with the aim to address the different dimensions of organizations, such as structure, strategies and processes. In this paper, we described the objectives and main scientific contributions of our work on offering theoretical support for evaluating and engineering combinations of some of these approaches. Moreover, we briefly discuss the application of these combined modeling solutions in real-world scenarios as well as the development of computational tools to support them.

Acknowledgement. This work is supported by the Brazilian Research Funding Agencies FAPES (process no. 45444080/09) and CNPq (process no. 309382/2008-4).

References

1. Cardoso, E.C.S., Guizzardi, R.S.S. and Almeida, J.P.A.: Aligning Objectives and Business Process Models: A Case Study in the Health Care Industry. Int. J. of Business Process Integration and Management (forthcoming)
2. Cardoso, E.C.S., Almeida, J.P.A., Guizzardi, G. and Guizzardi, R.S.S.: Eliciting Goals for Business Process Models with Non-Functional Requirements Catalogues. In: Proc. of 10th International Workshop on Business Process Modeling, Development and Support, CAISE 2009, vol. 29, pp. 33--45, Amsterdam, The Netherlands (2009)
3. Gonçalves, B.N., Zamborlini, V. and Guizzardi, G. An Ontological Analysis of the Electrocardiogram. Electronic Journal of Communication, Information and Innovation in Health, , vol. 3, pp. 1-26 (2009)
4. Guizzardi, G., Lopes, M., Baião, F. and Falbo, R. On the importance of truly ontological representation languages, Int. J. of Information Systems Modeling and Design, IGI-Global (2010)
5. Guizzardi, G. On the Representation of Quantities and their Parts in Conceptual Modeling, In: Proc. of 6th Int. Conf. on Formal Ontologies in Information Systems, Toronto, Canada (2010)
6. Guizzardi, G. The Problem of Transitivity of Part-Whole Relations in Conceptual Modeling Revisited. In: 21st Int. Conf. on Advanced Information Systems Engineering, LNCS, vol. 5565, pp. 94-109. Springer-Verlag, Berlin (2009)
7. Guizzardi, G., Guizzardi, R.S.S., Falbo, R.A.: Grounding Software Domain Ontologies in the Unified Foundational Ontology (UFO): The case of the ODE Software Process Ontology. In: Proc. of the Iberoamerican Workshop on Requirements Engineering and Software Environments, Recife, Brazil (2008)
8. Guizzardi, G.: Ontological Foundations for Structural Conceptual Models. PhD Thesis, University of Twente, The Netherlands (2005)
9. Guizzardi, R.S.S., Guizzardi, G., Almeida, J.P.A. and Cardoso, E.C.S.: Ontological Foundations for Agent-Oriented Organizational Modeling. In: 3rd International iStar Workshop, Recife/PE, Brazil (2008)
10. Guizzardi, R.S.S.: Agent-oriented Constructivist Knowledge Management. PhD Thesis, University of Twente, The Netherlands (2006)
11. Guizzardi, R.S.S., Guizzardi, G.: From Tropos to AORML, Using a Foundational Ontology. In Giorgini, P., Maiden, N., Mylopoulos, J. and Yu, E. (eds.): Tropos/*i**: Applications, variations and Extensions, Cooperative Information Systems Series, MIT Press (forthcoming)

A Framework for Iterative, Interactive Analysis of Agent-Goal Models in Early Requirements Engineering

Jennifer Horkoff[1], Eric Yu[2]

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

jenhork@cs.utoronto.ca, yu@ischool.utoronto.ca

Abstract. The early stage of domain analysis in requirements engineering is critical for understanding the stakeholders, their needs, problems, and how views of these problems differ. We advocate methods for early domain exploration which provoke iteration over captured knowledge, prompting analysts and stakeholders to review what is known, helping to guide elicitation, and facilitating early scoping and decision making. Specifically, we provide a framework to support interactive, iterative analysis over goal- and agent-oriented (agent-goal) models. The framework will allow for multiple types of analysis questions, manage alternative evaluations over a model, manage interactive results, capture model assumptions and arguments, and support iteration over all constructs. Initial case study experience shows that interactive evaluation provokes model iteration and domain exploration. Further case studies will be developed to test the benefits of framework expansions.

Keywords: Goal-and Agent-Oriented Models, Early RE, Model Analysis

1 Introduction

Early stages of domain analysis (Early RE), as characterized by Yu in [1], are critical for understanding stakeholders, their needs, inherent domain problems, and how views of these problems differ in the eyes of stakeholders. Early stages of analysis are characterized by incomplete and imprecise information. It is often hard to quantify or formalize critical success criteria such as privacy, security, employee happiness, or customer satisfaction in early stages. Early analysis involves a high-degree of stakeholder participation, not only gathering information from individuals using or affected by the proposed system, but presenting information gathered thus far, allowing validation and improved understanding in an iterative process.

If Early RE information is collected in an ad-hoc way it may be difficult to facilitate communication, convergent understanding, and, more importantly, aid the discovery of missing or misunderstood information. We advocate methods for early domain exploration which provoke iteration over captured knowledge, prompting analysts and stakeholders to review what is known, helping to guide elicitation, and facilitating early scoping and decision making.

Approaches have been introduced in order to facilitate elicitation, understanding,

and analysis when dealing with incomplete or imprecise information. For example, the Soft System Methodology is aimed at dealing with systems where objectives are difficult to clearly define and are often conflicting [2]. This approach uses rich pictures to capture the domain. Although the lack of defined syntax for such models allows for flexibility it discourages tool support, including analysis which makes explicit use of model structure and which may encourage model iteration.

Another popular approach for Early RE analysis is the application of Goal- and Agent-Oriented Models (agent-goal models), advocated in [1], where graphical models are created to represent goals and actors in the domain, including their decomposition, contributions, and side-effects. These approaches are applicable to Early RE analysis as they allow users to model fuzzy concepts (softgoals) and can provide useful views even if the models are not complete. However, domain exploration using agent-goal models often stops after a single round of modeling.

Several analysis procedures have been introduced for agent-goal models, employing methods such as the propagation of satisfaction or metrics over model constructs ([3], [4]). These procedures often require precise or specific domain information such as probabilities, costs, priorities, or quantitative estimates from "experts", difficult to acquire in early analysis stages. These approaches are typically fully-automated, "push-button"-type procedures where input is given, the procedure initiates, and an answer or results are provided. We believe that it is difficult for stakeholders to trust results produced automatically over incomplete and imprecise information, especially if the mechanism for deriving results is opaque or mysterious.

What is needed is a way to capture and analyze domain information in Early RE which specifically prompts iteration over domain knowledge, increasing the likelihood of discovering objectives, problems, and alternative remedies in the domain. We are interested in methods which allow interaction, receiving frequent input from stakeholders, but which can be enhanced by tool support. To this end we create a framework for iterative, interactive analysis of agent-goal models in early requirements engineering. Our aim is to expand the capabilities of agent-goal modeling in the following ways: allowing for multiple types of iterative analysis over models; supporting management of alternative solutions in the model; supporting management of user-entered judgments, assumptions, and rationale; supporting iteration over models and user judgments; and guiding model creation and analysis.

2 Objectives of Research

We aim to support iterative learning and understanding of a domain in the early stages of a requirements analysis project. Previous work has provided evidence that interactive qualitative forward analysis over goal models prompts users to make changes to the model, derive questions concerning the domain, and improve their understanding of the model and its subject matter [5], [6], [7]. We capitalize on these effects by extending this procedure as part of a framework supporting iterative domain exploration. Specifically, we aim to allow for analysis over incomplete and imprecise information, allow for the assessment of stakeholder objectives in light of alternatives, provoke iteration over the model and further elicitation in the domain,

and, overall, increase domain understanding among analysts and stakeholders, helping an organization learn about itself. We claim that accomplishing these objectives will help to ensure captured requirements effectively address problems in the domain, avoiding development of the "wrong" system.

3 Scientific Contributions

We outline components of our interactive framework in the following section. Some components, such as forward evaluation, have been well-described and applied in existing work, while other components are in various stages of development.

Forward Evaluation. An interactive, qualitative forward evaluation for i* models, an expansion of the procedure in [8], has been introduced and is described in [5], [6], and [7]. The procedure starts with an analysis question of the general form "How effective would a proposed solution be in meeting the desired goals?" The analysis makes use of a set of qualitative evaluation labels, assigned to intentions to express their degree of satisfaction or denial. The procedure propagates initial values iteratively from contributing elements to recipient elements through model links using defined rules. The interactive nature of the procedure applies when human judgment, based on domain knowledge, is used to combine multiple conflicting or partial values to determine the satisfaction or denial of a softgoal. An assessment is made as to whether the alternative is satisfactory, stimulating further analysis and potential model refinement. The procedure is currently implemented in the OpenOME tool [9].

Backward Evaluation. In addition to "What-if?" questions, it is useful to support "Is it possible?" questions. For example, "Is is possible for certain element(s) in the model to be satisfied? Answering these questions requires "backward" analysis, where desired values are placed on the model and the procedure works backwards (from recipient elements to contributing elements) to find alternatives in the model which produces these values. Work in [3] has implemented a fully-automated, two-value procedure for non-agent goal models using a SAT solver. We expand on this approach, adapting it to consider agent-oriented concepts, a single evaluation value for each element, and the role of human intervention, producing an iterative, interactive procedure. An initial description of the procedure can be found in [10].

Multiple Evaluations over a Single Model. Experience has indicated that it is useful to store the evaluation results of each alternative, allowing users to flip between views of the alternatives, facilitating a comparison. GRL as implemented in the jUCMNav tool currently allows users to store multiple analysis results; however these results are automatically recalculated when changes are made to the model [11].

Human Judgment Management. It is useful to revisit evaluation judgments for alternatives over a model. Users should be able to see all judgments for a particular element, either specific to an alternative, or across all alternatives.

Assumptions and Argumentation. We would like to capture information, especially domain assumptions and the rationale for evaluation decisions, as part of the modeling process. Modelers should be able to attach assumptions and arguments to parts of the model or to human judgment in evaluation. Work in [12] has used satisfaction arguments to justify the satisfaction of selected i* elements, including

domain assumptions. Our framework will capture arguments and assumptions over more model constructs, incorporating this information into evaluation.

Supporting Model Iteration. Our framework will allow users to make changes to the model, their judgments, and their textual arguments and assumptions. Whenever changes are made, the user will be shown which evaluation results are potentially affected, and will be able to interactively re-evaluate these parts of the model.

Suggested Methodology. We will guide the iterative creation and analysis of Early RE agent-goal models by providing a suggested methodology. An initial version, included in [6], [7], will be expanded to cover all framework components.

The proposed framework will advance beyond current work in several ways:

- Allowing analysis over informal, incomplete, agent-goal models in Early RE without requiring detailed or quantitative information. This goes beyond the algorithm sketch provided by the NFR Framework ([8]) by allowing users more freedom in their judgments and working over agent-oriented syntax.
- Providing interactive forward and backward analysis, letting users make decisions over partial or conflicting evidence. Our previous work in [6] allowed only a single type of analysis and had limited support for iteration.
- Unlike other forward satisfaction algorithms for agent-goal models ([3]), the algorithms are iterative, continually adapting to input provided by users.
- Presenting the partial results of the algorithm to users as they are evaluating a model, helping to promote transparency and buy-in.
- Supporting iteration over the model by showing users what analysis values may be affected by model and judgment changes.
- Providing an incremental algorithm which remembers past states and supports minimum re-evaluation after model or judgment changes.
- Other frameworks have supported management of alternatives [11], storage of assumptions or arguments [12], or supported (automatic) backward evaluation [3], this framework combines these aspects together, allowing complimentary interaction between the features and providing a single implementation.
- Focusing on the iteration and elicitation prompted by analysis through application of case studies.

4 Conclusions, Ongoing and Future Work

The forward procedure component of the framework has already been tested via several case studies, including a demonstration of the differences between proponents and opponents of Trusted Computing Technology [13], and an analysis of an online counseling in a large social service organization, with selected results reported in [14], [15], and [16]. Evaluation over models in both studies demonstrated the ability of the procedure to provoke elicitation and model iteration, as evaluation results sometimes led the modeler to further investigate sources and often to modify the model to more accurately reflect the domain.

Further studies will be performed to test the utility of backward analysis and additional framework components. We plan to use both an action research approach, using the framework to work with an organization and analyze its needs, as well as

individual studies, looking at how users analyze models with and without the framework. Study results should confirm whether the backward procedure also prompts iteration and model improvement.

Future work could investigate extending the framework with varying levels of human interaction, tabular views of model elements, assumptions or justifications, and views which allow comparisons between analysis results over alternatives.

References

1. Yu, E.: Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. In 3rd IEEE International Symposium on Requirements Engineering (RE'97), pp. 226--235. IEEE Press, New York (1997)
2. Checkland, P.: Soft systems methodology: A Thirty Year Retrospective. Systems Research and Behavioral Science 17, (S1) S11--S58 (2000)
3. Giorgini, P., Mylopoulos, J., Sebastiani, R.: Simple and Minimum-Cost Satisfiability for Goal Models. In: Persson, A., Stirna, J. (eds) CAiSE 2004. LNCS, vol. 3084, pp. 20-3-5. Springer, Heidelberg (2004)
4. Franch, X.: On the Quantitative Analysis of Agent-Oriented Models: In Dubois, E., Pohl, K. (eds) CAiSE'06. LNCS, vol. 4001, pp. 495--509. Springer, Heidelberg (2006)
5. Horkoff, J., Yu E.: A Qualitative, Interactive Evaluation Procedure for Goal- and Agent-Oriented Models. In: CAiSE Forum. CEUR Workshop Proceedings (2009)
6. Horkoff, J., Yu, E.: Evaluating Goal Achievement in Enterprise Modeling – An Interactive Procedure and Experiences. In: A. Persson, J. Stirna (Eds.), 2nd IFIP WG 8.1 Working Conference on The Practice of Enterprise Modeling (PoEM'09). LNBIP, vol. 39, pp. 145-160. Springer, Heidelberg (2009)
7. Horkoff, J., Yu, E.: Interactive Analysis of Agent-Goal Models in Enterprise Modeling. Int. J. Inf. Sys. Modeling and Design. (to appear).
8. Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. Kluwer Academic Publishers, Norwell, MA (2000)
9. OpenOME, https://se.cs.toronto.edu/trac/ome/wiki
10. Horkoff, J., Yu, E.: Qualitative, Interactive, Backward Analysis of i* Models. In: 3rd International i* Workshop. pp. 43—46, CEUR-WS.org (2008)
11. Amyot, D., Ghanavati, S., Horkoff, J., Mussbacher, G., Peyton, L., Yu, E.: Evaluating Goal Models within the Goal-oriented Requirement Language. Int. Journal of Intelligent Systems (IJIS). (to appear).
12. Maiden, N.A.M., Lockerbie, J., Randall, D., Jones, S., Bush, D.: Using Satisfaction Arguments to Enhance i* Modelling of an Air Traffic Management System. In: 15th IEEE Int. Con. on Requirements Engineering (RE'07). pp.49--52. IEEE Press, New York (2007)
13. Horkoff, J., Yu, E., Liu, L.: Analyzing Trust in Technology Strategies. In: International Conference on Privacy, Security and Trust (PST 2006). pp. 21--32. (2006)
14. Easterbrook, S. M., Yu, E., Aranda, J., Fan, Y., Horkoff, J., Leica, M., Qadir, R. A.: Do Viewpoints Lead to Better Conceptual Models? An Exploratory Case Study. In: 13th IEEE International Requirements Engineering Conference (RE'05). pp. 199--208. IEEE Press, New York (2005)
15. Strohmaier, M., Yu, E., Horkoff, J., Aranda, J., Easterbrook, S.: Analyzing Knowledge Transfer Effectiveness - An Agent-Oriented Approach. In: 40th Hawaii International Conference on Systems Science (HICSS-40). pp 188b, IEEE Press, New York (2007)
16. Strohmaier, M., Horkoff, J., Yu, E., Aranda, J., Easterbrook, S.: Can Patterns improve i* Modeling? Two Exploratory Studies. In: Paech, B., Rolland, C. (eds) REFSQ'08. LNCS, vol. 5025, pp. 153--167. Springer, Heidelberg (2008)

Automated Generation of Attack Routes for Service Security Analysis - A Preliminary Report

Tong Li[1], Golnaz Elahi[2], Lin Liu[1], Eric Yu[3],

[1] School of Software, Tsinghua University, Beijing, China, 100084
{litong08, linliu}@tsinghua.edu.cn
[2] Department of Computer Science, University of Toronto, Canada
gelahi@cs.toronto.edu
[3] Faculty of Information, University of Toronto, Canada
yu@ischool.utoronto.ca

Abstract. i* modeling has been used to characterize service-oriented computing in terms of intentional concepts such as agents, goals, dependencies, as well as services they provide or consume. The intentional models provide a rich basis for various security related reasoning, such as vulnerability analysis, attack and countermeasure evaluation, risk assessment, etc. In this work, we aim to explore a reasoning method over the i* models that goes beyond evaluating the satisfaction of security properties. We propose a service security modeling approach for automated generation of attack routes against a specific service. We analyze the security level for each service by using the resulting models. We aim to discover countermeasures and incorporate them into the security analysis process.

1 Introduction

The i* framework offers a tool box for modeling social, organizational, or software system agents, their intentions, actions, and dependencies to other actors to achieve their goals. The i* framework does not only provide a modeling notation, but also offers a framework for thinking about systems and services: their goals, network of dependencies, alternative strategies to satisfy the goals, etc. In addition, the resulting models provide a rich basis for various types of reasoning.

The same i* modeling elements can be reused for modeling different conceptual entities such as business goals and processes [1], software services [2], software requirements [3], organizational or social relationships, knowledge entities [4], the law and regulatory [5], security goals [6,7] security vulnerabilities [6], and security attacks [6, 8, 9].

Consequently, the resulting models can be used in numerous different analysis and reasoning approaches. For example, the i* security-related models have been shown (or argued) to be useful for vulnerability analysis in social and organization networks [6], security risk assessment [9], attack analysis [6, 8], security trade-off decision analysis [7], trust analysis in the chain of dependencies [10], etc.

In most of the existing work, a model of the system is developed and certain properties under specific assumptions are checked. Several methods rely on goal

model evaluation techniques, which propagate satisfaction labels through the goal graph to check the ultimate satisfaction status of the goals. We believe the reasoning power over the i* models is not yet comprehensively explored. The i* models (with light-weight security extensions) could be useful for other types of reasoning, beyond checking some security properties. For example, the models supplied with enough security knowledge could be used for discovering security vulnerabilities, attacks, attack routes, critical entities, and countermeasures.

In this work, we aim to take the reasoning over the i* models one step ahead, and discover possible attack routes against a given service in the context of Service-Oriented Architecture (SOA). Before adopting a security mechanism, there are many questions to be answered: what are possible ways that a service can be attacked and whether they can be avoided? Are the risks high enough to adopt defensive techniques? A careful analysis of security threats at the early requirements analysis phases, before design solution is decided, would prevent adoption of (unnecessary) security mechanisms in an ad hoc way.

In this work, we propose a service security modeling framework (SSMF) as well as a reasoning method over the i* model to automatically identify the potential attack routes to a particular service. An attack route includes a path of task decompositions (AND/OR) and delegations to satisfy a top anti-service. In this method, the attacker is treated as an agent in the service environment who can conduct reasoning to meet his goals by using his capabilities. For example, the attacker composes several distributed attack actions to meet his higher level attack goal. Ultimately, based on the resulting attack routes, potential countermeasures are identified. The service compositions are then assessed with the presence of discovered countermeasures. This iterative process of identifying attack routes and countermeasures continues until the risk level of the potential attacks is tolerable. This paper reports on the ongoing work toward the discussed service security modeling framework (SSMF).

2 Research Objective

This work aims to analyze the system from the attackers' point of view and discover possible attack routes in a given service-oriented system. We model services using the i* notation and express capabilities and requirements of each service provider and consumer. Then, a hypothetical attacker is added to the service environment. The attacker has malicious goals which threaten specific services in the service environment. We use rule-based reasoning to assess whether the attacker can achieve his malicious goals. The results of this reasoning would help us to decide if the risks are high and whether security countermeasures are needed to prevent the attacks.

The main objectives of this work are toward two main directions: our first goal is to help security analysts to identify potential attack routes against a particular service. Unlike the traditional risk propagation analysis that focuses on the effects of a specific given risk, our method focuses on identifying the risks that threaten a specific given service. To discover the attack routes, we treat the attacker as an agent in the service environment who can conduct reasoning to meet his requirements using his own capabilities or by delegating some services to other agents.

The other main objective of this work is to identify the required countermeasures to defend the specific services from the attacks. The resulting attack routes are checked to see whether the top malicious goals of the attacker are satisfied. Then a number of counter attacks are added to service models of the actors which are under the attacks. In another round of evaluation, the security goals of actors under the attacks are assessed with the presence of discovered countermeasures. This iterative process of identifying attack routes and countermeasures continues until the risk level of potential attacks is tolerable.

3 The Contribution: Service Security Modeling Framework

In order to analyze the security problems in a service-oriented system and conduct automatic reasoning, we need to model and formalize the service environment. Based on the service requirements modeling ontology (SRMO) [11], we introduce some (security and service related) concepts and adjust basic concepts to aid security analysis, and build a new framework SSMF.

3.1 The SSMF Concepts

The SSMF formally defines the required concepts for analyzing threats from the point of view of the attacker. An *actor* denotes the one who carries out actions to fulfill its requirements with its capabilities. *Services* denote tasks or goals which can be required or provided in service environment. So services can be refined using *AND/OR decompositions*. An actor may have some *capabilities* to provide some services, or *Require* some services. In the latter, the actor can *delegate* his required services to other actors. For security-specific analysis, the concept of *malicious actor*, as a type of actor is considered. Malicious Actors focus on attacking other actors' services. An attack is defined as a triple relation, which involves an actor, a *malicious task* and a service (under the attack). Malicious tasks (or a non-malicious service) may *obstruct* other services.

Actors may have knowledge about certain facts, e.g., services, decomposition of a service, who can provide a service, etc. The knowledge assumptions are later used for attack reasoning and generating potential attack routes. To enable the reasoning and generation of attack routes, we have defined three operations: *add* which inserts a new piece of knowledge into the knowledge set of an actor; *conduct* which decomposes a service; and *satisfy* which is used to represent an actor has satisfied has required service.

3.2 Service Security Analysis

The concepts and operations introduced earlier would provide the bed to reason about the possible attacks in a service-oriented setting. We have organized the service security analysis into four steps: 1) Scenario and environment modeling; 2) Attack goal identification; 3) Attack reasoning from the attacker's point of view; 4) Attack identification and assessment. Fig. 1 illustrates the discussed process for a search service in the web environment.

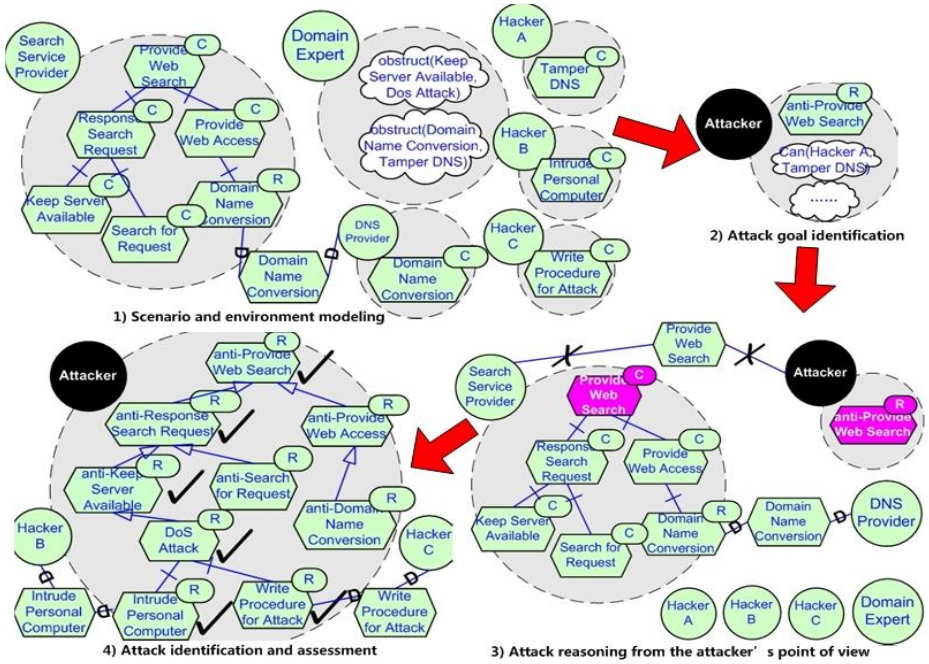

Fig. 1. Service security modeling and reasoning for a web search service example.

1) Scenario and environment modeling. The first step in service security analysis is to model the service environment, i.e. modeling the capabilities and required services within the actors' boundaries, modeling the interactions (delegation of services) among the actors, and formalizing the knowledge each actor hold.

2) Attack goal identification. In the next step, a number of services for which we need to analyze attacks and the security level are selected. Then, the anti-services against the target services are generated. These anti-services are added to a hypothetical attacker which requires achieving the anti-services.

3) Attack reasoning from the attacker's point of view. Since attackers are external entities and we do not have sure knowledge about their capabilities and level of knowledge, we consider the worst possible case in which attackers have enough knowledge in the service environment. In order to discover the attack routes and assess whether an attacker is able to satisfy his ultimate anti-service requirements, we define a number of reasoning rules. The predicates of these production rules are either the specification of services' refinement or available knowledge about possible attacks. The reasoning rules are used to refine the anti-service into malicious tasks and services, delegate the required malicious services of attackers to other actors, and add a piece of knowledge to the attackers' knowledge set. Finally, the satisfaction propagation rule is used to check if the discovered attack route would ultimately satisfy the top anti-services.

4) Attack identification and assessment. Given the discovered attack routes, we can assess the severity or probability of each possible attack route, and accordingly, decide on proper countermeasures. If the attack reasoning cannot uncover a feasible attack route, it proves that the target service is safe enough in the service environment.

4 Conclusions, Limitation, and Ongoing and Future Work

This paper reports on an ongoing work to develop a reasoning method over i* models analyzing the risks against a given services in a SOA setting. In this work, i* models supplied with enough knowledge about the attacks are used for automatic generation of possible attack routes. The results of the reasoning would enlighten the required countermeasures to protect the services under the attack.

However, the current reasoning method and the set of rules we have developed do not support automatic discovery of the countermeasures. Besides, iterative analysis of countermeasures and re-generation and assessment of possible attacks with the presence of countermeasures is not yet incorporated into the method. The other shortcoming of this work is focusing only on the attacks that obstruct a service and make it unavailable. In real world, an attack may threaten integrity or confidentiality of the data that is exchanged or produced by a service. Our next step is to formalize the impacts of malicious tasks against integrity and confidentiality of services, and define the required rules for automatic generation of attacks that threaten these two properties. Finally, we aim to implement the proposed reasoning method using the JESS reasoning engine which uses Rete algorithm to process rules and written in Java.

References

1. Grau, G., X. Franch, and N.A.M. Maiden, *PRiM: An i*-based process reengineering method for information systems specification.* Information and Software Technology, 2008. 50(1-2): p. 76-100.
2. Wang, P., et al., *Building toward Capability Specifications of Web Services Based on an Environment Ontology.* IEEE Trans. on Knowl. and Data Eng., 2008. 20(4): p. 547-561.
3. Yu, E.S.K., *Towards Modeling and Reasoning Support for Early-Phase Requirements Engineering*, in *Proc. of the 3rd IEEE Int. Symposium on Requirements Engineering*. 1997, IEEE Computer Society. p. 226.
4. Strohmaier, M., et al., *Analyzing Knowledge Transfer Effectiveness--An Agent-Oriented Modeling Approach*, in *Proc. of the 40th Annual Hawaii International Conference on System Sciences*. 2007, IEEE Computer Society. p. 188b.
5. Ghanavati, S., D. Amyot, and L. Peyton, *Compliance Analysis Based on a Goal-oriented Requirement Language Evaluation Methodology*, in *Proc. of Int. RE Conf.,* IEEE Computer Society. p. 133-142.
6. Liu, L., E. Yu, and J. Mylopoulos. *Security and privacy requirements analysis within a social setting*, in *Proc. of Int. RE Conf.*, 2003.
7. Elahi, G. and E. Yu, *Modeling and analysis of security trade-offs - A goal oriented approach.* Data Knowl. Eng., 2009. 68(7): p. 579-598.
8. Elahi, G., E. Yu, and N. Zannone, *A vulnerability-centric requirements engineering framework: analyzing security attacks, countermeasures, and requirements based on vulnerabilities.* REJ, 2010. 15(1): p. 41-62.
9. Matulevičius, R., et al., *Adapting Secure Tropos for Security Risk Management in the Early Phases of Information Systems Development*. 2008. p. 541-555.
10. Giorgini, P., et al., *Modeling Security Requirements Through Ownership, Permission and Delegation*, in *Proc. of Int. Conf. on RE*. 2005, IEEE Computer Society. p. 167-176.
11. Liu, L., et al., *Towards a service requirements modelling ontology based on agent knowledge and intentions.* International Journal of Agent-Oriented Software Engineering, 2008. 2(3): p. 324-349.

On Temporally Annotating Goal Models

Sotirios Liaskos[1] and John Mylopoulos[2]

[1] School of Information Technology, York University
Toronto, Canada
liaskos@yorku.ca
[2] Department of Computer Science, University of Toronto
Toronto, Canada
jm@cs.toronto.edu

Abstract. Goal models are theories that describe how various stakeholder goals relate to each other. The constructs that such models use to represent these relationships focus on characterizing the nature of causality that connects goals, without, however, including temporal aspects such as the order in which goal satisfaction takes place. Nevertheless, introducing constructs to allow explicit representation of this ordering aspect has been shown to be useful for a variety of applications. Furthermore, representation of such information need not necessarily be done through formalization or use of external representations; it is also possible through simple annotations on the core goal model. This allows for representing the temporal dimension of goal models in a lightweight and concise manner. However, it does not come without influencing the established way to perceive goal models. In this paper, we discuss our experience in augmenting goal models with temporal information about goal satisfaction, which we performed for the purpose of representing and reasoning about behavioral variability.

Keywords: requirements engineering, goal modeling, i-star

1 Introduction

Goal models constitute theories of how stakeholder goals relate to each other, through the representation of a variety of relationships amongst them, such as means-ends, decomposition and contribution links. In their semi-formal form, i.e. in *i**, goal models represent these relationships in a time-independent way, leaving temporal and ordering considerations outside their scope. This has not prevented these models to prove remarkably useful tools for understanding and reasoning about domains and early requirements, through offering a clean representation of the intentional aspect of a requirements problem.

However, explicit representation of ordering and temporal aspects of goal fulfillment, particularly in the form of direct annotations to the goal diagrams, has often been attempted in the *i**-related literature ([6, 3, 2]). In these efforts, adding ordering annotations constitutes a preparatory step for eventually translating the graphical model into a formal representation (such as Golog or Formal Tropos) for subsequent enrichment and analysis. This seems to be an indication that, for many, such annotations add value to plain graphical goal models in terms of not only guiding the subsequent formalization

process, but also simply communicating this additional aspect of goal fulfillment – the temporal one.

In this short presentation we discuss our own attempts for augmenting semi-formal goal models with temporal information for the purpose of preference-based variability analysis and offer some insights on the benefits and caveats that may accompany such an effort.

2 Objectives of Research

In our recent work we have been studying the potential of using goal decomposition structures, in order to automatically reason about variability in the problem domain ([4]). The structures we are using are single-agent AND/OR decomposition models which are directly analogous to means-ends and decomposition trees that may appear in *i** strategic rationale diagrams. In these models, variability is expressed through the presence of OR-decompositions, which in *i** terms reflect alternative means (subgoals) to fulfill certain ends (parent goal). The presence of OR-decompositions implies that there are alternative *sets* of leaf level tasks that constitute solutions of the problem described by the root goal – these sets are solutions of the AND/OR-decomposition tree. Thus, by constructing such trees, significant amounts of requirements *variability* can be concisely captured. Furthermore, the presence of contribution links of each alternative to soft-goals allows us to further raise the level of abstraction in which reasoning about goal variability can take place.

Nevertheless, we found that there is significant amount of variability in ordering (temporal) aspects of goal fulfillment that is not captured this way. A merchant desires to *Send a Shipment* after the *Payment is Received* for particular customers; but not for others. A meeting initiator may *Announce a meeting* only after the *Meeting Room is Confirmed*, or she may not be so cautious, depending on the nature of the meeting and how busy the room is known to be. At a lower level, an ATM system may or may not *Provide the Card Back* to the user before *Money is Dispensed*. The existing core constructs for building semi-formal goal models do not seem to explicitly accommodate such ordering relationships between goals and tasks.

In our work, we explore ways by which this type of variability can be represented in goal models and subsequently translated into forms that allow useful reasoning about alternatives. The literature seems to offer two fundamental alternatives for adding a temporal dimension to goal models. One is exhaustive formalization of individual goals using for example LTL (see KAOS - [1]), a practice that, although useful for rigorous analysis, it could be characterized as requiring significant effort investment and expertise, while offering a result of potentially reduced comprehensibility. A second alternative, which maintains low formality, is the use of an external representation in sync with the goal models as done in [5] where use case maps are used. However, external representations may also require a minimum of additional non intention-related information to be defined. We believe that direct annotation on the goal models with ordering information, is also a useful practical alternative and, if done carefully, it can significantly increase the amount of information that goal models convey about the domain. We sketch how we do it in our work in the next section.

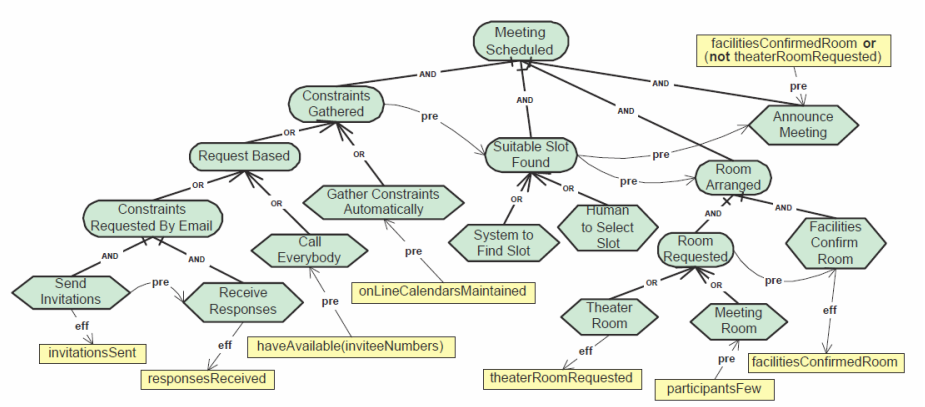

Fig. 1. Goal Models with Temporal Annotations

3 Contributions: Modeling Ordering Constraints

In our proposal for temporally extending goal models, we simply considered that the ordering of goal fulfillment and task performance is relevant. Thus what potentially fulfills the top level goal is not just a set of tasks but a *sequence* thereof. We call such sequences *plans*, borrowing from the corresponding term used in the AI-planning community. A consequence of this extension is that the number of alternatives to be analyzed increases dramatically: the number of solutions of the AND/OR structure is now multiplied by the possible orderings of the tasks that constitute each solution.

Visual representation of ordering constraints on the model itself serves the purpose of controlling the comprehensibility issue that this explosion introduces to users of the model. We introduced the *precedence link* that connects hard elements (hard-goals or tasks) with each other. A precedence link between two goals or tasks implies that satisfaction (resp. performance) of the latter is not possible unless the former has been satisfied (resp. performed) first. In Figure 1, precedence links applied to a goal model for the Meeting Scheduling example are shown. Thus, the task *Facilities Confirm Room* can be performed only if the goal *Room Requested* has been satisfied, hence the precedence link. As seen in the figure, in our application, indicative temporal constraints dictated by the domain realities rather the stakeholder desires, were found to be served very well by the precedence link.

A somewhat bolder addition to the graphical representation that we have attempted is that of domain variables playing the role of precondition and effects of tasks. Domain variables, in the form of predicates, are used to represent the *state* of the environment in which performance of tasks and achievement of goals is attempted. These can describe both volatile and more stable facts in the domain. Thus, in meeting scheduling, *invitationSent*, *haveAvailable(inviteeNumbers)*, *participantsFew* or *theaterRoomRequested* are examples of domain variables. Simple logical expressions of these variables can then be constructed and set as precedence conditions inside the diagram itself as seen in Figure 1. For example, the expression linked to the task *Announce Meeting* on the

right end of the figure says that this task should be performed only after room confirmation by the facilities office has been performed unless it was not the theater room that was requested. In addition, such predicates or lists thereof can be set as *effects* of the performance of task. For example, when the task *Send Invitations* is performed the domain variable *invitationsSent* becomes true. Note that upon introduction of domain variables, plans need to be calculated subject to predefined initial conditions. Through the addition of environmental variables inside precedence and effect elements, plain goal models can be used to visually construct a skeleton of a dynamic domain, to be subsequently automatically translated in a more formal form for further enrichment and analysis.

As we describe below, these seemingly small additions introduce interesting possibilities but also influence the way that a goal model is read and understood, while introducing some challenging conceptual issues.

4 Conclusions

We have applied the visual annotations we described in a variety of goal models either based on artificial data (e.g. Meeting Scheduler or ATM) or based on projects that offered more realistic data (e.g. on the nursing domain). While a formal empirical study is still pending, our so far exploration does not offer evidence that viewing goal models from a temporal point of view obstructs comprehension of the model or the domain. We have however realized that certain implications must be made explicit to avoid ambiguity and lift potential perception problems.

As example of the kind of attention that needs to be paid, consider the precedence link. While the link implies that the destination must be preceded by the origin, it does not necessitate that the origin must be followed by the destination; the latter may as well be absent. To represent that the destination must follow the origin, whenever the latter is satisfied or performed, a different link definition is needed, e.g. a response link. In such a link, non-satisfaction of the origin does not prevent satisfaction of the destination. Note that these two definitions have different consequences when the origin is alternative or optional.

As another example of how temporally extended goal models influence the established understanding of goal models, consider the relationship between soft-goals and precedence. Firstly, direct definition of plain precedence between soft-goals is not necessarily intuitive. For example, simply specifying that *Privacy* is satisfied before *Convenience* may be an unintuitive statement, when for both soft-goals crisp satisfaction criteria do not exist in the first place. Indirect precedence in soft-goal satisfaction via *hurts* and *helps* contributions from ordered hard elements, however, is possible. But this leads us to an understanding of soft-goal satisfaction as a degree (e.g. of satisfaction evidence) that fluctuates during the course of a plan. Reasoning about soft-goal satisfaction is then necessarily subjected to temporal modalities. Thus statements of the form “we want Privacy to be at least partially satisfied” are now extended as “we *never/always/eventually* want Privacy to *never/always/eventually* be at least partially satisfied”.

We found that the modeler who wishes to rely on visual annotations of temporal constraints needs to address issues such as the above by cleanly defining the intuitive meaning of each introduced construct and explaining its implications. This is separate and in addition to the definition of formal semantics, which does not necessarily support comprehension when users are not trained to the underlying formalism (PDDL, Situation Calculus, Formal Tropos/SMV, LTL etc).

5 On-going and Future Work

Our current understanding is that, despite its potential cost and time investment, the best tool to validate compliance with comprehensibility and usability criteria we set for our extension proposals is the empirical study. In our work, we aim at developing variability and preference representation and analysis techniques that are usable not only by analysts whose training may not include understanding of formal languages, but potentially also by common software users with no experience in any kind of conceptual modeling whatsoever. Empirical investigation implies experimental designs that may include a variety of tests, such as successful communication of the intended meaning of the model, the effort it takes for this to happen or the effort it takes for the construction of temporally annotated models and how correctly this is done for different participant profiles in terms of their experience in modeling. These plans for empirical work have gradually claimed a larger and larger piece of our research agenda.

References

1. Anne Dardenne, Axel van Lamsweerde, and Stephen Fickas. Goal-directed requirements acquisition. *Science of Computer Programming*, 20(1-2):3–50, 1993.
2. Ariel Fuxman, Lin Liu, John Mylopoulos, Marco Pistore, Marco Roveri, and Paolo Traverso. Specifying and analyzing early requirements in Tropos. *Requirements Engineering*, 9(2):132–150, 2004.
3. G. Gans, M. Jarke, G. Lakemeyer, and T. Vits. Snet: A modeling and simulation environment for agent networks based on i* and ConGolog. In *Proceedings of the 14th International Conference on Advanced Information Systems Engineering (CAiSE'02)*, Toronto, Canada, 2002.
4. Sotirios Liaskos, Sheila McIlraith, and John Mylopoulos. Towards augmenting requirements models with preferences. In *Proceedings of the 24th IEEE/ACM International Conference on Automated Software Engineering (ASE 2009)*, Auckland, New Zealand, 2009.
5. Lin Liu and Eric Yu. Designing web-based systems in social context: A goal and scenario based approach. In *Proceedings of the 14th International Conference on Advanced Information Systems Engineering (CAiSE 2002)*, Toronto, Canada, 2002.
6. X. Wang and Y. Lesperance. Agent-oriented requirements engineering using ConGolog and i*. In *AOIS-2001 Bi-Conference Workshop at Agents 2001 and CAiSE'01.*, 2001.

Using *i** to Support a Summative Evaluation

James Lockerbie [1], Neil Maiden [1], Amir Dotan [1] & Valentina Lichtner [2]

[1] City University London, Centre for HCI Design, London EC1V 0HB, UK
{J.Lockerbie@soi., N.A.M.Maiden@, Amir.Dotan.1@}city.ac.uk

[2] London School of Economics and Political Science, Information Systems and Innovation Group, London WC2A 2AE, UK
v.lichtner@lse.ac.uk

Abstract. Summative evaluation of a software tool requires the assessment of the defined target outcomes, or high-level goals, of the product. This poses the challenge of how to carry out the assessment in practice. We report our research into addressing this problem by using *i** modelling for a summative evaluation of a work-based learning tool. We describe our use of *i** to identify a set of detailed goals suitable for qualitative assessment. In particular, we report the development and characteristics of the large-scale SR model used in the process, and the utility it provided to contribute towards a successful evaluation. We believe this to be a novel application of *i**, and we present our research outcomes and lessons learned in this area.

1. Introduction

The *i** approach has been widely used in case studies during the early phases of requirements engineering, including our own application to a number of projects as part of our RESCUE process [1]. One such project, called APOSDLE (Advanced Process-Oriented Self-Directed Learning Environment), included *i** modelling to identify future system boundaries, actor dependencies and important system goals for a new knowledge management tool that supports self-directed learning at work. The analysis of this work provided us with a novel insight that the *i** approach would lend itself well to supporting the summative evaluation of the tool at the end of the project.

Summative evaluation of a software tool assesses its defined target outcomes or impacts, and takes place after it has been completely implemented and adequate time has passed to expect outcomes to occur [2]. For APOSDLE, the defined outcomes were expressed as three high-level goals underlying a project vision. Having assessed and evolved the APOSDLE tool itself during two formative evaluations, the aim for the summative evaluation was to assess the satisfaction of these high-level goals to determine whether the product could effectively support learning in the workplace.

2. Objectives of the Research

The main aim of this research was to investigate and evaluate the use of *i** modelling to support a summative evaluation. In particular, our objectives for the study were: (i) to assess the characteristics of an *i** model needed to identify a set of detailed and

measurable goals suitable for qualitative assessment; and (ii) to assess the utility provided by *i** in the evaluation.

In working towards these objectives, we sought to identify lessons learned in order to form an agenda for future work in this area. The outcomes of our research are summarised in the next section.

3. Scientific Contributions

In order to assess the three high-level goals we needed to identify a set of lower level goals suitable for qualitative assessment. Therefore we applied the *i** approach, using models developed in our *i** modelling tool, REDEPEND [3], during four workshops held with project partners. We initially focused on capturing work-based learning soft goals from the application partners, who would later provide the work domains and participants for the summative evaluation. These goals were captured at the start of the project prior to any concrete implementation of the APOSDLE tool, and reflected a more detailed decomposition of the main high level goals – worker support, learner support, and expert support.

We then focused on the input from the technical partners to model potential solution ideas for achieving, or contributing towards, the application partner soft goals. Figure 1 shows the large scale of the SR model, which includes the soft goals of the application partners and the functionality of the APOSDLE system. The expanded section shows an example of the APOSDLE tool (actor B in the figure) contributing towards a non-disturbing learning environment for the knowledge worker (actor A).

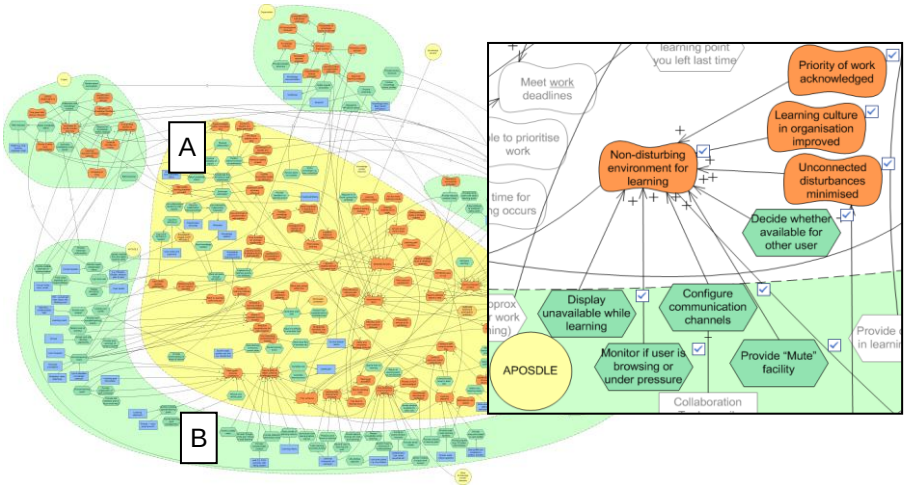

Figure 1: The APOSDLE SR model, with an inset showing functions of the APOSDLE tool and contributions to the knowledge worker soft goals

Based on an assumption of goal hierarchy, lower level goals are more specific than higher level goals and as a consequence lend themselves better to measurement, as

illustrated in Figure 2. Therefore, three goal hierarchies were extracted from the SR model, and the leaf nodes of these hierarchies were taken as goals that could be measured in a meaningful and reliable way. The leaf node soft goals related to the three main aspects of APOSDLE: to support learners, workers and experts in the workplace. As it was not practical to assess all of the low-level soft goals, the application partners identified the ones that were a high priority for the evaluation.

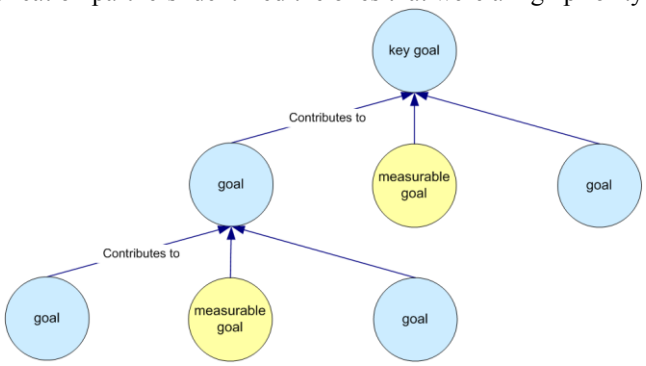

Figure 2: Goal hierarchy showing how lower-level measurable goals can be used to evaluate key high-level goals

Ten key soft goals were selected by the project partners through a questionnaire and follow up meetings as the focus of analysis and evaluation. Qualitative data was collected over a 4 month period, including first-order diary entries [4], interview scripts and log data. Qualitative evidence from the evaluation suggested that the APOSDLE tool contributed towards 9 out the 10 goals, albeit to varying degrees. Given these results, we then explored whether a second-order analysis of the SR model could provide additional insight for the evaluation.

We ran propagations on the prioritised soft goals to identify higher-level soft goals in the hierarchy, and instantiated these parts of the model for each application partner in order to understand the impact of APOSDLE on each of the three domains. It was interesting to find positive contributions applied to a few soft goals that were not supported by the final implementation of the APOSDLE tool. This showed that APOSDLE had system-wide qualities that went beyond the direct implementations intended to achieve application partner soft goals. As expected, the views of the application partners on soft goal achievement varied according to the work domain. It was also interesting to find higher-level goals with positive satisfaction despite lower-level supporting soft goals being reported as unsatisfied.

4. Conclusions

The research is not complete, but evidence suggests that *i** is an effective tool for structuring a summative evaluation. We assess our two research objectives below.

Our first objective was to assess the characteristics of the *i** model needed to identify a set of measurable goals suitable for qualitative assessment. The lack of clear soft goal hierarchy in the SR model was an issue. We focused on soft goals with the most contributions and flattened out the contributing elements, ensuring that the

majority of the soft goals in the model were covered. A more hierarchical model would have been better suited to the evaluation. Also, it would have been beneficial to have explicitly focused the structure of the SR model on the top level project goals during its development. The interpretation of soft goal descriptions caused problems, with different stakeholders having different understandings, or even no understanding. We later provided rationales for each of the soft goals which improved comprehension. Another challenge for the project partners was the scale of the model, therefore a set of soft goals needed to be prioritised for the evaluation. The project partners had different priorities, and this affected the completeness of the assessment. However, the scale and detail of the model was useful for the analysts, and as such represented a common scalability trade off experienced in *i** modelling.

Our second objective was to assess the utility of applying *i** to the summative evaluation. The main observed benefit was that the SR model provided a set of lower-level goals to structure the evaluation – goals that we otherwise would not have had. These soft goals were also connected to aspects of the tool's functionality, providing context for the evaluation and helping goal selection for the assessment. Highlighting important dependencies and relationships was useful, and showed that the soft goals were not isolated, and that contributing factors propagated throughout the socio-technical system. Whilst the lack of clear hierarchy in the SR model made the identification of measurable soft goals more difficult, this same characteristic of the model also added value to the evaluation. We were able to show that system-wide qualities of APOSDLE went beyond the direct functional implementations intended to achieve application partner soft goals. In addition, we were able to identify contributions that did not fit with the notion of a set goal hierarchy i.e. higher-level goals with positive satisfaction were identified despite lower-level supporting soft goals being evaluated as unsatisfied. Work from this analysis provided additional results and valuable insight for the evaluation.

5. Ongoing and Future Work

We will take forward these lessons learned for our next project in order to develop more fit-for-purpose models, and to further exploit the observed benefits of applying the *i** approach to summative evaluation.

References

1. Jones S.V. & Maiden N.A.M., 'RESCUE: An Integrated Method for Specifying Requirements for Complex Socio-Technical Systems', In: Requirements Engineering for Socio-Technical Systems, ed. J.L. Mate & A. Silva, Ideas Group, 2005, pp245–265.
2. http://www.evaluationwiki.org/index.php/Summative_evaluation – retrieved on 03/03/2010
3. Lockerbie, J. & Maiden, N.A.M., 'REDEPEND: Tool support for *i** modelling in large-scale industrial projects', Proceedings of the Forum at the CAiSE'08 conference, Montpellier, 2008, pp69–72.
4. Lichtner, V., Kounkou, A., Dotan, A., Kooken, J., Maiden, N.A.M, 'An online forum as a user diary for remote workplace evaluation of a work-integrated learning system', Proceedings of Conference on Human Factors in Computing Systems, 2009, pp2955–2969.

On the use of the Goal-Oriented Paradigm for System Design and Law Compliance Reasoning

Mirko Morandini[1], Luca Sabatucci[1], Alberto Siena[1], John Mylopoulos[2], Loris Penserini[1], Anna Perini[1], and Angelo Susi[1]

[1]Fondazione Bruno Kessler - IRST, Trento, Italy
{morandini,sabatucci,siena,perini,penserini,susi}@fbk.eu
[2]University of Trento, Italy
jm@disi.unitn.it

Abstract. The concept of goal may be used to model intentions of human actors, such as requirements analysts or designers, as well as the reasons for pro-active behaviour of software agents.
This short paper describes three ongoing research efforts on the application of the Goal-Oriented paradigm to system requirements analysis, system design and development of self-adaptive software agents.

Key words: Goal-Oriented paradigm, Requirements Engineering, Design Patterns, Software Agents

1 Introduction

The concept of *goal*, as a state of affairs that an actor (human, organization or system) wants to achieve, together with social aspects and other intentional concepts define the Goal-Oriented (GO) paradigm that has been largely studied and applied in Requirements Engineering (RE) since more than ten years [1, 2].

Evidences of the usefulness of the GO paradigm can be found in a variety of real world experiences that exploited available GO methodologies that support software system development activities ranging from requirements acquisition, analysis and understanding, to design and test cases derivation.

A key feature of the GO paradigm is that of allowing to model and reason about alternatives. These alternatives are usually represented in terms of OR-decompositions of goals or tasks, which lead to the definition of goal trees, whose alternative paths may be evaluated against possible situations of benefit, drawback or conflict.

This paper summarizes three ongoing research efforts at FBK-IRST in which the GO paradigm plays a central role at support of decision making for domain and system analysts, system designers and also for proactive artificial agents, in different contexts: (i) when analysts have to decide about the compliance of a set of the system's requirements with respect to law; (ii) when designers have to choose a suitable design pattern; (iii) when software agents of a self-adaptive system have to decide at run-time which one among their alternative sub-goals to achieve, while attempting to satisfy the main goals assigned to them by the designers.

2 An *i** Framework for Law Compliance: *Nòmos*

Laws and regulations address processes and associated information systems within organizations. Available methods and techniques for system design give little support to the requirements engineer when analysing the impact of those regulations during the definition of requirements for a new system, or when an existing organization has to restructure and re-engineer its operation in order to achieve compliance.

Goal-oriented requirements engineering rests on the idea of deriving the requirements for a software system from the analysis of the goals that the system-to-be will support once developed and deployed. However, when the stakeholders are addressed by laws, the system-to-be has to be aligned with the legal prescriptions, too, and goals per se do not provide information about such an alignment. This is the problem of law compliance of goals models. Finding a solution to this problem means finding the assignment of actors' responsibilities (goals) such that if every actor fulfils its goals, then law is respected. To address this problem we adopt a modelling approach which consists in starting from a model of legal prescriptions, and building the model of goals in an incremental way that maintains the alignment with the prescriptions.

The *i** modelling language focuses on intentional elements as the key to describe and understand a given organizational setting. Laws play a different role as they have (i) physical existence as natural language sentences in legal texts; and (ii) prescription objectives with regard to the organization. The contribution of the *Nòmos* framework consists in a conceptual binding between the two conceptions of intentions and regulations. The binding relies on the analysis of legal sentences, which ultimately allows to identify the juridical concept of **normative proposition** (NP), as the most atomic proposition able to carry a normative semantics, containing information concerning: (i) the subject(s) addressed by the NP itself; (ii) the legal modality; and (iii) the description of the object of such modality. The legal modality is one of the eight elementary rights, classified by Hohfeld [3] as: *Privilege*, which is the entitlement for a person to discretionally perform an action, regardless of the will of others and *Claim*, which is the entitlement for a person to have something done from another person, with their correlative rights *No-claim* and *Duty*; *Power*, which is the (legal) capability to produce changes in the legal system towards another subject, and *Immunity*, which is the right of being kept untouched from other performing an action, and their correlatives rights, *Liability* and *Disability*.

In order to support modelling of laws and compliance solutions, the *i** metamodel (in the variant proposed by the Tropos methodology [4]) has been extended with the *Nòmos* concepts, integrating the two set of concepts. Moreover, a systematic process has been defined to support the building of compliant requirements models, as described in detail in [5, 6], considering the following issues: *the binding of domain stakeholders with subjects addressed by law*, *the identification of legal alternatives*, *the identification of potential realisations of normative propositions*, *the identification of legal risks*, *the identification of proof artefacts*, *the constraining of delegation of goals to other actors*.

As future work, we are investigating the use of argumentation framework in order to support the acceptability of compliance solutions.

3 Design Pattern Representation with Motivations

Software patterns are reusable solutions to recurring design problems and — since their definition — are considered a mainstream of software reuse practice. They are typically documented with a textual description of the context where they can apply, the purpose for their reuse and forces to balance. For encouraging the understanding of design patterns and to ease their application during the design phase, many approaches have been proposed to provide the solution by using formal, semi-formal graphical notations or logic languages [7].

We propose to use i^* to represent not only the pattern solution, but also the whole reasoning process that led to its formation, including motivation, trade-offs and alternatives [8]. The main motivations of this approach are (i) to improve the communication encapsulated in a design pattern without changing the informative content and (ii) to provide some criteria for motivating pattern selection and reuse during the design process.

The proposed abstraction considers the design activity as the application area in which we apply the i^* framework, and the *Designer* as the main *Actor* of the design activity, whose job is to balance design forces coming from the system to be modelled. The designer's activities arise from needs, such as: (i) the achievement of *Design Goals* to solve specific design problem emerging during the modelling of the system, and (ii) the compliance with *Design Properties* (or soft goals) that specify qualities of the system. In this context a pattern is a collection of collaborating roles, intended as autonomous holder of design intentions that are delegated of some responsibilities from the designer, namely: (i) design goals/soft- goals to be achieved, (ii) design tasks for introducing a well-known solution, and (iii) system elements to introduce or to organize in the solution.

In this approach, the designer is supported with techniques for balancing pattern contextual forces and for customizing the pattern implementation to the specific application context. We exploit the Strategic Dependency model for representing the high-level responsibility organization of a design pattern, and the Strategic Rationale model for entering in detail in the solution structure.

The Strategic Dependency main role is always the designer who delegates design intentionality to pattern roles. This view allows for highlighting main intents and motivations of a pattern, and it is an instrument for quick selection of the pattern to reuse from a catalogue. On the other side the Strategic Rationale model allows for reasoning on design issues, considering consequences and balancing design alternatives, thus customizing the solution to meet forces coming from the context.

This approach opens new research directions we are working on: (i) to represent — and reason on — pattern composition and conflicts, and (ii) the use

of an ontology for standardizing design intentional elements, that may help in automatic discovery of patterns to reuse.

4 Goal-Oriented Development of Self-Adaptive Systems

Self-adaptive Software (SAS) aims at dealing autonomously with unpredictable changes which occur in the dynamic environment it executes (at run-time), on the basis of its knowledge and of the objectives it has been designed for. We aim at defining a process and a tool-supported design framework to develop SAS with the necessary knowledge that enables adaptation of their behaviour at run-time. Belief-Desire-Intention (BDI) agents [9] were chosen as reference architecture and implementation platform.

The proposed framework and development process, called *Tropos4AS* (Tropos for Adaptive Systems) [10], exploits the basic Tropos early and late requirements phases, with an extensive use of variability modelling [11], and extends the design phase with environment modelling, extended goal modelling and failure modelling.

The *environment model* captures the non-intentional entities involved in, used and perceived by the SAS, which are necessary for interfacing the system with the surrounding world. For instance, the environment for a cleaner robot includes the floor, the dustbins and a battery charging station. These entities are represented through *artifacts* [12], non-intentional entities that provide functionalities usable by agents to sense and act in the environment.

In *extended goal modelling*, the goal model is linked with the environment. The process of goal achievement is related to the environment by defining the context where the goal is applicable, the conditions which lead to its adoption, its achievement and failure states. Different goal types further characterise the process of goal achievement, distinguishing among goals that the SAS has to achieve in a given situation (e.g. clean a wet floor), goals that the SAS has to achieve and maintain all along its life cycle (e.g. maintain a battery loaded), and goals that can be rather described as executing a given procedure (e.g. searching for dirt). Goal types were already present in the Formal Tropos [13] language, which was however developed with the aim of consistency verification via model-checking techniques. On the contrary, Tropos4AS focuses on the semantics of an agent's goal model that can be directly coded in a BDI agent programming language (e.g. Jadex, 2APL or Jack), following predefined mapping rules to link the goals in the design artefacts to the agent goals in the code, respecting the semantics of the goal model [14].

In *failure modelling*, Tropos4AS aids the designer in anticipating possible failures, giving a process to elicit errors possibly causing them and analysing the possibilities to fix them. Entities added to the Tropos4AS meta-model for supporting failure modelling are: *failures*, representing undesirable states known to the designer for the impossibility to achieve a goal, perceivable *errors* that may be the cause of these failures, and *recovery activities*, i.e. actions that the SAS may undertake to recover from errors, preventing failure.

The implementation is based on a tool-supported mapping of goal models to Jadex BDI agent code, maintaining the goal model with its semantics also at run-time. This goal model drive the agent's behaviour at a *knowledge level*, defining the relationship between requirements (goals) and the agents' capabilities, and to ensure traceability of run-time choices back to the design.

The main issue in our research agenda concerns the consolidation of the tools supporting the development and testing of self-adaptive software. Moreover, a validation of the framework on more realistic scenarios will be performed, focusing on the adaptive qualities of the system under development.

References

1. Mylopoulos, J., Chung, L., Yu, E.: From object-oriented to goal-oriented requirements analysis. Commun. ACM **42**(1) (1999) 31–37
2. van Lamsweerde, A.: Goal-oriented requirements engineering: A guided tour. In: RE, IEEE Computer Society (2001) 249
3. Hohfeld, W.N.: Fundamental Legal Conceptions as Applied in Judicial Reasoning. Yale Law Journal 23(1) (1913)
4. Susi, A., Perini, A., Mylopoulos, J., Giorgini, P.: The tropos metamodel and its use. Informatica (Slovenia) **29**(4) (2005) 401–408
5. Siena, A., Mylopoulos, J., Perini, A., Susi, A.: Designing law-compliant software requirements. In: 31st International Conference on Conceptual Modeling (ER'09), Gramado, Brasil (November 09 2009) 472–486
6. Siena, A., Mylopoulos, J., Perini, A., Susi, A.: Towards a framework for law-compliant software requirements. In: ICSE Companion. (2009) 251–254
7. Mikkonen, T.: Formalizing design patterns. In: Proceedings of ICSE '98, Washington, DC, USA, IEEE Computer Society (1998) 115–124
8. Sabatucci, L., Cossentino, M., Susi, A.: Introducing motivations in design pattern representation. In: Proc. of ICSR 11. LNCS, Springer (2009) 201–210
9. Rao, A.S., Georgeff, M.P.: Bdi agents: From theory to practice. In: ICMAS. (1995) 312–319
10. 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, New York, NY, USA, ACM (2008) 9–16
11. 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)
12. Omicini, A., Ricci, A., Viroli, M.: *Agens Faber*: Toward a theory of artefacts for MAS. Electr. Notes Theor. Comput. Sci. **150**(3) (2006) 21–36
13. Fuxman, A., Pistore, M., Mylopoulos, J., Traverso, P.: Model checking early requirements specifications in Tropos. In: IEEE Int. Symposium on Requirements Engineering, Toronto (CA), IEEE Computer Society (August 2001) 174–181
14. 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)

Using *i** Meta Modeling for Verifying *i** Models

Antonio de Padua Albuquerque Oliveira [1, 2], Julio Cesar Sampaio do Prado Leite [2], Luiz Marcio Cysneiros[3]

[1] Universidade do Estado do Rio de Janeiro – UERJ
Rua São Francisco Xavier, 524 - 6 andar - Maracanã - Rio de Janeiro, Brazil
[2] Pontificia Universidade Catolica do Rio de Janeiro – PUC-Rio
Departamento de Informatica, Rua Marques de Sao Vicente 225 – Rio de Janeiro, Brazil
[3]York University – School of Information Technology
4700 Keele St. – Toronto, Canada
padua.uerj@gmail.com - www.inf.puc-rio.br/~julio - cysneiro@yorku.ca

Abstract. The i* Framework has been regarded as a suitable organizational modeling approach for representing early requirements of complex software systems. Intentionality in organizational context is the aim of i* Framework. We believe that a general lack of awareness about the i* language is the main reason for some authors mistakes including the lack of focus on intentionality. Aiming to help changing this scenario we made an exercise of modeling i* modeling using only i* concepts. Considering that building any diagram is more difficult than reading it we propose to use the i* meta model as basis for a series of check-list based questions. Based on the meta-model these questions work as a check-list for building an i* model, or if used after model creation as a basis for check-list reading as per Fagan's inspection. We believe our contribution relies on providing a systematic and well founded way of improving i* models quality.

Keywords: meta-modeling, Goal Oriented Requirements Engineering (GORE), early requirements, verification.

1 Introduction

The i* Framework uses two models: the Strategic Dependencies Model (the SD Model) and the Strategic Rationale Model (the SR Model). Furthermore several simple elements are used by these two models in order to represent social actors and dependency relationships among actors inside an organization. We have modeled i* in i* using the same perspective adopted by the i* Framework: "the intentionality perspective". Intentionality means to represent motivations and desires of actors [2].

In this way, first we considered the SD model as the organization and therefore actors (agents occupying positions and playing roles) are the elements (the actors and the four kinds of dependencies) that act in an SD model. Second, by the same token, we consider the SR model as the organization and i* elements were considered the actors (dependencies, all kinds of means-ends links and task-decomposition) that act in the SR model. Applying this abstraction exercise we believe that the intentionality of all elements and their relationships are exposed in a concise model.

This concise model is the basis for deriving the SD and SR check-lists. In this abstract, for space considerations we have shown just the SA diagrams, the SD and SR diagrams may be seen in a technical report [4]. However we have abstracted from these two meta-diagrams their key-points, as to better explain the check-list derivation.

2 Objectives of the research

Making check lists based on i star framework concepts

i* Modeling Framework's concepts and ideas are the basis for our meta-model which provides a clear statement: "*goals are states of affairs that an actor plans to achieve*" [2]; they are not activities or functions. Because there are some misuses of this definition we strongly recommend the adoption of the following standardization used by [3]: (i) **goal** → object + BE + verb in passive voice; (ii) **softgoal** → quality attribute + [object or task as topic]; (iii) **task** → verb in infinitive + object; and (iv) **resource** → name of the object.

In the next section diagrammatic results either by diagram or by key points are explored after enforcing this rule in representing the i* language.

3 Scientific contributions

Verifying an SD Model

The aim of the SD model is to represent strategic dependencies among actors. Using an abstraction we considered that all possible elements from i*, e.g. tasks, goals, positions, roles, and so on will be "actors" in our abstraction. Applying this abstraction we consider "actors" in SD model as being agents, which occupy positions and play roles.

Consequently, as we show in Figure 1, elements (links and nodes) are mapped as agents. Figure 1 is an SA Diagram, this diagram was proposed by Leite et al. [1] as way of structuring the i* concepts of actor, agent, role and position [2]. One agent, in the SD model, can occupy only two positions; either a position of an actor or a position of a dependency, because those "actors" can be classified in two kinds: "actors" which represent actors in the strict sense and "actors" which represent dependencies between actors, as per our abstraction. While occupying an actor's position an element can cover two kinds of roles: *dependee* or *depender*, roles are specializations of actor. On the other hand, while occupying a dependency position an element can cover four kinds of roles: *Goal Dependency*, *Resource Dependency*, T*ask Dependency*, or S*oftgoal Dependency.* So, each one of these four roles is a *dependum* as in [2].

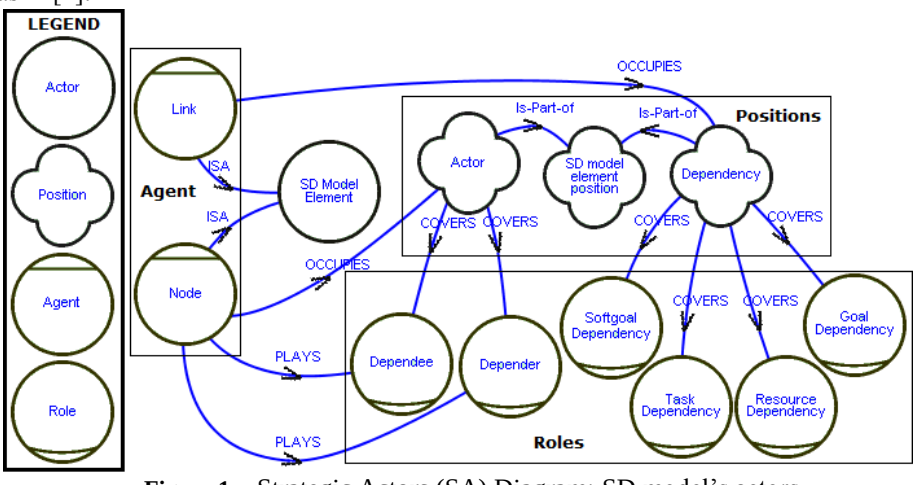

Figure 1 – Strategic Actors (SA) Diagram: SD model's actors

Building an SD Model for the SD Model

Continuing our abstraction exercise for the i* SD model, we created an SD model [4], from which we extracted three key points in order to represent the four types of dependency relationships (see the four roles of dependency (dependum) in Figure 1).

Key points: (1) Strategic dependency means that there is always a depender's goal to be achieved, (2) dependee has a commitment with a depender - Yu's thesis p. 12 [2], and (3) depender believes that dependee is able to carry out the commitment.

SD model Check List

- Is each element in the SD model either actor or dependency?
- For each dependency: Is one actor the depender and the other the dependee?
- For each goal dependency:
  - Does goal dependency obey the goal standardization?
  - Can the dependee achieve the goal the depender wants to?
  - Why the dependee is going to achieve the goal the depender wants to?
- For each softgoal dependency:
  - Does softgoal dependency obey the standardization?
  - Can the dependee achieve the softgoal the depender wants to?
  - Why the dependee is going to achieve the softgoal depender wants to?
- For each task dependency:
  - Does task dependency obey the goal standardization?
  - Can the dependee perform the task the depender wants to?
  - Why the dependee is going to perform the task the depender wants to?
- For each resource dependency:
  - Does resource dependency obey the standardization?
  - Can the dependee provide the resource the depender wants to?
  - Why the dependee is going to provide the resource depender wants to?

Verifying an SR Model

The aim of the SR model is to represent strategic rationale inside the actors' boundary. Applying our abstraction, the SR model has "actors" which can appear in the SD model and has other actors that are peculiar to SR model.

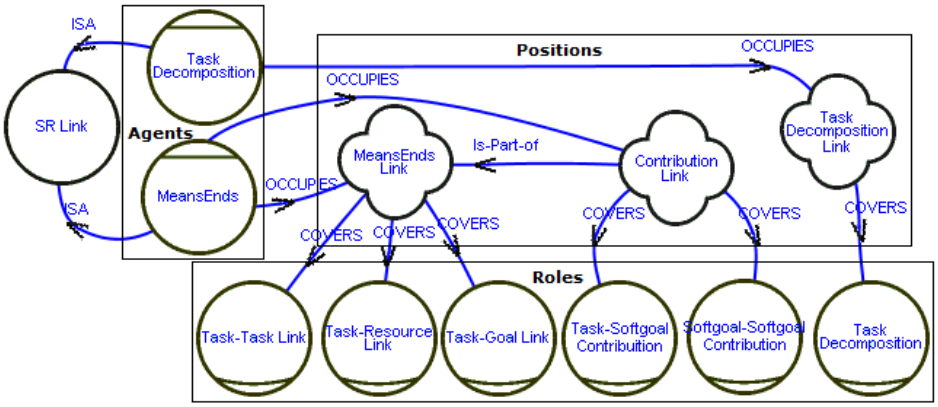

Figure 2 – Strategic Actors (SA) Diagram: SR model's actors

Figure 2 shows SR Links. SR models use two kinds of links, which were represented being agents: *Task Decomposition* and *MeansEnds*. We can observe that

the agent *Task Decomposition* can occupy only one single position in the SR model (Task Decomposition Link) which covers only one single role (Task Decomposition), but on the other hand the agent MeansEnds can occupy two positions: *MeansEnds Link* or *Contribution Link* which are considered part of *MeansEnds Link*. The position *MeansEnds Link* covers three roles: *Task-Task Link*, *Task-Resouce Link* and *Task-Goal Link*. The position *Contribution Link* covers two roles: *Softgoal-Softgoal Contribution* and *Task-Softgoal Contribution*.

Key points: (1) there are two constructs to represent the rationale inside actor's boundary: means-ends and task-decomposition, (2) there is only one way to represent task decomposition, (3) there are five kinds of means-ends link, and (d) means-ends links concerning a softgoal (always as an end) is named "contribution link".

Building an SD Model for the SR Model

Regardless of intentionality, an actor in the SR model should have in the highest level two ways to express desires and motivations: goals or softgoals. Accordingly, it was represented in [4] that the *Actor* depends on either the agent *EndSoftgoal* to have a softgoal satisfied or the agent *EndGoal* to have a goal be achieved. We called endSoftgoal and endGoal because Yu's thesis [2] placed softgoals, goals, resources, and tasks as end positions. There are also two possibilities: (a) a subTask may be an instance (INS) of a MeanTask and may be an instance (INS) of an EndTask and (b) a ResourceFor may be an instance (INS) of an EndResource.

Key points (derived from [4]): (1) intentionality is represented in an SR model by goals and softgoals, (2) there are two ways to represent the rationale inside actor's boundary: using a means-ends or a task-decomposition links, (2a) there is only one way to represent a task-decomposition link and there are five kinds of means-ends links (four have a task being a mean agent and one have a softgoal being a mean agent), (2b) a decomposed task can have four kinds of sub components: subTask, ResourceFor, subGoal and SoftgoalFor, (3) there are five situations of instances: (3a) a meanSoftgoal may be a an endSoftgoal, (3b) a meanGoal may be a subGoal, (3c) a resourceFor may be an endResource, (3d) a subTask may be a meanTask, and (3e) a meanTask may be a endTask.

Building an SR Models for the SR Model

As we noted before (Figure 2), an SR model is based on two "agents" for representing the rationales inside strategic actors, the links: Means-Ends and Task-Decomposition. In the meta-model we considered as organization "The SR Model" and consequently we represented the SR model actors for this organization [4].

Our experience in i* modeling suggested us a reduction mechanism: a proposal for simplifying the SR model. The means ends links *task-task* and *task-resource* should be eliminated. They are not necessary because they can be considered and modeled as a *task-goal* link, like "task Be performed" and "resource Be prepared".

SR model Check List

I - For each actor: Are goals and softgoals the roots in the highest level?

- Is each element a Contribution Link, a Task Decomposition Link or a MeansEnds Link?
- Is each MeansEnds Link, a Task-Goal Link, a Task-Task Link or a Task-Resource Link?

II - For each softgoal: Does the softgoal obey the standardization?

- In case of a single softgoal: Is there a NFR catalog for that softgoal?
- In case of a contribution: Is the contribution, a Task-Softgoal Contribution or a Softgoal-Softgoal Contribution?

III - For each goal: Does the goal obey the goal standardization?

- Can the actor achieve the goal by him(her)self? Why not?
- Is the task good enough to achieve the goal? Why?

IV - In case of Task Decomposition:

- For each softgoalFor: Has the softgoal answered the questions in II?
- For each subGoal: Has the goal answered the questions in III?
- For each subTask: Does subTask obey the standardization?
  - Can the actor perform the task? Why not?
  - Is the task necessary for the main task?
- For each resourceFor: Does it obey the standardization?
  - Is the resourceFor already prepared?
  - Is the resourceFor necessary for the main task?

V - For each goal dependency: Does dependency obey the standardization?

- Can dependee achieve the goal depender wants to?
- Why dependee is going to achieve the goal depender wants to?

VI - For each softgoal dependency: Does it obey the standardization?

- Can dependee achieve the softgoal depender wants to?
- Why dependee is going to achieve the softgoal depender wants to?

VII - For each task dependency: Does dependency obey the standardization?

- Can dependee perform the task depender wants to?
- Why dependee is going to perform the task depender wants to?

VIII - For each resource dependency: Does dependency obey the standardization?

- Can dependee provide the resource depender wants to?
- Why dependee is going to provide the resource depender wants to?

4 Conclusion

The main goal of this work is to improve the understanding of the i* framework so that requirements engineers can fully explore i* strengths. The work reminds the orthogonal role of each element, gives emphasis over what should be modeled and also shows the possibilities of i* modeling as a meta-modeling representation.

We have applied the i* Check Lists asking graduated students for verifying classmate's diagrams for simple modeling exercises. Our results are encouraging; however, we need to apply the strategy in different situations in order to get practical evidence of the effectiveness of our strategy. While carrying out these experiments we will also evaluate how well the approach scales to more complex models.

References

1. Leite, Julio; Werneck, Vera; Oliveira, A. Padua A.; Capelli, Claudia; Cerqueira, Ana Luiza; Cunha, Herbert; Baixauli, Bruno; "Understanding the Strategic Actor Diagram: An Exercise of Meta Modeling", The X Workshop on Requirements Engineering; Toronto, Canada - May/2007.
2. Yu, E. Modelling Strategic Relationships for Process Reengineering. PhD Thesis, Graduate Department of Computer Science, University of Toronto, Toronto, Canada - 1995.
3. Zheng You, Using meta-model-driven views to address scalability in i* models, Master of Science thesis, Graduate Department of Computer Science, University of Toronto, 2004, pp. 231.
4. Oliveira, A. Padua A.; Leite, J. C. S. P; Cysneiros, L. M.; "An exercise of Meta-Modeling: the Case of i*", In Monografias em Ciência da Computação, DI/PUC-Rio - 2010.

Using *i** and Tropos in a Software Engineering Contest: Lessons Learnt and Some Key Challenges

João Pimentel[1], Emanuel Santos[1], Bárbara Santos[1], Clarissa Borba[1], Josias Paes[1], Carlos Lima[1], André Bezerra[1], Jaelson Castro[1], Fernanda Alencar[2], Carla Silva[3], Ricardo Ramos[4], Marcia Lucena[5]

[1] Universidade Federal de Pernambuco - UFPE, Centro de Informática, Recife, Brazil
{ jhcp, ebs, bss, ccb, jpsj2, cdql, alrb, jbc }@cin.ufpe.br
[2]Universidade Federal de Pernambuco - UFPE, Departamento de Eletrônica e Sistemas, Recife, Brazil,
fernandaalenc@gmail.com
[3]Universidade Federal da Paraíba - UFPB, Centro de Ciências Aplicadas e Educação, Rio Tinto, Brazil
ctaciana@ccae.ufpb.br
[4]Universidade Federal do Vale do São Francisco – UNIVASF, Colegiado de Engenharia da Computação, Juazeiro- BA, Brazil
ricargentonramos@gmail.com
[5]Universidade Federal do Rio Grande do Norte - UFRN, Departamento de Informática e Matemática Aplicada Natal, Brazil
marciaj@dimap.ufrn.br

Abstract. In this paper we present some of the lessons learnt when using *i** and Tropos in the SCORE 2009 competition (Student Contest on Software Engineering Contest). During the development of the BTW-UFPE Project we had to address several challenges, including: limitations of modeling notation, ensuring the quality of the intentional models, transition from requirements to architecture description as well as from architecture description to detailed design. Moreover, we identified the need to deal with intentional and domain variability in *i** models and the lack of appropriate tool support. In this paper we also present some of the ongoing research which is aimed at addressing some of the identified challenges.

Keywords: *i*, Tool Support, Reuse*

1 Introduction

In the years of 2008 and 2009, we fully developed a multi-agent system which was a finalist of the Student Contest on Software Engineering – SCORE 2009 [18]. We chose the "BTW - If you go, my advice to you" project, which is related to the development of an information recommender system intended to help travelers when walking around streets unknown to them [11]. We adopted an agent oriented approach, based on the best practices of Tropos [21], to deliver our project. Agents are a natural choice when it comes to advice suggestion [24]. In the sequel we report

on some of the lessons learnt and present various challenges for the technology adoption:

i) Requirements Elicitation - Our project relied on the PRiM process [7] for the elicitation phase. It was very useful for identifying actors, tasks and resources. However, it was of limited assistance for the discovery of goals and softgoals. An interesting possibility would be to use an approach based on Ground and Activity Theory [5] for finding (soft) goals. Unfortunately, due to time limitation we could not pursue this option.

ii) Different versions of the modeling language and tool support- There are different versions of the requirements language. For example there is the original *i** [22] and the *i** wiki version. Besides, different dialects have been proposed by the research groups, such as Tropos [4] and GRL [2]. In particular, in our group we have developed extensions to deal with variability and cardinality of elements [3]. We also developed two approaches to improve the modularity of the requirements models [9, 17]. This large collection of i* dialects leads to uncertainty when learning and selecting the most appropriate version for the job at hand. Moreover, the lack of standardization also constrains the usage of modeling tools, since most of them are designed for a specific version of the notation. It is urgently required a family of tools to support the various goal modeling language variants. For the Score contest we modeled requirements according to the U-Tropos process [21] and used the OME tool.

iii) Quality of models - Once we have built a model, we need to assess its quality with respect to some criteria. Several metrics have been defined for the *i** language [16, 6]. However, we also need an approach to relate the criteria of interest (quality attributes) to the questions to be answered and metrics to be collected. Moreover, the evaluation phase should also be linked to an improvement stage, where the potential problems detected could be addressed [14]. In our project we had to rely on the team members' expertise to assess the quality of the models.

iv) Transition from requirements to architecture models – A key challenge is to relate requirements and architectural models. Although some approaches have been proposed in the literature [9], the available tool support is very limited. In our project we did not use any systematic means for the derivation of the architectural model. Hence, it was difficult to assess if the derived architecture fulfilled (all) the requirements. Furthermore, the rationale for the choice of a candidate architecture (based on non-functional requirements or softgoals) was not recorded. This is especially critical on iterative projects.

v) Transition from architecture models to detailed design - Once the architectural model is stable, a detailed design has to be delivered. This is a daring task. Little assistance is currently provided. We relied on some UML artifacts to describe the design information. The members experience was a key factor for this task. A more systematic approach is urgently required.

vi) Transition from detailed design to source code – In the context of object-oriented development there are plenty of tools to generate a draft source code from design models. We predict that a draft source code could also be automatically generated from an architectural and detailed design models. For the contest our team had to generate the target code manually.

vii) Reuse of Multi Agent Systems - A lot of effort is necessary to develop a single multi-agent system (MAS), i.e. the BTW in our SCORE competition. If a similar or related MAS is required, often none of the previous artifacts are reused. Hence, it is paramount to promote software reuse in the context of multi-agent development. One of the key issues is to be able to express the common and variable parts of the artifacts. For example, there are recent works representing variability in *i** models [1, 3, 8]. However, it is not clear yet how this variability information can be used to develop an Agent-based Product Line (APL).

Our research group is addressing some of these issues, namely ii, iii, iv and vii. In the remainder of this paper we are going to describe our current research lines related to tool support and promotion of reuse. Some of the other issues are partially handled in other works [9, 14, 17]. In Section 2, we describe our research objectives. In Section 3, some contributions and published works will be discussed. In Section 4 we present some conclusions. The last section points out some ongoing and future works.

2 Research objectives

Regarding the issues describe in the earlier section, in this paper we describe our research towards the following directions:

(1) Tool Support: an SPL approach

In the last few years, several extensions of the modeling language based on the *i** / Tropos framework have been proposed, due to the specific needs of various research groups, eg. [3, 4, 17]. However, building a suitable tool support for each one of these extensions leads to a high development cost. In the mean time, the Software Product Line – SPL paradigm [13] has gained significant popularity in the software industry and academia. It promotes software reuse by specifying a family of software products through artifacts capturing their common and variable features. Thus, we aim to use their principles to provide a set products, i.e. specific goal modeling tools, to support different versions of *i**/Tropos. Each tool will be configured according to a set of specific features related to chosen modeling language [15].

(2) MAS Reuse: An SPL approach

Tropos [4] is considered one of the most complete agent oriented methodologies, since it spans all stages of multi-agent systems development. Since the initial proposal, in 2000, various versions and extensions have been proposed. However, these proposals have adopted different activities and notations, decreasing their adoption by software developers [20]. Our goal is to extend the Tropos process to enable the development of multi-agent systems according to the SPL approach. Hence, we need to add *Domain Engineering* and *Application Engineering* phases. Moreover, some form of *Feature Modeling* and *Configuration Knowledge* may also be required.

3 Contributions

In this section we describe some of the ongoing work of our research group, related to the fulfillment of the research goals presented in Section 2.

(1) Tool Support: an SPL approach

Due to the diversity of goal modeling languages based on *i**, we needed to identify the common and variable constructors present in the several *i*/Tropos* extensions/dialects. Our purpose is to develop a product line of tools that can be easily configured to support any of the analyzed extensions. Hence, inspired by the Software Product Line paradigm [13] we are defining Core Assets, as well as Domain Engineering and Application Engineering Phases. Based on common *i*/Tropos* constructors, we proposed a core metamodel to support the goal modeling variability [10]. Depending on the language we want to use, different constructors could be inserted in the core metamodel, producing a new metamodel for a specific *i** extension. As a first result, we developed a version of our *i** modeling tool - called iStarTool [15] – which currently supports the original version of *i** [22].

(2) MAS Reuse: An SPL approach

Initially we considered best practices of the several Tropos approaches. This resulted in U-Tropos: a proposal for an unified process to develop agent oriented software [21].

In the current phase of this work, the SPL technology [13] has been investigated as an alternative to promote reuse in agent oriented systems development. In this context, we examined how goal modeling languages could be used to support product line variability. In particular we tried to relate goal models to feature models [3].

Goal oriented requirements engineering (GORE) can be used to discover variable and common requirements in a software product line (SPL), as well as to reduce costs related to the configuration of a specific product in such product family. Recently, a comparison among some GORE approaches to deal with software variability has pointed out that they have limited expressivity to represent variability in SPL, as presented in [3]. This has motivated us to investigate the use of i* framework as a GORE approach for SPL. The work presented in [19] proposes an extension of the i* modeling language, called **i*-c** (*i* with cardinality*), which allows the insertion of cardinality in some of their modeling elements. The **G2SPL** (*Goals to Software Product Line*) approach proposes a process to identify and model common and variable requirements in a SPL using i*-c models. This approach also guides the configuration of a specific product in a SPL.

4 Future and Related works

In this section we present future works we plan to perform in order to achieve the goals described in Section 2. We also present some related works.

(1) Tool Support: An SPL approach

Also based on SPL concepts [13], we intend to use the core *i** metamodel, the extended *i** metamodels and their identified variabilities, to create and configure a family of tools to support goal modeling. We expect that this solution will improve the maintainability and extensibility of the current and future tools. The next product will support the *i** wiki version. Later aspectual *i** [17] will also be incorporated in the product line. We also envisage support for *i**-c (i* with cardinality)[3].

(2) MAS Reuse: An SPL approach

It is intended to extend the U-Tropos Process to include Domain and Application Engineering, for the development of agent based software using the methods and techniques of SPL. This new version of Tropos will be called Tropos-SPL (Tropos Software Product Line).

(3) Related works

We are also proposing an approach that combines variability analysis and non-functional requirements to drive the configuration of a business process. Applying this approach we can analyze variability in the model in order to assess the impact of the choices on the process quality constraints - the non-functional requirements. Moreover, it provides a rationale for the selection of a specific configuration and could support the variability representation in business process [23].

Lastly, we are using *i** models as a basis for identifying situations in which certain failures may be ignored [**¡Error! No se encuentra el origen de la referencia.**]. This work is being developed in the context of self-configurable systems, in which each failure would lead to a compensation.

References

1. Ali, R., Dalpiaz, F., Giorgini, P.: A goal modeling framework for self-contextualizable software. Book Chapter. In: Lecture Notes in Business Information Processing, vol. 29 (Enterprise, Business-Process and Information Systems Modeling), p. 326-338, 2009.
2. Amyot, D., Horkoff, J., Gross D., Mussbacher, G.: A Lightweight GRL Profile for *i** Modeling. In: Proceedings of Third International Workshop on Requirements, Intentions and Goals in Conceptual Modeling – RIGiM'09, Advances in Conceptual Modeling – Challenging Perspectives. LNCS 5833, p. 254-263, 2009.
3. Borba, C., Silva, C.: A Comparison of Goal-Oriented Approaches to Model Software Product Lines Variability. In: Proceedings of Third International Workshop on Requirements, Intentions and Goals in Conceptual Modeling – RIGiM'09, Advances in Conceptual Modeling – Challenging Perspectives. LNCS 5833, p. 244-253, 2009.
4. Castro, J., Kolp, M., Mylopoulos, J.: Towards Requirements-Driven Information Systems Engineering: The Tropos Project. In: Information Systems Journal, vol. 27, p. 365-389, 2002.
5. Cruz Neto, G., Gomes, A.S., Castro, J.: Mapping Activity Theory Diagrams into *i** Organizational Models. In: Journal of Computer Science and Technology, vol. 5, p. 57-63, 2005.
6. Franch, X.: A Method for the Definition of Metrics over *i** Models. In: Proceedings of 21st International Conference on Advanced Information Systems (CAiSE'09), 2009.

7. Grau, G., Franch, X., Maiden, N.: PRiM: An *i**-based process reengineering method for information systems specification. In: Journal of Information & Software Technology, vol. 50, p. 76-100, 2008.
8. Lapouchnian, A., Mylopoulos, J.: Modeling domain variability in requirements engineering with contexts. In: Proceedings of 28th International Conference on Conceptual Modeling (ER'09). LNCS 5829, p. 115-130, 2009.
9. Lucena, M.J.N.R., Castro, J., Silva, C.T.L.L., Alencar, F.M.R., Santos, E.B., Pimentel, J.H.C.: A Model Transformation Approach to Derive Architectural Models from Goal-Oriented Requirements Models. In: Proceedings of 8th International Workshop on System/Software Architectures (IWSSA'09). LNCS 5872, p. 370-380, 2009.
10. Lucena, M.J.N.R., Santos, E., Silva, M.J., Silva, C.T.L.L., Alencar, F.M.R., Castro, J.F.B.: Towards a Unified Metamodel for *i**. In: Proceedings of Second International Conference on Research Challenges in Information Science (RCIS'08), 2008.
11. Pimentel, J.; Santos, E.; Castro, J. Conditions for ignoring failures based on a requirements model. In: Proceedings of the 22nd International Conference on Software Engineering and Knowledge Engineering (SEKE), 2010, in press.
12. Pimentel, J.H.C., Borba, C., Xavier, L.: BTW: if you go, my advice to you Project. July, 2009. Web-site. In: https://jaqueira.cin.ufpe.br/jhcp/docs/ (last access, Feb 2010).
13. Pohl, K., Bockle G., Linden, F.V.: Software product line engineering. Springer: Verlag, Berlin, Heidelberg, 2005.
14. Ramos, R.A.: AIRDoc - An Approach to Improve the Quality of Requirements Documents: Dealing with Use Case Models. PhD Thesis. Federal University of Pernambuco, 2009.
15. Santos, B. S., Paes Junior, J.: iStarTool. 2009. Web-site. In: http://portal.cin.ufpe.br/ler/Projects/IStarTool.aspx (last access, Feb 2010).
16. Santos, E. B.: A Proposal of Metrics to evaluate *i** Models (in Portuguese: Uma Proposta de Métricas para Avaliar Modelos *i**). MSc Dissertation. Federal University of Pernambuco, 2008.
17. Alencar, F., Castro, J., Lucena, M., Santos, E., Silva, C., Araújo, J., Moreira, A.: Towards Modular *i** Models. In: Proceedings of Requirements Engineering Trank, 3th Ed., 25th ACM Symposium on Applied Computing, 2010.
18. SCORE 2009. Web-site. http://score.elet.polimi.it (last access, Jan 2010).
19. Silva, C., Borba, C., Castro, J.: G2SPL: A Goal Oriented Requirements Engineering Process for Software Product Line (In Portuguese: G2SPL: Um Processo de Engenharia de Requisitos Orientada a Objetivos para Linhas de Produtos de Software). In: Proceedings of 13th Workshop on Requirements Engineering (WER), 2010 (to appear).
20. Silva, M.J., Maciel, P., Pinto, R., Alencar, F., Tedesco, P., Castro, J.: Extracting the Best Features of Two Tropos Approaches for the Efficient Design of MAS. In: Proceedings of X Iberoamerican Workshop on Requirements Engineering and Software Environment (IDEAS'07), p. 3-16, 2007.
21. Silva, MJ.: U-TROPOS: an unified process approach for agent oriented software development (In Portuguese: U-TROPOS: uma proposta de processo unificado para apoiar o desenvolvimento de software orientado a agentes). Msc dissertation. Federal University of Pernambuco, 2008.
22. Yu, E.: Modelling Strategic Relationships for Process Reengineering. PhD thesis. University of Toronto, 1995.
23. Santos, E. B., Pimentel, J. H., Castro, J., Sanchez, J.: Configuring the Variability of Business Process Models Using Non-Functional Requirements. In: Proceedings of 15th International Conference on Exploring Modelling Methods for Systems Analysis and Design (EMMSAD'10).
24. Montaner, M., López, B., De la Rosa, J. A Taxonomy of Recommender Agents on the Internet. In Artificial Intelligence Review, June 2003.

Requirements Engineering for Control Systems

Dominik Schmitz[1], Hans W. Nissen[2], Matthias Jarke[1,3], and Thomas Rose[1,3]

[1] RWTH Aachen University, Informatik 5, Ahornstr. 55, 52056 Aachen, Germany
{schmitz,jarke}@dbis.rwth-aachen.de
[2] Cologne University of Applied Sciences, Institute of Communications Engineering, Betzdorferstr. 2, 50679 Köln, Germany hans.nissen@fh-koeln.de
[3] Fraunhofer FIT, Schloss Birlinghoven, 53754 Sankt Augustin, Germany
thomas.rose@fit.fraunhofer.de

Abstract. In this paper, we report on the application of i^* to the combined capture of control system and software requirements in the context of software-intensive controllers for engines in the automotive domain. Our work has revealed the need to explicitly represent concrete domain knowledge. Revolving around the notion of "domain models", several contributions have been made: a domain model-based approach to requirements capture to speed up the modeling process, a model-based similarity search to support reuse, advanced support to cope with the evolution of domain knowledge (and thus domain models), and the integration into the further development by establishing a transformation link toward mathematically-founded tools such as Matlab/Simulink.

1 Introduction

Control system functionality, for example in cars, increases the comfort and safety of driving a car or reduces the fuel consumption and exhaust gas emissions. Experiences and knowledge in physics, mathematics, and control theory are required to design a stable controller with good performance. While for many years the control systems for vehicle engines were designed solely by control engineers, in the last decade it has been recognized that massive reductions in pollution and gas consumption as well as advanced driver assistance systems can only be realized if software-based controls are embedded in these systems. However, control systems development continues to be different from software systems development. In the following we shortly present some major differences that had an impact on our work.

In industrial practice, the development process is still mainly driven by control system engineers. They design the platform and architecture purely driven by functional considerations. Software engineers are involved only at the implementation phase to efficiently implement the control algorithms. The software engineers reject this approach and argue that a system's structure should follow from the consideration of non-functional requirements (NFRs) in order to implement safe, reusable, and efficient systems. NFRs are currently to a large degree ignored by control system engineers.

Interestingly both disciplines claim to pursue model-based approaches but with a quite different understanding of the main concepts. For control system development, the model of the controlled system, e. g. the engine, is at the center of interest and a model is always expected to be executable in mathematical tools such as Matlab/Simulink. In contrast to this, within software engineering models usually describe the system to be developed. In addition, whereas the design is entirely model-based, at the level of requirements textual approaches still prevail in the control systems domain. Software engineers on the other hand prefer model-based requirement specifications, in particular goal-based, to enable a better structuring, traceability and a smarter transition from requirements to subsequent development steps.

Eventually, in the control systems development sector, small- and medium-sized enterprises (SMEs) play an important role as innovation drivers that perform individual engineering tasks for multiple customers. Their development process is typically initiated by a customer who asks for the development of a controller for a new engine. The time frame for the supplier to respond with a competitive offer is very short. After capturing the requirements from a developer's point of view, a first system design is needed in order to estimate costs. To keep the development costs low and to win the contract, the supplier must reuse as many software artifacts and simulation models as possible from previous projects. But, if after winning the contract in later design phases it is discovered that the selected components are in fact not reusable, their new development may result in a project loss. Thus, a very careful investigation has to take place.

2 Objectives of the Research

The core aim of the ZAMOMO project "Integrating model-based software and model-based control systems engineering" is to improve the interaction of control engineers and software engineers. In particular, interdisciplinary issues – the lack of mutual understanding, colliding uses of terminology, the strict separation of the development processes – need to be addressed. Furthermore, the model-based development of controllers needs to be completed in regard to model-based requirements engineering, while accounting for some particularities of control systems such as the importance of sensors and actuators. The modeling formalism should include means to cope with non-functional requirements as they have received insufficient attention during control system development yet. Eventually, control system development is indeed a very customer- and project-oriented business. Although similar on an abstract level, engines always differ in detail, thereby precluding long-term planning of product lines due to the individuality of the developed solutions. Accordingly, a project-oriented approach supporting a fast and reliable identification of reusable components must be established. Furthermore, there is a high frequency of innovations in this field. The knowledge changes and grows quite fast. With each new development project, new engine components, sensors, actuators, and construction styles may arise. The according knowledge must fast and easily be made available to the developers.

3 Scientific Contributions

Combined Investigation of Control and Software Requirements We propose *i** as a common notation for control system and software requirements [4]. The few and simple modeling constructs, in particular "goals" and "agents", address interdisciplinarity. The model-based approach fills the gap in the otherwise already entirely model-based development of control systems. Softgoals allow to consider non-functional requirements explicitly. And also the important concepts "sensor" and "actuator" can be represented suitably (via resource dependencies).

Requirements Specification Based on i Domain Models* To address the need for fast requirements capture, we propose to establish a specific domain *i** model reflecting the knowledge and experiences in a particular field the SME is specialized in. Certainly, it is up to the SME to introduce separate models for different (sub)fields it is active in. A domain model serves as a suitable starting point for the creation of a problem-specific requirement model [5]: the engineer eliminates the parts from the model that do not apply for the current project and adds new elements that are specific to the project at hands. This way rapidly a requirements model of the new control problem can be established. It is composed of reused parts from the domain model and project-specific extensions.

Similarity Search To support a competitive and reliable cost calculation a similarity search is provided that helps identifying similar projects and hence reusable components [5]. Unfortunately, a fully automated identification of reusable software artifacts is not possible due to the complexity and variance in details. But our domain model-based search algorithm reduces significantly the number of finalized projects the engineer has to inspect in detail. The domain model forms a necessary premise for the search since it ensures consistency of models across several projects. For the technical realization, we refer to the formalization of *i** in Telos and the corresponding tool support ConceptBase [1]. This allows to define comparison queries referring to standard domain features as well as project-specific model extensions. The comparison of the outcome of these queries for the current project with the outcome for finalized earlier projects results in a ranking of the finalized projects based on the number of similar features. The engineer can then focus the higher ranked projects and investigate them in detail to decide about reusability.

Support for Evolving Domain Models The usefulness of a domain model depends heavily on its adequacy for the day-to-day work of the engineers [3]. Neither overly large nor too small domain models are helpful. In the first case, the need to delete large portions of the modeling jeopardizes the advantages in regard to a fast requirements capture. Similarly, a too small model slows down the process by requiring to model similar details over and over again. The latter also adds to avoidable inhomogeneity of the modeling. Instead, a domain model must suitably and continuously be tailored to the particular needs of the SME. Advances in technology can easily be adopted by simply reflecting the findings via modifications of the domain model. But the more interesting changes result from

the SME's individual experiences within customer projects. If a certain project-specific extension has been added several times or if parts of the domain model have always been deleted within the recent past, these are obviously good candidates for extensions and reductions of the domain model, respectively. While reductions can be identified quite easily, the detection of similar project-specific extensions is more complicated. A first heuristic compares the "anchor objects" of a project-specific extension in the domain model for different projects [2]. Anchor objects are the modeling objects of the domain model to which the project-specific extension is connected. After adopting such a project-specific extension into a domain model, we provide measures to reestablish the accuracy of the similarity search.

Transformation of Requirements Models to Later Development Phases By again building on the formalization in Telos, partially automated support for the transformation to Matlab/Simulink is provided [6]. After manually resolving design alternatives, a Matlab/Simulink skeleton model is generated from the i^* model. Since the conceptual model behind Simulink models is rather simple (block diagrams), the matching of concepts is straight forward. Most importantly, various i^* relationships are mapped on the nesting of corresponding "system" blocks. The mapping can interactively be improved by incorporating existing hardware and platform components from SME specific Matlab libraries.

4 Conclusions

The feedback from control engineers both from academia as well as industry within the project context has been very encouraging. The control engineers got rather fast familiar with the requirements representation and saw advantages due to the broader span of issues that is representable in i^* compared to their specific formalisms, e. g. block diagrams. The industrial partner pointed out the unsatisfactory maturity of the tool support. In particular, they miss a clear guideline when to apply which modeling construct and how to cope with really large i^* models. The domain model related support facilities have been very much embraced, maybe in particular since they provide a kind of such guidance. Furthermore a domain model allows an SME to capture consolidated and specific engineering knowledge originating in former customer projects. Together with the similarity search this provides a means to support reuse and to cope with variability while still remaining flexible, innovative, and in particular customer- and project-oriented at the core.

5 Ongoing and Future Work

The ideas on how to support the evolution of domain models have just been started in [2]. From our current experiences we expect that the proposed heuristic to detect similar project-specific extensions (based on anchor objects) needs to be combined with several other heuristics to provide for sensible suggestions. Text

related issues as well as heuristics that take *i** structural modeling information into account are conceivable.

Furthermore, to match with the importance of simulation during the later control system development, simulation means at requirements level have to be established. Also the characteristics and features of *i** in particular in regard to the sociality of actors needs to be closer investigated in the context of this more technical setting where most actors do not represent humans but artificial components.

Eventually, the application of the domain model based requirements engineering approach has been exemplified here for the field of control systems. While a concrete domain model is as a matter of course domain specific, we assume that in many other engineering disciplines with similar characteristics as control systems development – customer-oriented development projects, high enforcement of reuse, high frequency of innovations – the basic ideas behind our approach are applicable as well. Targeted examples are access control and burglary warning systems for buildings or the construction and set-up of flexible automated manufacturing systems. The claim for a broader applicability of the proposed domain-model based approach needs to be confirmed in additional case studies, for example, in the above mentioned fields.

Acknowledgment. This research was in part funded by the German Ministry of Education and Research (BMBF) on the project ZAMOMO, grant 01 IS E04. Thanks to our project partners Dirk Abel, Peter Drews, Frank J. Heßeler, Stefan Kowalewski, Jacob Palczynski, Andreas Polzer, and Michael Reke.

References

1. M. A. Jeusfeld, M. Jarke, and J. Mylopoulous, editors. *Metamodeling for Method Engineering.* MIT Press, 2009.
2. H. W. Nissen, D. Schmitz, M. Jarke, and T. Rose. How to keep domain requirements models reasonably sized. In *2nd Int. Workshop on Managing Requirements Knowledge (MaRK)*, pages 50–59, Atlanta, USA, 2009. IEEE.
3. H. W. Nissen, D. Schmitz, M. Jarke, T. Rose, P. Drews, F. J. Hesseler, and M. Reke. Evolution in domain model-based requirements engineering for control systems development. In *17th Int. Requirements Engineering Conference*, pages 323–328, Atlanta, USA, 2009. IEEE.
4. D. Schmitz, P. Drews, F. Hesseler, M. Jarke, S. Kowalewski, J. Palczynski, A. Polzer, M. Reke, and T. Rose. Model-based requirements capture for software-based control systems (in German). In *Software Engineering, Feb. 18-22*, LNI P-121, pages 257–271, Munich, Germany, 2008.
5. D. Schmitz, H. W. Nissen, M. Jarke, T. Rose, P. Drews, F. J. Hesseler, and M. Reke. Requirements engineering for control systems development in small and medium-sized enterprises. In *16th Int. Requirements Engineering Conference*, pages 229–234, Barcelona, Spain, 2008. IEEE.
6. D. Schmitz, M. Zhang, T. Rose, M. Jarke, A. Polzer, J. Palczynski, S. Kowalewski, and M. Reke. Mapping requirement models to mathematical models in control system development. In *5th Europ. Conf. Model Driven Architecture (ECMDA-FA)*, LNCS 5562, pages 253–264, Enschede, The Netherlands, 2009. Springer.

*i** on ADOxx®: A Case Study

Margit Schwab[1], Dimitris Karagiannis[1], Alexander Bergmayr[1]

[1] Universitaet Wien, Faculty of Computer Science,
Department of Knowlede and Business Engineering,
Bruennerstrasse 72, 1210 Vienna
{ms, dk, ab}@dke.univie.ac.at

Abstract. Based on the ADOxx® meta-modelling platform, the conceptualization of the *i** method is discussed by means of a case study. The focus lays on the "translation" of *i** concepts into a conceptual model leveraging the instantiation of meta-classes provided by the utilised ADOxx® platform. Thereby the consideration of all concinnities of both the *i** meta-model and corresponding instance models within a specific domain is essential. The claim is that the ADOxx® platform supports this with adequate abstraction mechanisms. When using a meta-modelling platform, the first step of semantic integration can be achieved, where the modelling language and the platform use a meta-modelling approach as a concept. The result of this case study is accessible on www.openmodel.at/istar.

Keywords: meta-model, modelling language, *i**, ADOxx®, meta-modelling platform, method-engineering and -engineer;

1 Introduction

The construction of models and by their means processing particular information is nowadays a common procedure. Depending on the domain, the purpose and the underlying meta-model, the resulting instance models are "by definition" more or less complex. We use the classification of model hierarchies and language levels according to Kühn [11, p 32]. As we speak of instance models we think of graphical models and refer to the distinction of different types of models [9, 10, 8]. The end user of the meta-model, let's call him/her modeller, will use the modelling method at hand ideally in terms of the method engineer. In addition s/he will shape and refine the information to be conveyed with the instance model in the best possible way. This task is in general at least twofold.. Firstly, there is the design of the model and secondly, there is nearly at all times the need to consider further concinnities, like additional model descriptions in natural language, time-related data, necessary skills during processing, or applicable forms or regulations of the model. In fact the effort spent for the latter varies depending on the purpose and the target group the instance model is designed for. This demands flexibility from the underlying meta-modelling platform, in concrete for the case study ADOxx®. Although there are already a number of solutions available providing an implementation of this method [2], the crucial distinction feature in this study case is that the design and realization of the *i** method [6, 16, 19] is based on a meta-modelling

approach, as described in Section 2. Section 3 is devoted to lessons learned. Section 4 concludes the paper and gives an outlook on further research questions to be addressed.

2 The *i** Method Case Study

In this case study the *i** method on one hand and the ADOxx® meta-modelling platform on the other hand have been used. The former provides a specification for the syntax, the semantic and the notation. The method concepts: *actor, role, agent* and *position* are subsumed under the term "intentional actor". Furthermore there are the elements *goal, softgoal, task, resource* and *belief*. These elements form the group "intentional elements". Connections comprise the constructs of a *dependency link*, *association link*, *means-end link*, *decomposition link* and *contribution/correlation link* [6, 19]. On the other side the ADOxx® supports the process of method customization and allows the - mostly graphical - creation, persistence, maintenance and usage of models. Further it offers functionality to freely define and configure arbitrary meta-models of modelling languages including the definition or adaptation of the corresponding procedures and mechanisms applicable to models. By means of these tools the method engineer elaborates the translation of the *i** method into a conceptual model. The result of this development is a semi-formalised structure of the available modelling concepts and their dependencies. For the construction of graphical models the *i** method also provides integrated mechanisms, especially to perform goal satisfaction evaluations, based on these models. For these mechanisms further requirements related in particular to the notation of the modelling concepts can be specified during the developing phase by the method engineer. In the following the focus lays on the elaboration of the *i** conceptual model, in the translation part and the customization concepts, in the instantiation part [1].

2.1 Part I: The Translation

The starting point for the translation is the ADOxx® **metameta-model** which exhibits the structure for the matching of *i** concepts. In the following selected concepts of this metameta-model are used to exemplarily demonstrate mappings between the method and the platform. The method engineer has to know these concepts in order to fulfil the intellectual process: **to design a holistic view of the *i** conceptual model** (see Fig.1.).

The first applicable concept is **library.** The library is a container to which all formalisms and constructs of an instance of a modelling language are assigned to. Yet, the meta-model possesses also a particular structure so that the assigned elements are not loosely arranged abreast on an equal level. The next step is to allocate the constructs of the modelling language to **model types.** The *i** method comprises two different types of models, the *strategic dependency model* and the *strategic rationale model*. The model type is a modularisation element for the available modelling concepts of a method. Hence, a model type strategic dependency model groups for example all modelling concepts necessary to map strategic dependencies for a particular scenario.

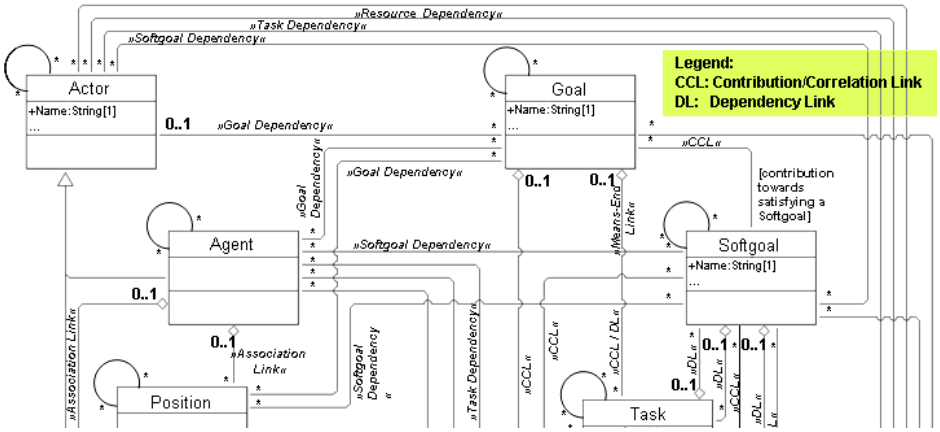

Fig. 1. Holistic View of the *i** Conceptual Model [part]

The modelling concepts are described with **classes** which are assigned to a particular model type. In ADOxx® **modelling classes**, which are in the *i** method, *actor, agent, role etc.* and for the **relation classes**, which are *dependency link, association link etc.* are distinguished.The different classes have particular properties. In this point the *i** method gives the method engineer an opportunity of a precise formal description about the syntax of the classes by means of **attributes.** The only mandatory requirement of the platform is that each class, modelling class or relation class, has a **name attribute**, because technically speaking, it becomes a global identifier. We distinguish between class attributes and instance attributes. The differene between these two lay in the values the attribute can adopt. Class attributes are context neutral and not to be filled by the end user or modeller using the method after implementation. Instance attributes are context dependent and will be used by the modeller to capture data and convey certain information [11, p. 100]. The **attribute type** is determined by the value the attribute can adopt when using the method, i.e. which data should be captured. Beside commonly known datatpyes ADOxx® additionally provides support for inter model references, expressions, tables, or programm calls to name some of them. This list can be extended for applying the algorithms and mechanisms on the instance models.

2.2 Part II: The Instantiation

The instantiation should lead to a mapping between the ADOxx® meta-classes with the *i** modelling and relation classes. After this step the customizing effort can be determined. The customizing is conducted on the level of the modelling language - considering the notation, the syntax and the semantic - on the method procedure level as well as from the processing point of view within mechanisms and algorithms. For demonstrating the work which is involved is shown in an example concerning the customizing of the notation.

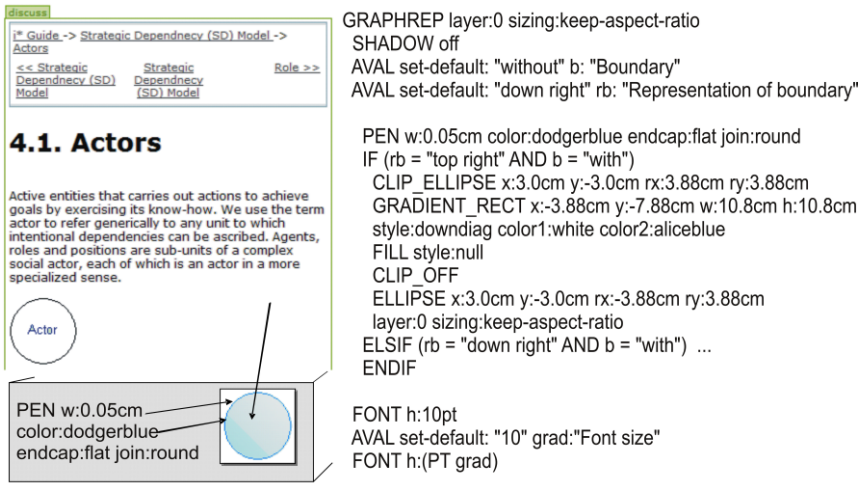

Fig. 2. Actor *i** Notation: Graphrep Representation

The *i** method gives guidelines how the different classes look. It specifies the shape. Furthermore that for the modelling classes the value of the name attribute is displayed in the centre of the shape. The intentional actors can furthermore possess a *boundary*. The intentional actors are represented with a boundary, to express that all intentional elements within this boundary are explicitly desired by that intentional actor. For the relation classes association link and contribution/correlation link several type of links, for example if the association link is a "covers", "plays" etc. association, are specified. They are visualised with a respective label. This requirement can be fulfilled with the **Graphrep** formalisms which are provided by ADOxx®. Fig.2. gives an example of the actor specification and the realization in the ADOxx® Graphrep. In analogy, the customization effort for syntactical and semantical requirements is fulfilled through ADOxx® functionality. The version of the *i** method which has been translated and customized on the ADOxx® platform as described in the case study at hand is available on the open models platform.

3 Some Lessons Learned

The *i** method provides a high degree of maturity in terms of method specification. Nevertheless, the mapping on the meta-modelling platform ADOxx® showed that the offered platform functionality gives new input to further conceivable extensions. Suggestions for extensions rely on the analysis of instance models provided in different papers and can be structured by *extensibility* for syntax and notation as well as by *interpretability* [16, 17].

Extensibility. The syntax is related to the notation as some attributes are only necessary for the "orchestration" of a certain graphical representation, for example that the actor is represented with a boundary. The actor boundary belongs to a very specific actor and if a boundary is required to express the delineated semantic, it should be

possible that the modeller can activate and deactivate the boundary on the drawing area as needed. In ADOxx® this can be done by defining an attribute, e.g. boundary and two possible predetermined values 'with' or 'without'. Another extension concerns the colour as it is an important distinction element, to keep the available shapes and as a consequence their meaning apart, there is the suggestion to use coloured elements. The reason for this is that if the instance models are of a certain size they tend to become hard to overlook and to read. From the experience of working with end users and addressees of instance models we know that the rectilinear an instance model is mapped, the easier the reader picks up the content and the better s/he understands the scenario captured with it.

Interpretability. During the analysis phase of instance models it became obvious, that the two previously identified model types – strategic dependency model and strategic rational model – are candidates for applying the ADOxx® view concept. As both types use most of the available classes of the *i** method mutually. The latter refines the former entirely or only in parts by using the same instance objects. Hence, what in the *i** method is expressed by two different types of models is considered as a view or **mode** in the meta-model of the ADOxx® platform. Despite of using the mode concept at least one model type needs to be defined. The suggestion for a name of the newly specified model type is ***Intentional actors and elements model.*** As the name should be specific for the *i** method and should convey as precisely as possible the context that should be documented with a strategic dependency model and a strategic rational model. The new model type has two modes, once the strategic dependency mode and twice the strategic rational mode whereas the former is defined as the default mode.

The explained lessons learned are concerning the conceptualization of a particular method. The deployment of the instantiated method is beyond the focus of the case study, although there are related technical questions the method engineer needs to answer in order to provide a "ready-to-use" tool. The ADOxx® platform offers respective support for this task.

4 Conclusions

Instance models and with them the modelling method they have been created with are commodity, this is also valid for the *i** method. There will always be the need for new methods or extended functionality of existing ones in order to convey information in the intended way. Once modelling methods are to be used by a broader group of people and drift off the initial inventing team it becomes advantageous if the method is supported by a tool. The more the modelling method avoids ambiguity in expressing certain information the more difficult it is to "translate" this modelling method and support it by a platform. From our experience one reason for that is that today's meta-modelling platforms lack in providing the full range of required functionality. Further research on the side of meta-modelling platforms and furthermore on the end user side is essential. This even more if this is seen in context of the integration of different modelling methods. For the later this is a claim in form of a non-functional requirement with regards to the utilisation of the modelling method by the end user resulting in user-friendly handling and as a consequence user acceptance [5, 18].

Acknowledgments. The authors thank all the colleagues from the University of Vienna and the Open Models Community for helpful discussions and comments during the realization of this case study.

References

1. Karagiannis, D.; Kühn, H.: ”Metamodelling Platforms”; in Proceedings of the Third International Conference EC-Web 2002 – Dexa 2002, Aix-en-Provence, France, September 2 – 6, 2002, LNCS 2455, Springer Verlag, Berlin Heidelberg, p 182 ff.
2. List of *i** tools, http://istar.rwth-aachen.de/tiki-index.php?page=i*%20Tools, last access 5th of March 2010.
3. Cares, C., Franch, X., Perini, A., Susi, A.: „iStarML The i* Mark-up Language: Reference‘s Guide; Barcelona, Spain, August 2007.
4. Franch, X., Grau, G.: “Towards a Catalogue of Patterns for Defining Metrics over i* Models”, CAiSE 2008, Springer-Verlag Berlin Heidelberg 2008, LNCS 5074, pp. 197–212.
5. Mylopoulos, J., Chung, L., Yu, E.: “From Object-Oriented to Goal-Oriented Requirements Analysis”, Communications of the ACM, Vol. 42, No. 1, January 1999.
6. Yu, E.: “Strategic Actor Relationships Modelling with i*, Part 1, Part 2, Part3, A tutorial given at IRST / University of Trento, Italy, December 2001; http://www.cs.toronto.edu/~eric/#istar-tut-ppt; last access 25th of February 2010.
7. Open Models Initiative, http://cms.dke.univie.ac.at/uploads/media/Open_Models_Feasibility_Study_SEPT_2008.pdf; last access 25th of February 2010.
8. Open Models Initiative; http://www.openmodels.at/web/istar/1-5; last access 25th of February 2010.
9. Harel, D., Rumpe, B.: “Meaningful Modeling: What's the Semantics of 'Semantics'?”. In: IEEE Computer, Vol. 37 No. 10, 2004, 64-72.
10. Strahringer, S.: Metamodellierung als Instrument des Methodenvergleichs. Shaker, Aachen, 1996.
11. Kühn, H.: “Methodenintegration im Business Engineering”; Dissertation, Universität Wien, April 2004.
12. Braun, C., Wortmann, F., Hafner, M., Winter, R.: “Method Construction – A Core Approach to Organizational Engineering”, in Applied Computing 2005, Proc. of the 2005 ACM Symposion on Applied Computing, Santa Fe, New Mexico, USA, 13.03.2005, ACM Press, New York, NY, USA, 2, 2005, pp. 1295-1299.
13. Tan Jun: “Software Develop Method Construction A Kernel Approach to Organizational Engineering”. In Software Engineering WCSE '09, WRI World Congress on Software Engineering, Volume 4, May 2009.
14. Becker, J.; Holten, R.; Knackstedt, R.; Neumann, S.: Konstruktion von Methodiken – Vorschläge für eine begriffliche Grundlegung und domänenspezifische Anwendungsbeispiele, Institut für Wirtschaftsinformatik, Universität Münster, 2001.
15. Software Engineering Institute, SEI, “Introduction to the Architecture of the CMMI® Framework”, http://www.sei.cmu.edu/reports/07tn009.pdf, last access 5th of March 2010.
16. Fazel-Zarandi, M., Yu, E.: “Ontology-Based Expertise Finding”, In Proceedings of the 7th International Conference of Practical Aspects of Knowledge Management, Yokohama, Japan, 2008. Springer-Verlag, Berlin Heidelberg (2008), pp. 232-243
17. Samavi, R., Yu, E. Topaloglou, Th.: “Strategic Reasoning about Business Models: A Conceptual Modeling Approach”, Information Systems and E-Business Management, Springer , Berlin / Heidelberg, Vol. 7, No. 2, March 2009.
18. Mylopoulos, J., Chung L., and Nixon, B., "Representing and using Non-functional Requirements: A Process-oriented Approach", IEEE Transactions on Software Engineering, June 1992.
19. Yu, E. S.: “Social Modeling and i*”, in “Conceptual Modeling: Foundations and Applications”, Borgida, A. T., Chaudhri, V. K., Giorgini, P., Yu, E. S. (Eds.): Mylopoulos Festschrift, LNCS 5600, Springer Verlag, Berlin Heidelberg, June 2009, p 99.

Deriving Adaptive Behaviour from *i** models

Kristopher Welsh, Pete Sawyer

Lancaster University, Computing Dept., Infolab21 LA1 4WA Lancaster, UK
{k.welsh, p.sawyer}@lancs.ac.uk

Abstract. Dynamically Adaptive Systems (DASs) adjust their behaviour at run-time to tolerate changes in context. With potentially incomplete or inaccurate knowledge of an operating environment, tailoring specific behavioural adjustments to individual contextual changes is a time-consuming and error-prone process. i* models of a DAS' behavioural adjustments can aid understanding, and can form part of the specification of the DAS' adaptive behaviour; speeding the specification process. This paper presents an approach by which a DAS' adaptive behaviour may be derived directly from a set of i* models, and a tool capable of performing the derivation automatically.

Keywords: Dynamically Adaptive Systems, Istar, Model-Derived

1 Introduction

Dynamically Adaptive Systems monitor changes in their operating environment, and re-configure to better suit prevailing conditions. We have previously [1] likened this to the balancing of conflicting softgoals as the environment dictates changes in what constitutes an acceptable balance between them. DASs commonly utilise some form adaptive middleware (for example [2]), which separates the concerns of environmental monitoring, adaptation planning and effecting component substitution from the DAS' business logic. Adaptive middleware codifies system configurations (as combinations of components) and the conditions under which they are adopted in adaptation *policies*. These can be thought of as re-statements of decisions made after analysis of the environment, and of the available system components during the Requirements Engineering process.

Our existing LoREM process [3] offers an i*-based modelling approach for DASs operating in environments that can be partitioned into distinct *domains*. An individual steady-state system, termed a *target system* is derived for each domain, as conceptualised in [4]. From a modelling perspective, each target system is as complex as a traditional, non-adaptive system developed for the domain, with the key artefact to emerge from the modelling process being a selection of components to be adopted in each domain. This paper explores a method by which these selections of components, (modelled in i* SR diagrams) combined with i* models of environmental partitions can be used to derive the policies used by the system's adaptive middleware to control the DAS' adaptive behaviour.

2 Objectives of the Research

Given our belief that adaptation policies re-state the results of component selection decisions taken during the RE process, and that those decisions are recorded in i* models in LoREM, the research has three aims: **1.**To establish a link between the i* based environmental modelling carried out as part of the LoREM process and DAS adaptation policies. **2.** To develop an approach by which the adaptation policies can be derived from the i* models and **3.** To automate the process.

Understanding the link between the LoREM i* models and a DAS' final adaptation policies is important in pinpointing modelling deficiencies and performing maintenance on the DAS. Given the complex nature of the environments for which a DAS will typically prove most useful, understanding may be incomplete or inaccurate. Being able to trace a sub-optimal or failing target system back to the environmental analysis is crucial to a time and cost efficient evolution process.

The ability to derive adaptation policies directly from LoREM models could reduce the time and effort spent in their development. DASs are hugely complex systems, and any support mitigating some of this complexity is beneficial.

Automating the derivation process not only saves time and helps to reduce the possibility of errors being made during policy derivation, but opens up an interesting possibility: the DAS may be able to perform the derivation itself. A combination of this ability, requirements monitoring and models@runtime [5] could allow a DAS to perform some limited self-maintenance, in the form of correcting modelling deficiencies in response to monitored data, and re-creating its adaptation policies based on the updated model. This possibility is discussed further in section 5.

3 Scientific Contributions

The LoREM process involves creating i* SR models of each target system, illustrating the degree to which it satisfices the DAS' softgoals in each domain. These models are known as level 1 models. Level 2 models, show how the DAS' adaptive infrastructure monitors the environment, plans and effects adaptation for each valid transition between target systems. Level 3 models aid in the selection of the adaptive infrastructure, but are beyond the scope of this paper.

In [6] we augmented the level 1 i* models with NFR framework [7] claims, which are used to record assumptions about the domain, or the behaviour of the DAS itself. Claims are attached to contribution links, either making or breaking the attached link. A made contribution link (of any i* type) is lent special credence in the decision-making process, whereas a broken link's contribution is disregarded. This differs from the use of fine-grained contribution links in that a claim speaks to the importance of a contribution, whereas fine-grained contribution links speak to its magnitude. Claims differ to i* beliefs too, in that a belief is held by an actor to be true, with no presumption of truth by the analyst. Claims allow us to strike different softgoal balances for different target systems, even if softgoal contributions are unchanged.

To allow us to demonstrate our policy derivation method, we present a conceptually simple DAS first presented in [8]. The adaptive image viewer was

designed as a pedagogical example, and its sole adaptive capability is to introduce a caching component as the latency encountered in loading files increases beyond a set threshold. The system's operating environment can be easily divided into 2 domains: low (D_1) and high (D_2) latency. For each domain, a target system is modelled: S_1 and S_2 respectively. Figure 1 shows the level 1 i* models for each target system.

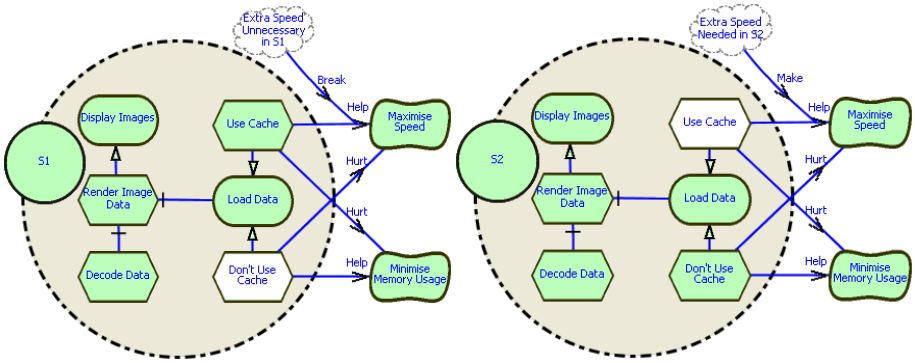

Figure 1. LoREM Level 1 models for Image Viewer S_1 and S_2 Target Systems

As Fig. 1 shows, the only difference between the two target systems is the means by which the "Load Data" goal is satisfied. With two conflicting softgoals: "Maximise Speed" and "Minimise Memory Usage", the system opts not to waste memory using a cache in the low latency domain, but tries to prioritise speed in the high latency domain. The claims on the models explain the rationale behind the selection decision, despite the contributions of the "[Don't] Use Cache" tasks remaining constant.

By comparing the two level one models in Fig. 1, it is possible to infer the component substitutions involved in transitioning from S_1 to S_2 and vice-versa. The precise component (or class) names associated with each selectable task can be placed in a lookup table when generating policies. Noting which component substitutions are necessary to transition between target systems is the first step in deriving the policies.

Figure 2 shows the level 2 i* model for the S_1-S_2 transition. A similar model is produced for each valid transition, showing the three roles of a DAS' adaptation infrastructure. The monitoring mechanism observes the environment, providing data to the decision-making mechanism, which identifies when the environment switches from one domain to another and triggers adaptation. The Adaptation mechanism performs the component substitutions needed to adopt the appropriate target system.

To derive adaptation policies, the two key elements of the level 2 model are the transition being triggered, and the trigger itself. The transition is represented by the "Adapt from S_1 to S_2" task on Fig. 2, and the trigger by the "Fire HIGH_LATENCY event" task. As with the level 1 models, a lookup table can be used to associate a specific class with the "Fire HIGH_LATENCY event" task. Identifying these two elements is the second step in deriving adaptation policies.

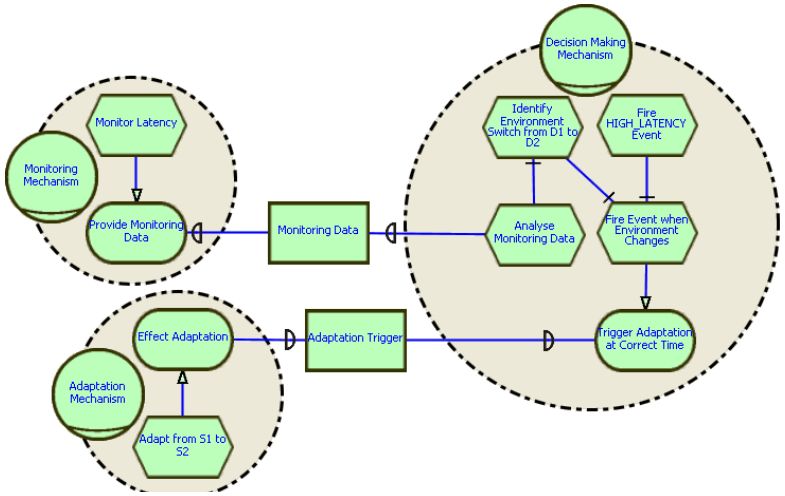

Figure 2. LoREM Level 2 model for Image Viewer S_1-S_2 Transition

From Fig. 2, we can see that the S_1-S_2 transition is triggered by the HIGH_LATENCY event, and by comparing the two level 1 models in Fig. 1, we can see that the only reconfiguration necessary is to swap the "Don't Use Cache" image loader with the "Use Cache" one. Hence, Fig. 3 shows the relevant portion of an adaptation policy compatible with the GridKit adaptive middleware [2].

```
<ReconfigurationRule>
  <FrameWork>Cache</FrameWork>
  <Events><Event><Type>HIGH_LATENCY</Type><Value/></Event></Events>
   <Reconfiguration>
      <FileType>Java</FileType><Name>Reconfigurations.Cache</Name>
   </Reconfiguration>
</ReconfigurationRule>
```

Figure 3. Snippet from Image Viewer Adaptaion Policy

The rule shown in Fig. 3 is a snippet taken from a full adaptation policy generated by the tool created to automate the process. The tool operates on LoREM level 1 and 2 models created using the Organisation Modelling Environment (OME) i* modelling tool . The tool can be adjusted to produce policies for other adaptive infrastructures, and operates on the OME tool's saved models.

4 Conclusions

DASs are a notable class of system, representing a first step on the road to fully autonomous systems. The complexity of DASs, and the environments for which they are conceived presents a problem of scale to the software engineering process. The i* based LoREM process offers a way to model and specify DASs where the

environment can be partitioned. This work builds upon LoREM, speeding the development of adaptation policies, aiding DAS implementation and maintenance.

We have demonstrated a link between a DAS' level 1 and 2 i* models and the system's adaptation polices. It is possible to derive the policies automatically, directly from models created with the OME i* modelling tool. Confirming this link not only improves support for policy creation, but allows sub-optimal adaptive behaviour to be traced back to the environmental understanding that led to its specification.

5 Ongoing and Future Work

The LoREM process is applicable only to DASs in partitionable environments, which is not always the case. We are considering ways in which other environments may be better supported by similar processes. Within the LoREM process, we are examining ways in which i* models can be used to validate developed DAS behaviour.

Perhaps the most important possibility opened up by this work is (as mentioned in section 2) is that of a DAS deriving its own adaptation policies from models at runtime. For this to be useful, the models would also need to be modifiable. By monitoring system performance and the environment it may be possible to identify modelling deficiencies automatically. In response, the models could be modified by the DAS, which would then re-derive its adaptation policies, thus adjusting its behaviour to fit the new models. Systems performing this kind of adaptation could be described as self-maintaining or self-tuning, and are the subject of ongoing work.

References

1. Welsh, K., Sawyer, P.: When to Adapt? Identification of Problem Domains for Adaptive Systems. Requirements Engineering: Foundation for Software Quality. pp. 198-203 (2008).
2. Grace, P., Coulson, G., Blair, G., Mathy, L., Yeung, W.K., Cai, W., Duce, D., Cooper, C.: GRIDKIT: Pluggable Overlay Networks for Grid Computing. 1463--1481 (2004).
3. Goldsby, H.J., Sawyer, P., Bencomo, N., Cheng, B.H.C., Hughes, D.: Goal-Based Modeling of Dynamically Adaptive System Requirements. Proceedings of the 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems. pp. 36-45 IEEE Computer Society (2008).
4. Berry, D.M., Cheng, B.H.C., Zhang, J.: The four levels of requirements engineering for and in dynamic adaptive systems. 11th International Workshop On Requirements Engineering Foundation For Software Quality (REFSQ). (2005).
5. Blair, G., Bencomo, N., France, R.B.: Models@ run.time, (2009).
6. Welsh, K., Sawyer, P.: Requirements Tracing to Support Change in Dynamically Adaptive Systems. Proceedings of the 15th International Working Conference on Requirements Engineering: Foundation for Software Quality. pp. 59-73 Springer-Verlag, Amsterdam, The Netherlands (2009).
7. Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J.: Non-functional requirements in software engineering. Springer (2000).
8. Lapouchnian, A., Liaskos, S., Mylopoulos, J., Yu, Y.: Towards requirements-driven autonomic systems design. SIGSOFT Softw. Eng. Notes. 30, 1-7 (2005).

Exploring Risk-Awareness in *i** Models

Constantinos Giannoulis, Jelena Zdravkovic,

Department of Computer and Systems Sciences
Stockholm University
Forum 100, SE-164 40 Kista, Sweden

{cgia, jelenaz}@dsv.su.se

Abstract. The i* modeling technique focuses on an early-phase of requirements engineering aiming at understanding how a system would meet organizational goals, how it would fit within the organizational context, why would it be needed and why should it be preferred among other possible alternatives. Analysts are able to understand early the organizational context that bridges system requirements to organizational goals. However, it is not clear how uncertainty, potential threats and opportunities are taken into account both when developing a strategic dependency model and a strategic rationale model, to facilitate a continuous risk management. This paper proposes a set of guidelines for refining i* models based on risk.

Keywords: i*, risk, goal, dependency, vulnerability, requirements.

1 Introduction

During an early-phase of requirements engineering activities, i* models are used to capture the intentional aspects of a system. What do stakeholders intend to do using the system, how would the system add value to the stakeholders, how would the system support the stakeholders achieve their goals. The objective is to build the context for the system by linking it to the operational and business environment, the organizational structure, to fit stakeholders' expectations and intentions to their goals.

According to [2] echoed by [1], the aim of the early-phase is understand the "whys" of the system requirements, whereas the later-phase is focused on "what" to conclude with requirements specification. Capturing the "whys" provides insights on satisfying the stakeholder's interests, their viability and uncertainty involved.

However, as acknowledged by [3], risks considered early along with stakeholders' goals, can prevent from costs arising from their late discovery (e.g. during or post development) and can contribute good criteria for the analyst to choose among different alternatives when defining requirements. According to [4], risk refers to "... exposure to a proposition of which one is uncertain.", where [4] refers to perceived risk. This definition, from one hand stresses the operational nature of risk relating it to stakeholder's intentions and perception of the organizational environment, while on the other hand, it fits any risk management approach ([5], [6], etc.).

2 Objectives of the Research

In order to capture uncertainty, threats and opportunities during the early-phase of requirements engineering activities we aim at:

- Embedding risks into the development of i* models,
- Linking the early-phase requirements engineering activities and goals set to risk management.

3 Scientific Contributions

Capturing stakeholder interests, as well as how they could be addressed relies on the interaction between analysts, stakeholders and decision makers [1] expressed by the SDM[1] and the SRM[2]. We propose a set of guidelines for refining i* models considering risk through NASA's risk matrix [6], which provides a qualitative understanding of risks, without adding complexity.

To illustrate our guidelines, each step is accompanied with an example coming from a Massively Multiplayer Online Game (MMOG) scenario. There are four actors identified, a game provider (GP), an internet service provider (ISP), a shipping agency and a customer. The GP is the principal actor, creating game content, selling and distributing the game on CDs to customers. The GP obtains the services from an ISP for selling and providing the game. The ISP receives payment as compensation for their service. The GP's game software delivery service takes place through a shipping agency. Customers access the game servers in order to play and pay the GP.

The first step focuses on populating a detailed list of dependencies considering both the SDM and the SRM, focusing on the actor of interest. In step 2 dependencies are analyzed for relevant risks to be identified. Step 3 assesses the identified risks based on their impact and likelihood, resulting into a classification through the risk matrix. Step 4 covers possible influences on the SRM and the SDM coming from different mitigation strategies.

Step 1: Identify detailed dependencies.

The SDM is built where strategic relationships are captured through dependencies. Focus is put on the actor of interest whose dependencies are listed. The analyst builds the SRM for the actor of interest enriching each dependency with intentional relationships. The impact of a dependency within an actor becomes visible, as well as the rationale of the dependency itself. The outcome is a detailed list of dependencies.

In our example, from the SDM, we list the dependencies of the GP. The GP depends on customers for the goal "Game Sales" and the task "Pay for Games", on the ISP for the resource "Hosting Service" and on the shipping agency for the resource "Shipping Service" (figure 1).

[1] The strategic dependency model (SDM) captures the dependency relationships among actors [7].

[2] The strategic rationale model (SRM) captures the rationale behind dependencies and reveals actors' intentions [7].

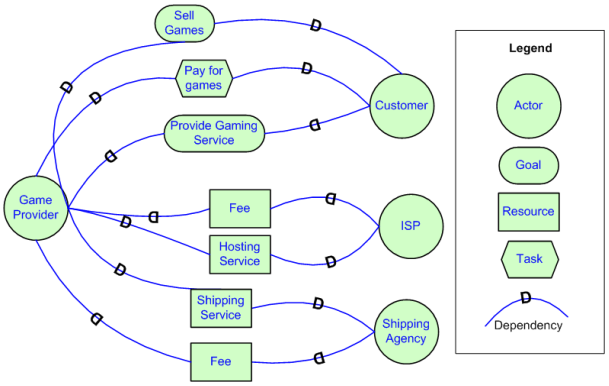

Figure 1: The SDM for the MMOG scenario

The list of dependencies of the GP is enriched with intentional relationships in the SRM (figure 2). The GP depends on:

- Customers for achieving the goal "Game Sales"; this goal can be met by coordinating game provisioning (means-end link), which in our example consists of (decomposition link) the resource Game and the tasks Deliver CDs and Sell Online Gaming,
- Customers for carrying out the task "Pay for games", as the coordinate game provisioning task requires this task be performed,
- The ISP provider for the resource "Hosting service", as the sell online gaming task makes use of it,
- The shipping agency for the resource "Shipping Service", as the Distribute CDs task makes use of it.

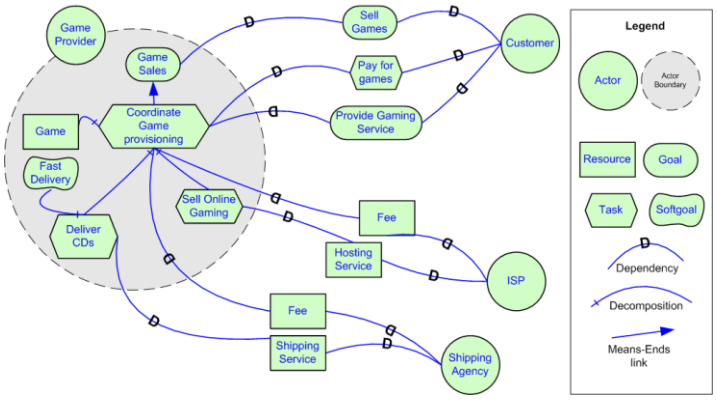

Figure 2: The SRM for the MMOG scenario

Step 2: Identify risks

For each elaborated dependency the analyst aims at discovering, interactively with stakeholders, what could go wrong and what would be the consequences if things

went wrong[3]. Each dependency should be examined to identify whether undesired events could happen, how and when, thus capturing exposure and uncertainty, ergo list risks.

In our example, the GP is running the risk of customers not buying the game (A), or not paying for the game (B), the risk of the ISP provider failing to provide adequate hosting service (C) and the risk of the Shipping agency providing bad service (D).

Step 3: Assess risks

The analyst should assess the impact and likelihood of occurrence for the risks identified. The dependency classification of i* according to vulnerability (open, committed and critical [7]) reflects on impact, whereas for likelihood, probability scales are adequate. The two scales become the two axes of the risk matrix [6] and provide a risk classification.

For our example, considering impact due to vulnerability (1-3), bad shipping service (D) belongs to open (marked with 1), inadequate hosting service (C) belongs to committed (marked with 2) and both not buying (A) and not paying (B) belong to critical (marked with 3). Regarding likelihood, we use a five score scale of occurrence with 0%-20%, 20%-40%, etc. of vulnerabilities being compromised. According to table 1, highest risk lies on (A) and (C)[4].

Table 1: The risk matrix for MMOG

Likelihood				
	5			
	4	**D**	**C**	
	3			**A**
	2			
	1			**B**
		1	2	3
		Impact		

Step 4: Mitigate Risks

The analyst revisits the SRM and considering the matrix starts addressing risks. Mitigating risks should involve stakeholders to help address what if questions relevant to the vulnerabilities. This could include modifying existing intentional relationships within actors, introducing additional to minimize likelihood or make vulnerabilities explicit for the requirement specification or even introducing new dependencies. Changes in the SRM may not necessarily reflect on the SDM (e.g. introducing soft goal decomposition). However, on the later-phase of requirement engineering activities when producing the requirement specification, such changes will appear as additional constraints.

For our example, regarding Hosting service (C), a new soft goal dependency could be introduced between the two actors for Good Online Service. This means the GP

[3] It is not within the scope of this paper to elaborate on the identification of vulnerabilities like [8] and analysis of risks, like [6].

[4] Probabilities identified rely on empirical information coming from stakeholders.

would define what is satisfactory to benefit from the ISP's capabilities. Introducing such a new dependency would result into the modification of the SDM.

An alternative mitigation strategy could be to introduce a soft goal like "Good ISP" for the Sell online gaming task through decomposition. That would serve as a quality goal for the task, and would restrict the selection among alternatives, but would not appear on the SDM as no new dependency is introduced. Ergo, appear as a constraint in the requirement specification for selecting an ISP minimizing exposure, uncertainty or both.

4 Conclusions and Future Work

The proposed guidelines lead to refined i* models, embedding risks qualitatively (exposure and uncertainty). Applied iteratively, the guidelines enhance the use of i* by allowing stakeholders to assess risks related to their goals, elaborate on available possibilities for using information systems and provide risk assessment on using the system to achieve these goals.

5 Ongoing and future work

This paper has presented our effort to embed risks into i* models and provide a link between an early-phase of requirements engineering and risk management. Future work includes examining multi-actor risks and relating i* constructs to risk classifications (e.g. associate critical dependencies to functional requirements, soft-goals to non-functional requirements, etc.).

6 References

1. Yu E.: Towards Modeling and reasoning Support for Early-Phase Requirements Engineering. In: 3rd IEEE International Symposium on Requirements Engineering (RE'97), pp. 226--235. IEEE Press, Washington D.C. (1997)
2. Yu E., Mylopoulos J.: Understanding Why in Requirements Engineering-with an Example. In: Workshop on System Requirements: Analysis, Management, and Exploitation, Schloss Dagstuhl, Saarland, Germany (1994)
3. Asnar Y., Giorgini P., Mylopoulos J.: Risk Modeling and Reasoning in Goal Models. Technical report, DIT-06-008 (2006)
4. Holton G.A.: Defining Risk. J. Financial Analysts, Vol. 60, 6, 19--25 (2004)
5. van Lamsweerde A.: Requirements Engineering: From System Goals to UML Models to Software Specifications. Wiley, West Sussex (2009)
6. NASA IV&V Management System, Guidelines for Risk Management, S3001, http://www.nasa.gov/centers/ivv/pdf/209213main_S3001.pdf
7. i* Quick Guide, http://istar.rwth-aachen.de/tiki-index.php?page=iStarQuickGuide
8. van Lamsweerde A., Letier E.: Handling Obstacles in Goal-Oriented Requirements Engineering. J. IEEE Transactions on Software Engineering (TSE), Vol 26, 10, 978—1005 (2000)

From Adaptive Systems Design to Autonomous Agent Design

Alexei Lapouchnian[1] Yves Lespérance[2]

[1] Department of Computer Science, University of Toronto, Canada, alexei@cs.toronto.edu
[2] Department of Computer Science and Engineering, York University, Toronto, ON, Canada, lesperan@cse.yorku.ca

Abstract. Interest in adaptive systems design has been steadily growing in the SE community, in part due to the ever-increasing complexity of modern software-intensive systems. Inspired by control theory, various types of controllers (e.g., feedback) are beginning to appear in software architectures for many applications. Within those controllers, distinct activities such as monitoring, analysis/diagnosis, planning, and execution are present. Approaches for requirements engineering, software architectures, and the design of such systems are beginning to emerge. In the case of agent-oriented systems, however, their hallmark is the ability to operate in highly dynamic and incompletely specified environments, handle tasks that may not be known a priori, etc. In this position paper, we look into what needs to be included in requirements-driven methods for designing agent-oriented systems, which, while drawing on ideas from control theory and adaptive systems design, would support much more autonomy, flexibility, and dynamism.

1 Introduction

As the complexity of software-intensive systems increases, more research in software engineering (SE) is being dedicated to the modeling, analysis, and implementation of adaptive systems, which promise to take on a certain range of tasks related to self-maintenance and self-adaptation. The main motivation for this is to reduce the maintenance overhead for these systems, to allow them to adapt to changing environments and user needs while continuing to deliver their functionality.

There are a number of ways to implement self-adaptation in software. A recent paper [1] identified several of the most common approaches for adaptive systems design. One of these approaches, which advocates the use of control loops, has roots in control theory, while another, which uses agents and multiagent systems, in artificial intelligence. There is a certain overlap between control loops and agents since in both paradigms, monitoring/sensing of the environment is followed by some analysis/reasoning and the enactment of the appropriate behaviour, both agents and control loops can be organized hierarchically, etc. The emerging control loop-based approaches for developing adaptive systems attempt to use well-founded systematic techniques to specify the details of self-adaptive systems. However, what distin-

guishes the agent-based approach is that it can support a higher degree of autonomy and distributed decision making, as well as be effective in highly dynamic and incompletely known environments, among other things. In this position paper, we consider several problems. Firstly, we look into which adaptation scenarios warrant the selection of agent-based approaches over control loop-based ones. Secondly, we look into what needs to be included in requirements-driven methods for designing agent-oriented systems in a more transparent and predictable way, which, while drawing on ideas from control theory and adaptive systems design, would achieve greater autonomy, flexibility, etc.

2 Background and Related Work

Control loops and especially feedback loops, have a long and successful history in engineering, and have recently been promoted as a promising way to implement self-adaptive software systems [2, 3]. Moreover, it is argued that adaptation concerns should be modeled and designed separately from the main functionality of the system (e.g., [3]), which fits nicely with the control loop approach. Feedback loops provide a generic mechanism for self-adaptation. To realize self-adaptive behaviour, systems typically have a number of controllers that can be organized into hierarchies. The main idea of feedback control is to measure the system output and achieve control objectives (e.g., maintaining a CPU utilization rate for a server) by adjusting system parameters. Thus, feedback controllers can be viewed as having to monitor the system, analyze the captured data, perform diagnosis, and plan and execute a course of action. One can then concentrate on aspects of these specific activities within the feedback controller. However, while there have been numerous advances in requirements engineering for this, little attention has so far been paid to the elicitation and analysis of the adaptation requirements such as deciding what to monitor, how to perform the analysis of the monitored data and the system diagnosis, as well as when and how to do compensation.

There are a number of recent agent-oriented approaches (e.g., [6]) that attempt to extend the Tropos requirements-driven approach [7] to support the design of adaptive systems. In [6], an approach for Belief-Desire-Intention (BDI) agent systems is proposed. Tropos extensions include the modeling of failure symptoms, possible causes, and compensations. This significantly constrains the amount of autonomy that the agents have in dealing with dynamic and incompletely known environments.

3 Research Objectives

One of the key questions for the research proposed in this position paper is to determine, given the benefits of agents and multiagent systems (MAS) as well as their costs and limitations, in which circumstances the use of agent technology is warranted in the design of adaptive systems. Also, this may help in determining the complexity and architecture of the agents needed. We propose to look at the recent research on the dimensions of self-adaptive software systems [8] and identify particular dimen-

sions, and values within those dimensions, that demand flexibility and autonomy that can be provided by the use of agents, thus indicating in what circumstances the use of an agent-based adaptive system is most promising. Here, we list some of the dimensions taken from [8] that seem like the most relevant for the use of agents:

- Goal *flexibility* is related to the level of uncertainty and flexibility in goal specifications. The values in [8] for this dimension are: rigid, constrained, and unconstrained. Common SE approaches are mostly applicable to rigidly specified goals. However, agents are capable of handling much more uncertainty in goal specifications up to the point where goals may not be known at design time. Thus, the need to handle unconstrained goals is pointing to planning agent-based solutions.
- *Anticipation* of change captures whether change (i.e., the cause of adaptation) can be predicted. The values are foreseen (taken care of), foreseeable (planned for), and unforeseen (not planned for). Clearly, the need to handle unforeseen changes may require the use of planning/reasoning agents capable of selecting and/or constructing plans (albeit in particular pre-analyzed domains) at runtime.
- *Autonomy* of adaptation mechanisms identifies the degree of outside intervention during adaptation (from autonomous to assisted – by a human or another system). While both control loop-based and agent-based approaches provide some degree of autonomy, in the context of unforeseen changes, reasoning and social agents will generally be more autonomous and thus more applicable. In MAS, the notion of a system may be quite fluid, since agents may be joining and leaving the MAS.
- *Organization* of adaptation – centralized vs. decentralized. Agents naturally support decentralized adaptation, which is especially useful when information is distributed. The challenge here is to integrate adaptations addressing different concerns that may be implemented by different agents and manage conflicts between them. Approaches to this include social laws and mechanism design. Thus, multiagent systems can provide another level of control and governance to distributed adaptive systems. Another challenge is integrating agent-based adaptation mechanisms with other approaches, notably control loop-based methods.
- *Effect* (*predictability*) of adaptation – whether the consequences of self-adaptation can be predicted both in their nature and temporal extent. Degrees of predictability range from non-deterministic to deterministic. This predictability is associated with guarantees. In control loop-based approaches, the effects of adaptations are systematically studied and generally are predictable (still, external *disturbances* can make things less predictable). In agent-based systems, especially in complex ones, predicting *emergent behaviour* can be difficult. Careful derivation of constraints on reasoning components and other advanced agent features (e.g. using "anytime" methods) may improve predictability.

Another question that we propose to address is the following. On the one hand, agents and MAS offer a lot of power and flexibility when autonomy is needed, when dealing with dynamic and incompletely known environments, goals that are unknown at design time, etc. On the other, these advanced features frequently are not supported by systematic requirements-driven engineering approaches, are hard to represent visually in modeling notations, and may not provide enough predictability and transparency for some domains. Moreover, there is a variety of agent technology/architectures, ranging from simple rule-based reactive agents to planning and decision-theoretic agents. While the former can be seen as variations of feedback

loops and thus can use, e.g., the ideas in [4], the latter seem to require quite different modeling and analysis techniques and cannot be easily dealt with by existing requirements-driven approaches such as Tropos.

Thus, the challenge is to allow the use of advanced agent techniques as needed, while improving the transparency and predictability of agent-based adaptive systems. We need more systematic requirements-driven agent-oriented software engineering approaches. Here, we can treat agents as feedback controllers with distinct monitoring, reasoning, planning, and execution activities and then use ideas from feedback control-based adaptive systems and proposals such as [4] to systematically derive not only the functional requirements for the system, but also the adaptation requirements – for monitoring, reasoning, etc. The method should support explicit representation of reasoning and (classical or decision-theoretic) planning capabilities within agents. The requirements for these agent features can be identified by looking at the relevant adaptation dimensions as described above. Constraints on the behaviour of these agent components should also be elicited and represented. Declarative specifications of at least certain parts of agents will support their evolvability and help in avoiding the need to explicitly and exhaustively capture, e.g., situations requiring adaptation. These agent features support shifting goal refinement from design time to runtime.

One possible difficulty with the above approach is that to improve the predictability and transparency of agent-based adaptive systems, the proposed method needs to separate the specification of the adaptive functionality from the main system functionality. For example, if an agent has a component capable of constructing plans to achieve goals in a certain domain as well as to adjust these plans as they are being executed, not only does the component address the functional requirements, but the adaptation requirements as well, since it has to monitor plan execution and the state of the environment, compute diagnosis, and provide compensations/do replanning. It remains to be seen whether this idea has limitations.

4 Ongoing and Future Work

Many of the ideas suggested in the previous sections are for future work. We are performing a thorough analysis of the dimensions of self-adaptive software systems and the identification of the ones that warrant the use of agent-based adaptive systems approaches. Moreover, we are interested in identifying which adaptive system requirements and which values for adaptive systems dimensions can help us with the selection of a particular agent type/architecture (e.g., simple rule-based reactive agent vs. BDI agent vs. classical planning agent vs. decision-theoretic planning agent).

In [4], an attempt is made at deriving monitoring and analysis requirements for feedback loops given a particular class of meta-requirements (awareness requirements). These meta-requirements are captured using goal models *in addition* to the usual functional and non-functional requirements for the system. These models represent the requirements of *meta-processes* responsible for the adaptive behaviour of the system. Contexts [5] are used to model situations requiring adaptation (e.g., failures), while compensation goals are explicitly represented and refined. However, this approach involves explicit modeling of situations that require changing the behaviour of

the system as well as explicit specifications of adaptations/compensations. One cannot say in this approach that decisions about when and how to change the behaviour are to be made at runtime. We plan to use the approach of [4] as a starting point to integrate ideas from control loop-based approaches with relevant agent techniques to support the analysis and design of agent-based adaptive systems.

Integrating centralized control loop approaches with the distributed agent-based approaches is also a challenge. How can we seamlessly integrate these adaptation mechanisms? Can they be used in a hierarchical fashion (e.g., low-level feedback loops being controlled by higher-level goal-driven agents) or at the same level, each controlling a particular aspect of system adaptation?

5 Conclusions

Research in self-adaptive systems is growing in importance driven by the increasing complexity of software systems. While control loop-based approaches for engineering adaptive systems look promising, they lack support for distributed adaptive behaviour that supports dynamic and incompletely known domains. On the other hand, agent-based approaches, while being powerful in their flexibility, support for dynamic goals, etc., may lack predictability and transparency. In this position paper, we argue for a requirements-driven design approach that builds on control loop-based approaches to support more flexibility, autonomy, as well as transparency and predictability, in agent-based adaptive systems.

References

1. Cheng, B.H.C., et al. Software Engineering for self-adaptive systems: a research roadmap. *Software Engineering for Self-Adaptive Systems*, LNCS Vol. 5525, 2009, pp. 1–26.
2. Hellerstein, J.L., Dao, Y., Parekh, S., Tilbury, D.M. Feedback control of computing systems. Wiley, 2004
3. Brun, Y., et al. Engineering self-adaptive systems through feedback loops. LNCS Col. 5525, 2009, pp. 48–70.
4. Lapouchnian, A., Souza, V.E.S., Mylopoulos, J. Awareness requirements for adaptive systems. Submitted for publication.
5. Lapouchnian, A. and Mylopoulos, J. Modeling domain variability in requirements engineering with contexts. In Proc. *ER '09*, LNCS Vol. 5829, 2009, pp. 115–130.
6. Morandini, M., Penserini, L., Perini, A. Towards goal-oriented development of self-adaptive systems. In Proc. *SEAMS '08*, pp. 9–16.
7. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004.
8. Andersson, J., de Lemos, R., Malek, S., Weyns, D. Modeling dimensions of self-adaptive software systems. *Software Engineering for Self-Adaptive Systems*, LNCS Vol. 5525, 2009, pp. 27–47.

Strategy Representation Using an *i**-like Notation*

Lam-Son Lê[1], Bingyu Zhang[2], and Aditya Ghose[1]

[1] School of Computer Science and Software Engineering
Faculty of Informatics
University of Wollongong
New South Wales 2522, Australia
{lle, aditya}@uow.edu.au
[2] 49 New Dapto Road
Wollongong, New South Wales 2500, Australia
zhangbingyu007@gmail.com

Abstract. Assessing and achieving alignment between an organization's strategies and its IT/business functions has long been recognized as a critically important question. This paper reports on a project that seeks to overturn established management orthodoxy by establising that strategies can be adequately modeled using conceptual modeling notations and that methodological and tool support can be provided for the task of assessing and achieving alignment between the strategies of an organization and its service offerings. A key element of this enterprise has been the design of SML - the Strategy Modeling Language. This paper presents an interim report from this project that describes how a notation inspired by i* has been used to obtain the diagrammatic modeling component of SML, and how i*-like notions have been used to represent strategy decomposition (required to be able to refine strategies to a level where there is an onotlogical match between the languages used to describe strategies and services). We also comment on how i*-like notions would play a greater role in this project, as a complete model of the enterprise context is brought to bear on the alignment exercise. We provide a brief illustration, and a description of the toolkit implemented on the Eclipse platform.

Key words: i*; strategy modeling; strategic alignment;

1 Introduction

A large body of literature, spanning several decades, has highlighted the critical importance of strategic alignment to the management and information systems communities. Yet there is also considerable pessimism about the prospects for solving this problem using computer-mediated tools and methodologies. This

* Funding of this research was provided by the Smart Services CRC Initiative http://www.smartservicescrc.com.au/

paper presents an interim report from a project on strategic service alignment, within the Australian Cooperative Research Centre for Smart Services. We have approached the problem by first devising a language for modeling organizational strategy in a manner amenable to machine processing (the Strategy Modeling Language - SML), then defining a high-level business service modeling language (the Business Service Representation Language - BSRL) [2] and finally developing a machinery that enables the assessment of alignment and supports dynamic re-alignment in the face of changing business contexts.

In this paper, we focus on the modeling of strategy. The Strategy Modeling Language (SML) has been designed to include both a textual and diagrammatic interface for modeling and visualizing the strategic landscape of an organization. We describe how the i* framework [7] provided the basis for the design of the diagrammatic strategy modeling notation. We also note that the Strategy Modeling Language augments the i* ontology in two critical ways. First, it provides for modeling strategies as plans (see examples later in this paper). This is a significant departure from i*, which is sequence-agnostic. Second, it provides for modeling strategies as optimization objectives. While i* provides for modeling softgoals, which arguably have similar intent, SML explicit strategy modeling via *objective functions*. Unlike the informal account of softgoal decomposition in i* and related work on softgoals, our approach provides for *formal decomposition* of objective functions (the full explication of this is outside the scope of this paper).

Our current work has mainly focused on alignment within a single enterprise context, but is currently being extended to address cross-enterprise value chains. The capabilities offered by i* SD diagrams to model enterprise structure via actor/dependency models is critical for modeling strategy in a cross-enterprise context, and represents an important direction for future development.

2 Background

Current work on strategic alignment looks at how strategies can be specified in relation with other artifacts such as actors, business processes, resources. The key approaches in this space (outside of i*) include e3 Forces [3] (a framework for modeling perspectives of an organization including a strategy-oriented perspective) and the InStAl method [5] (aligning the strategy and functional aspects). Other proposals include GRL[2] (supports goal-oriented reasoning), GOORE (a goal-oriented method for requirements elicitation) [4]. None of these encompass the full range of modeling constructs that SML supports.

Modeling the decomposition of business strategy plays a vital role in the landscape of enterprise strategic alignment. To be able to align business services to organization's strategy, we need to have a rich model that represents the organization's strategy to work with. i* is the main the source of ideas that particularly influences our vision on strategy modeling. Our initial idea in this

[2] GRL online `http://www.cs.toronto.edu/km/GRL/`

direction explores the notion of contextual consistency in goal decomposition [1]. In this paper, we aim to tailor the notion of goal decomposition to capture organization's strategy and define a specific notation for it.

3 Business Strategy Decomposition

In this section, we present how we extend i* to cope with strategy decomposition (Subsection 3.2), which is illustrated by an example (Subsection 3.1).

3.1 Example

Let us consider an example that describes a multi-national book-seller whose management decides to provide their services via the Internet. The management sets out the strategy for the book-seller is to become the market leader amongst book-sellers in Australia. As the business of selling books involves marketing, optimizing operating costs and dealing with book suppliers, the main strategy is then broken down into three more concrete component strategies (i) To first gain market-share in New Zealand, then use United Kingdom market credibility to enter Australia (ii) To minimize operating costs (iii) To manage supplier relationships by providing purchase volume guarantees, and fast payment against invoices. This process can be carried out until the management reaches a set of strategies that are concrete enough to map to business services or business processes[3] that operationalize them.

Figure 1 gives the decomposition hierarchy of strategy for the book-seller. Note that each strategy is prefixed by a string followed by a colon that is in turn followed by textual description of the strategy being represented. The prefix is actually a concatenation of an abbreviation and a number. The former denotes the strategy type and the later signifies the hierarchical branch at which the strategy being represented is. To reason on strategy decomposition more effectively, we differentiate three types of strategy: business plan, functional goal and optimization objective [6].

3.2 Diagrammatic Representation of Strategy in Toolkit

In our project on strategic alignment, we have been developing a toolkit called ServAlign. This tool manages a repository for strategy and a catalog of business services. In addition, the tool permits diagrammatic representation of strategy and decomposition of strategy. The diagrammatic notation of strategy modeling

[3] Business services and processes can be regarded as the main vehicle for the operationalization of an organization's business strategy in a manner akin to the way in which object-oriented components have provided the basis for implementing software requirements in traditional software engineering thinking. The topic of how to align business services and processes to organization's strategy is out of the scope of this paper.

Main strategy: (FG) To become the market leader amongst book-sellers in Australia. This strategy can be decomposed into

- P1: To first gain marketshare in NZ, then use UK market credibility to enter Australia
  - FG11: Establish market presence in NZ
  - FG12: Establish market presence in the UK by
  - FG13: Establish Australian market leadership
    - OO131: Maximize market visibility
      - OO1311: Maximize proximity of store locations to high customer traffic locations
      - OO1312: Maximize customer "mindshare" via an advertising campaign
      - FG1313: Develop a website that supports sophisticated product search, online payments and a customer feedback facility
        - FG13131: Leverage a comprehensive back-end ERP system
          - FG131311: Re-use ERP systems built for the NZ and UK markets
    - OO132: Maximize market coverage
      - OO1321: Maximize product offerings covering all age segments
      - OO1322: Maximize product offerings covering all genre segments
      - OO1323: Maximize product offerings covering all relevant language/ethnic segments
- OO2: Minimize operating costs
  - OO21: Minimize inventory
    - OO211: Minimize warehouse-to-store delivery time
      - FG2111: Build inventories at each major metropolitan centre
      - FG2112: Engage efficient trucking service provider
    - FG212: Leverage demand forecasting capabilities of back-end ERP system
      - FG2121: Re-use ERP systems built for the NZ and UK markets
  - OO22: Minimize logistics costs
    - FG221: Build inventories at each major metropolitan centre
    - FG222: Engage efficient trucking service provider
- P3: Manage supplier relationships by providing purchase volume guarantees, and fast payment against invoices
  - FG31: Provide suppliers with minimum purchase volume guarantees
  - OO32: Minimize invoice-to-payment delay

Fig. 1. Decomposition of the main strategy of the book-seller

used in ServAlign is tailored from i* (see Figure 2). We reuse the i* pictogram of hard goal for our functional goal while introducing additional pictograms for other types of strategy. This addition includes a block arrow for business plan and triangles for optimization objectives (i.e. either maximization or minimization).

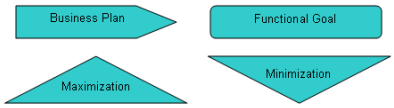

Fig. 2. i*-like diagrammatic notation used in ServAlign for modeling strategy

Figure 3 is a screenshot of our ServAlign prototype that is implemented as an Eclipse[4] plug-in. The Eclipse perspective to the left offers a tree-view that shows the entire strategy decomposition hierarchy. Each strategy in Figure 1 is now represented as a tree node. An out-zoomed notation is attached as icon to each tree node to visually illustrate the type of the strategy being represented by the tree node. The panel in the middle of the Eclipse window is dedicated to a diagrammatic interface of ServAlign. The diagram shown in Figure 3 shows the decomposition hierarchy of the book-seller's main strategy using the notation shown in Figure 2. Textual description of each strategy is printed below its pictogram. In this diagram, lines represent decomposition links.

[4] Eclipse homepage `http://www.eclipse.org/`

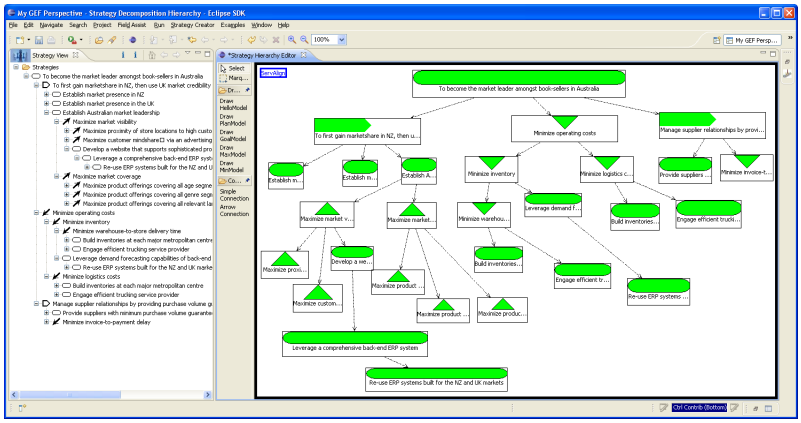

Fig. 3. Diagrammatic representation of strategy decomposition in ServAlign

4 Conclusions and future work

The novel ways in which i*-like notions can be deployed in strategic service alignment can provide useful insights to the i* community. We also expect to further leverage i* as we extend our account to cross-enterprise value chains. We are working towards (semi-)automatic, ontology-based strategy decomposition and establishment of strategic service alignment.

References

1. B. Brown and A. Ghose. Hierarchic decomposition in agent oriented conceptual modelling. In *Proceedings of 4th International Conference on Quality Software (QSIC 2004)*, pages 240–247. IEEE Computer Society, 2004.
2. L-S. Lê, A. Ghose, and E. Morrison. Definition of a description language for business service decomposition. In *Proceedings of First International Conference on Exploring Services Sciences (IESS 1.0)*, Geneva, Switzerland, 2010.
3. V. Pijpers and J. Gordijn. e3 forces: Understanding Strategies of Networked e3 value Constellations by Analyzing Environmental Forces. In *Proceedings of the 19th international conference on Advanced information systems engineering*. Springer-Verlag, 2007.
4. M. Shibaoka, H. Kaiya, and M. Saeki. Goore: Goal-oriented and ontology driven requirements elicitation method. *Advances in Conceptual Modeling - Foundations and Applications*, 4802:225–234, 2007.
5. L. H. Thevenet and C.: Salinesi. Aligning IS to organization's strategy: the INSTAL method. In *19th International Conference on Advanced Information Systems Engineering*, pages 203–217, 2007.
6. H-L. Wang and A. Ghose. On the foundations of strategic alignment. In *Proceedings of Australia and New Zealand Academy of Management Conference*, 2006.
7. E. Yu. Towards modelling and reasoning support for early-phase requirements engineering. In *Proceedings of the 3rd IEEE Int. Symp. on Requirements Engineering (RE'97)*, pages 226–235, 1997.

A Goal-Oriented Approach for Workflow Monitoring

Alicia Martinez[1,2], Nimrod Gonzalez[1], Hugo Estrada[1],
[1] CENIDET, Cuernavaca, Morelos, Mexico
[2] I.T. Zacatepec, México
{amartinez, nimrod, hestrada}@cenidet.edu.mx

Abstract. Nowadays, the use of software systems implemented for automatically guiding and monitoring business processes is becoming a common option in modern enterprises. This kind of systems (workflow management systems) is designed to monitor tasks, documents, rules, computers etc. To do this, these software systems implement an automated representation of the enterprise processes. In most cases, these systems are mainly oriented to capture the tasks performed in the business from a process-based view. However, current workflow systems have neglected the goals and objectives that enterprise wants to achieve with the performance of business tasks. Therefore, in current workflow approaches, it could be very complicated to determine if these tasks actually satisfy the business goals. In this paper, the framework Tropos is used to model the goals of the business processes and also, to establish the mechanism to control and monitor the goals of each one of the actors involved in the process. With this, it is possible to measure the impact of the fulfillment of the actor goals in the satisfaction of the enterprise goals. A software tool, called GoalFlow has been developed in order to validate the proposed methodology. This approach enables the managers to take the appropriate decisions according to the knowledge on relevant aspects in process execution: who, what, how, and why.

Keywords: goal business, workflow management systems, monitoring task.

1 Introduction

Nowadays, it is very frequent for the current enterprises to apply systems to improve their efficacy in productive and management aspects. In this context, some of the most used mechanisms for these purposes are the workflow systems, which are very useful for the enterprises to perform their functions. Systems of this kind arose to improve the efficiency by putting together persons, process and machines [1]. The objective of workflow systems is to automatically monitor and control the components of business process by using a software system installed in the enterprise´s network that implements an automated representation of the business processes. Tools and languages such as BPEL, XPDL, YAWL, openEDMS, OnBase, LiquiOffice, Cardiff o Bonita are the most well-known approaches for workflow modeling. All these tools offer a strong base to process model and these are becoming more popular every day. However, despite the advantages of the workflow systems, there are certain issues that need to be solved to allow these systems to make deeper

analysis of key aspects of the enterprise structure and the organizational behaviors, such as goals, dependencies, tasks, etc. Almost all current workflow approaches have modeling limitations, primarily because they are mainly based on representing the business tasks and they have neglected the use of goals and objectives to have a more complete view of the business processes. Therefore, in some cases, current task-based approaches cannot be enough to model a specific context.

In this paper, a goal-based approach is proposed to monitor and control the processes defined in a workflow system. To do this, the Tropos Framework [2] is used to determine the impact of the fulfillment of the actors' goals in the satisfaction of the enterprise goals. It is important to point out that *i** Framework could be also used in this context. A software tool, called GoalFlow has been developed in order to validate the proposed methodology.

The paper is structured as follows: objectives of our research are described in the next section. Scientific contributions are presented in section 3. Conclusions are reported in section 4. Finally, in section 5, ongoing and future works are sketched.

2 Objectives of the research

In this paper, a goal-based approach is proposed to improve workflow systems with the objective of exploiting the strengths of goal approaches for enabling the analysts to detail, trace and monitor business processes. Therefore, our objective is to monitor the performance of the business tasks based on the fulfillment of actor and processes goals. The idea of this approach is to integrate the specific goals of the business actors in the fulfillment of the enterprise goals. It is important to point out that current approaches to goals-processes alignment (such as the works of Nurcan [3], Bider [4], Soffer [5], etc) offer well-founded methods to manage goals in software lifecycle. Therefore, all goal-based techniques of these approaches could be applied in the method presented in this paper.

The proposed approach is composed by two main processes. The first process consists of defining the method to model business process from a goal perspective. Second process consists of defining the method to control and monitor business process. To do this, a set of metrics and axioms to measure process performance have been defined. Figure 1 shows the proposed approach to monitor processes based on business goals, where the fulfillment of actor goals contributes to completion of the enterprise processes.

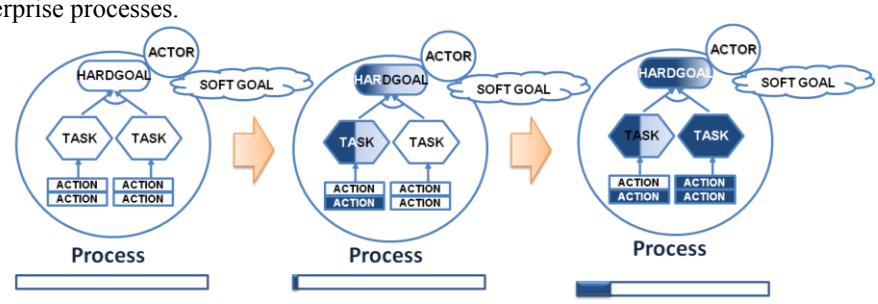

Fig. 1. The proposed monitoring method

The powerful characteristics of goal modeling are used to determine how, why, and when the actors develop tasks and processes in the business. To validate the proposed approach a software tool has been created to control and monitor business processes based on relevant aspects for their execution. It is important to point out that in our approach the goals are the mechanism used to ensure the logic connection among these aspects.

The use of goals in monitoring and controlling activities will allow to incentive the work of the stakeholders by making explicit the impact of the satisfaction of the individual goals of each business actor in the fulfillment of the business goals. In this context, all participants can be motivated by the explicit importance of their task in the business and also the participants could determine their role in the network of collaborations of each process.

According to this objective, a goal-based framework was needed to capture all relevant aspects to be modeled (who, where, how and why). We consider Tropos to be the appropriated framework for this purpose. Following, the advantages of using Tropos in this project are detailed:

- Tropos dependencies enable the analyst to determine the impact of relationships in the executions of the tasks involved in the processes.
- By following the trace of the main goals of the actors it is possible to monitor the state of the fulfillment of the milestones of the business processes.
- By refining each goal (until the level of task is reached) allows the analyst to monitor the performance of each actor and also to model the impact of this performance in the global processes.

The Tropos concepts used in this work are the following: actor, believe, capability, resource, hardgoal, softgoal dependency and task. One of the relevant concepts in this proposal is the Tropos plan because it permits to associate Tropos and Workflow concepts. We have added the concept of action to detail the Tropos plans.

Our Tropos-based method to model business processes is composed of seven steps, where the input of this method is the information of the business process to be modeled, and the output is a goal-based business process model. The steps of the method are the following: a) obtain, explore, analyze all possible sources that describe the process activities and is environment, b) identify the objectives of the process and their features, c) identify the actors that are responsible for executing some activities in the business processes, d) identify the capabilities and beliefs of each actor, e) identify the activities that are needed to carry out a process. These activities are structured in hardgoals and softgoals. Later on, these goals are associated to the actors responsible for its execution, f) refine each hardgoal identified in the previous stage. The refinement is carried out by specifying the set of the tasks needed for the execution of a goal. Later on, the required actions to execute each task are identified. The resources used in each task and action are also identified in this step, and finally, g) define the dependencies among the actors.

In this proposal, we have use metrics and axioms to implement the process to control and monitor the business processes defined in previous steps. The defined metrics allow us to quantify the progress of each one of the goals, besides quantifying the progress of each one of the modeling processes. We used the GQM approach (Goal-Question-Metric) of Basili and Rombach [6] in order to specify measurement objects. This is the first stage of the process to define metrics for business processes.

The GQM approach helps to derive meaningful questions to characterize the goals in a quantifiable way. Table 1 shows this metrics defined for our approach and its definition.

Table 1. Metrics for the goal-oriented business process.

Metric	Description	Axioms
Size of the process	Total size of the process measured according the number of actions that compose it.	An action *a* represent 100/*z* % of a process *P* and *z* is the number of actions in the process *P*.

Process performance	Progress in the execution of the goals and the tasks as the relation between the number of actions that have been executed and the total number of actions of a process.	Given a task t composed by a set *A* of *x* actions and an action *a* represent 1/*x* of a task *t*, the fulfillment of the action *a* increase the progress of a task *t*. A task *t* is finished when the actions *x* that composed it are also finished.
		If (*A* == finished) then task T = finished
		Given a hardgoal *m* composed by a set *B* of *w* actions in *n* tasks, and an action *a* represent 1/*w* of a hardgoal *m*, therefore, the fulfillment of an action *a* increase the progress of a hardgoal *m*. A hardgoal *m* is finished when the set *B* of actions that composed it are also finished.
		If (*B* == finished) then task T = finished

Efficacy in process completion	The advance in process execution as the relation between the number of actions involved in hardgoals and the process size.	Efficacy in process completion is (100/*z*) **c* % of a process *P* given *z* is the number of actions in the process *P* and *c* is the number of actions involved in hardgoals.

3 Scientific contributions

The main two contributions of this research work are pointed out: a) using a Tropos-based approach to control, trace and monitor the enterprise goals. It is important to point out that Tropos framework has been extended in order to consider monitoring activities over activities, tasks and goals. b) to make explicit how the actor tasks have impact in the satisfaction of the business goals.

The GoalFlow Tool has been developed in order to validate the proposed methodology and is composed by three main stages: actor modeling, goal modeling and finally, project monitoring and project management:

Actor modeling: In this stage the analyst must define the actor model (using Tropos) for each one of the workflows that is needed to analyze. The actor model has been adapted to indicate the start and end of the goals besides the dependencies. The softgoals are modeled in this stage to demonstrate the rate of contribution of the individual goals in the general objectives of the enterprise.

Goal modeling: The objective of this stage is to develop the goal model focus on specifying how the actor goals will contribute in the business goals. Once the actor and goal models have been created, a modeling stage is performed with the objective of creating a data tree that reflects the flows of works where the actors must collaborate regarding their assigned goals. In this stage, the analysts have a graphical

view of the organizational processes focus on representing how the goals are satisfied by plans that use resources to operate.

Project control and monitor: The objective of this stage is to determine the state of the business tasks and the progress in the actor goals. In this stage the managers have a complete view of the execution of the project regarding the fulfillment of individual goals and process goals. Several aspects related to actor performance can be visualized, also it is possible to visualize the way the actors interact among them to satisfy the business goals. This view of the complete state of the processes enables the project manager to take decisions about process performance.

4 Conclusions

In this paper, a goal-based approach is presented that use the Tropos Framework as a mechanism to monitor and control the Workflows of an enterprise. The use of this Tropos-based approach enables the analyst to establish the collaboration, cooperation and coordination at the business tasks (through the concept of dependency), and also enables the manager to determine the impact of these tasks in the fulfillment of the enterprise goals. To validate this approach, a software system (GoalFlow) was developed to apply the proposed approach in real uses cases.

5 Ongoing and future work

At present, we are working in developing a software system to manage the knowledge generated by the business actors. The system will also store the skills of each actor and the real participation and performance of each actor in the projects,

Acknowledgment: This research has been partially supported by DGEST by the project #24.25.09-P/2009.

References

1. BonitaSoft: Bonita Workflow, Available: http://www.bonitasoft.com, site visited 12-02-2010.
2. P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, and A. Perini: Tropos: An agent-oriented software development methodology. J. Autonomous Agents and Multi-Agent Systems 8(3):203-236, May 2004.
3. S. Nurcan, A. Etien, Anne, R. Kaabi, I. Zoukar, C. Rolland, Colette: A strategy driven business process modelling approach. Business Process Management Journal, Vol.11 (6), 2005.
4. B. Andersson, I.Bider, P. Johannesson, E. Perjons: Towards a Formal Definition of Goal-Oriented Business Process Patterns, Business Process Management Journal, Vol.11 (6), 2005.
5. P. Soffer, Y. Wand, Yair: On the notion of soft-goals in business process modeling. Business Process Management Journal, Vol.11 (6), 2005.
6. Basili V., Caldiera, G. y Rombach H.: The Goal Question Metric Approach, in: Marciniak, J. J. (ed.), Encyclopedia of Software Engineering, Wiley, pp. 528–532, 200.