An Empirical Study of Requirements Model Understanding: *Use Case* vs. *Tropos* Models

Irit Hadar
MIS, University of Haifa
Carmel Mountain, Haifa
31905, Israel
hadari@mis.haifa.ac.il

Tsvi Kuflik
MIS, University of Haifa
Carmel Mountain, Haifa
31905, Israel
tsvikak@mis.haifa.ac.il

Anna Perini
FBK - IRST CIT
Via Sommarive, 18
38123 Povo - Trento, Italy
perini@fbk.eu

Iris Reinhartz-Berger
MIS, University of Haifa
Carmel Mountain, Haifa
31905, Israel
iris@mis.haifa.ac.il

Filippo Ricca
DISI, University of Genova
Viale Dodecaneso, 35I
16146 Genova, Italy
filippo.ricca@disi.unige.it

Angelo Susi
FBK - IRST CIT
Via Sommarive, 18
38123 Povo - Trento, Italy
susi@fbk.eu

ABSTRACT

Visual modelling languages are commonly used to support software requirements analysis and documentation. A variety of languages are available, based on different conceptual paradigms. They can be roughly divided into two main groups: goal-oriented approaches and scenario-based approaches. In the last ten years, numerous works developed case studies that illustrate the effectiveness and limitations of goal-oriented and scenario-based approaches. A few works even suggest coupling these approaches in order to capture requirements from different perspectives. However, experimental comparisons of these approaches have been rarely addressed. This paper presents the design and preliminary results of an empirical study that compares two state of the art requirements modelling methods: Use Cases, which is a scenario-based approach, and Tropos, which is a goal-oriented approach. The objective is to evaluate different levels of comprehension of requirements models expressed in both methods, as well as to estimate the time required to perform simple analysis tasks using both methods. Preliminary results show that Tropos models seem to be more comprehensible, although more time consuming, than Use Case models to novice requirements analysts.

Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements/Specifications

General Terms

Requirements, Experimentation, Measurement

Keywords

Requirements Engineering, Controlled experiment, Tropos, UML Use Cases

SAC'10 March 22-26, 2010, Sierre, Switzerland.

1. INTRODUCTION

Conceptual modelling is considered a core activity along the software development process. It acts as a starting point for understanding the application domain and the requirements of the software system to be realized thus providing a common basis for designers, developers and users, improving understanding and communication. Different requirements modelling methods are available. They are commonly divided into goal-oriented and scenario-based [11, 15].

The basic building blocks of goal-oriented approaches [6, 16, 18] are goals, which capture the various objectives of the system to be developed at different levels of abstraction. Goal-oriented approaches focus on why systems are constructed, expressing the rationale and justification for the proposed system. They aim at (1) understanding the current organizational situation, (2) understanding the need for change, (3) providing the deliberation context within which the requirements engineering process occurs, (4) relating business goals to functional and non-functional system components, and (5) validating system specification against stakeholders' goals. Within the category of goal-oriented approach, agent-oriented methods [4, 10, 18] received a special attention. The agent-oriented paradigm rests on the idea that a system can be perceived as a group of interacting agents, each of which can be thought of as an autonomous and social entity that can communicate, coordinate, and cooperate with other agents to achieve individual and organizational goals.

Scenario-based approaches, on the other hand, promote the notion of a scenario, which is a sequence of interaction events between a system and its environment in the restricted context of achieving some implicit purposes [14]. Scenario-based approaches have been proposed for requirements elicitation and validation purposes. Jacobson's Use Case technique, which has been incorporated into UML, is a widely used scenario-based requirement engineering method [1].

In the literature, the effectiveness of goal-oriented and scenario-based approaches is analyzed in several works illustrating the application of different methods to case studies (e.g., [2, 3, 7]) or comparing the strengths and limitations of the approaches according to different criteria (e.g., [5,

13]). However, to the best of our knowledge, experimental comparisons of these requirements modelling paradigms using different visualization methods are rare. Such comparisons may raise insights and help decide which modelling paradigm to adopt for a given software development project. One important factor for comparison or evaluation is the immediacy in understanding the respective models by project's stakeholders, for instance by requirements analysts, who have to understand a given model during analysis and refinement tasks to accommodate new or changed requirements[1].

In our research we focus on how requirements analysts understand requirements models in two methods, namely Use Cases [1] (UC) and Tropos [2]. These methods can be respectively considered as representatives of the scenario-based and goal-oriented (or more accurately agent-oriented) paradigms. In the study, we measure comprehensibility of requirements models in terms of performance and time to complete tasks. Three categories of tasks can be distinguished: (1) determine consistency between the system story and the requirements visual models. This is important for checking completeness and correctness of requirements models; (2) understand the models (irrespectively of the system story) for different analysis and development activities; (3) cope with tasks that require changes and modifications of existing requirements models. Tasks in the third category usually require a higher level of model comprehension than tasks in the two first categories. The comprehension in the different categories can be measured by evaluating the performance of the requirements analysts in different (representative) tasks and the time it takes to perform them.

The objective of this paper is to describe the design of an experiment for comparing Use Case and Tropos models comprehension and to discuss preliminary results of its execution with a group of 19 bachelor students playing the role of requirements analysts. The experiment design follows the guidelines by Wohlin et al. [17] on how to document and report empirical studies in software engineering and exploits interesting solutions proposed in previous experiments [8, 12] about how to define and measure the comprehension level of a model.

The paper is structured as follows. Section 2 gives some details of the two compared modelling methods. In Section 3 the experiment, in terms of the precise research questions, hypotheses, subjects, design, material and procedure, is described. Section 4 presents and discusses the preliminary results. Section 5 reports on related work. Finally, Section 6 concludes the paper and gives some future directions.

2. REQUIREMENTS MODELS

Conceptual modelling in requirements engineering aims at understanding the application domain and at supporting the identification and specification of the requirements of the software system to be realized. Different modelling paradigms offer their own specific set of concepts to represent properties of domains and systems under analysis.

Use Case diagrams [1] offer constructs to describe the interactions between users and other systems, and the system-to-be-developed. They are graphs collecting the use cases, the actors taking place in the use cases, and the relationships among them. In particular, they visualize the relationships between different actors (specialization), actors and use cases (participation), and compound and elementary use cases (e.g. include relationships). In conjunction with the visual diagrams a description of the behaviors captured by a use case can be specified textually using *Use Case templates* containing information such as the pre-conditions or post-conditions for the described scenarios, and possible alternative scenarios.

Figure 1 depicts the syntax of Use Case diagrams and a commonly used template, which was used in our experiment.

For the goal-/agent-oriented paradigm, we considered the *Tropos* method [2]. This method rests on a set of intentional notions, such as actors, goals, plans, resources, and dependencies. An actor represents an entity that has strategic goals and intentionality within the system or the organizational setting. The actor is used to model both human stakeholders (single persons and organizations) and artificial agents (software and hardware systems and components). Goals represent states of affairs an actor wants to achieve. Executing a plan can be a means to realize a goal. Actors may depend on other actors to attain some goals or resources or for having plans executed.

Tropos models are visualized through actor and goal diagrams, the syntax of which is depicted in Figure 1. An actor diagram is a graph whose nodes represent actors and its labeled arcs - strategic dependencies between pairs of actors. A goal diagram represents the individual actor perspective in terms of its main goals, and their decomposition into sub-goals. Moreover, plans and resources that provide means for goal achievement are depicted through means-end relationships.

Both modelling methods offer a richer set of concepts and analysis techniques with respect to those recalled above, e.g. early vs. late requirements modelling in Tropos [2]. However, for the purpose of our empirical study, we consider subsets of the two modelling methods that can be comparable in terms of expressiveness, thus concentrating on comprehension of late requirements models that focus on new systems to be realized. For example, we use "OR decomposition" in *Tropos* and the "Alternative Flow" field in Use Case templates to specify alternative behaviours of the system-to-be, but avoid using UML activity diagram.

3. THE EXPERIMENT

The goal of the study is to consider requirements modelling methods based on different conceptual paradigms, namely the scenario-based UC and goal-oriented Tropos methods, with the purpose of evaluating their effectiveness when used to analyze and update the requirements models of distributed software systems.

3.1 Hypotheses, Variables and Measure

In the pilot study we focused on the following research questions and the corresponding set of *null* and *alternative* hypotheses.

- **RQ1** Which model, between Tropos Late Requirements, (i.e., Goals and Actors diagrams, *Tropos* for brevity) and UC (i.e., diagrams and templates), is more comprehensible in terms of consistency, understandability, and modifiability?

[1]Of course other elements usually contribute to this type of decision, such as the availability of a clear documentation of the method, as well as of tutorials and supporting tools.

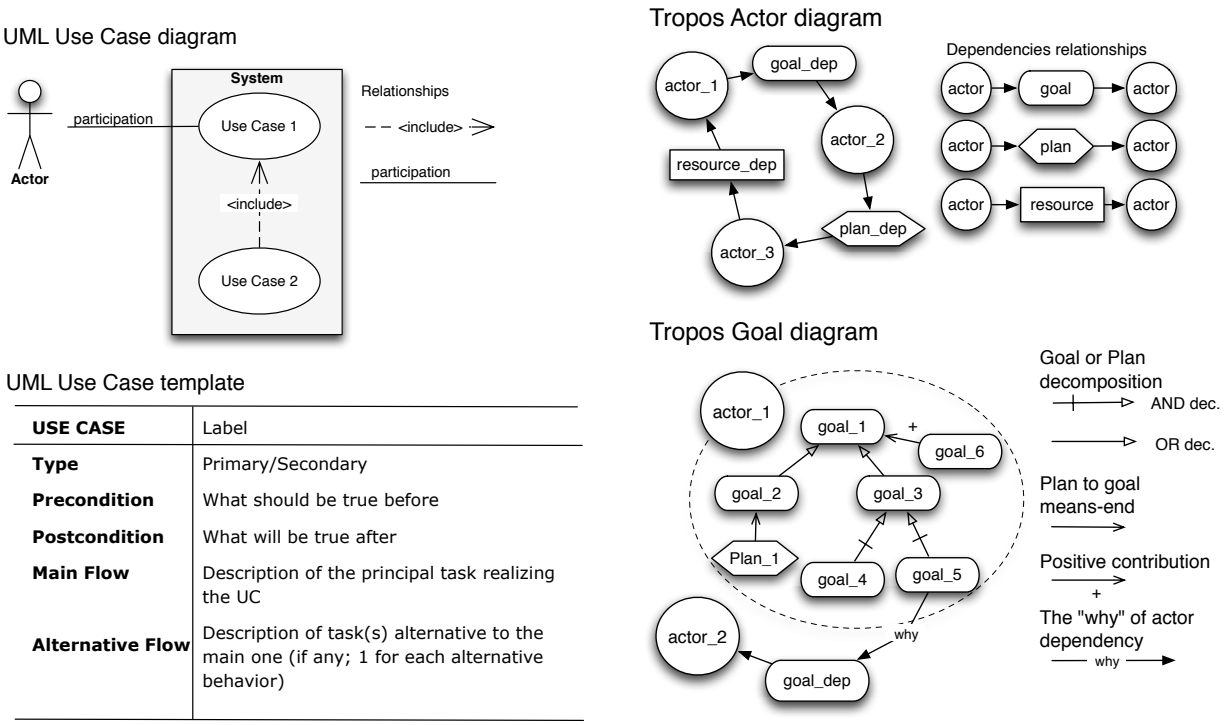

Figure 1: Diagrams and visual notations of Use Case (left) and Tropos (right) models.

- **RQ2** Which model requires less time to be evaluated for consistency, understandability, and updating?

These research questions have been translated to the following hypothesis:

- $H_0 1$ There is no difference in terms of comprehension level between *UC* and *Tropos* requirements models;
- $H_0 2$ The amount of time needed by analysts for evaluating consistency, understanding and updating *Tropos* and *UC* requirements models is the same.

When the null hypothesis can be rejected with relatively high confidence, it is possible to formulate an alternative hypothesis:

- $H_a 1$ There is a difference in the comprehension level of *UC* vs. *Tropos* requirements models;
- $H_a 2$ Evaluating consistency, understanding and updating *Tropos* requirements models need a different amount of time than evaluating consistency, understanding and updating *UC* models.

The independent variable of this experiment is the visual modelling method which can assume one of the values in $\{Tropos, UC\}$.

The dependent variables are the *comprehension level* of models and the *time* (in minutes) it took the subjects to perform different comprehension tasks. More specifically, to measure the comprehension level, we asked the subjects to answer a questionnaire and assessed the correctness of the answers (as in [12]). An example of question aiming at evaluating the understandability of a model is the following:

> "According to the model, who are the actors that provide the data about the available ingredients that the e-Restaurant uses to support the Chef? *[A]-Specify Actor(s) name(s).....*"

Since the answer to each question is expressed as a list of elements (e.g., actors, use cases and goals) we can count: (1) the set of elements mentioned in the answer to a question i by a subject s and (2) the number of elements in the answer that were expected as the correct answer to question i. Based on the above definition, we computed *precision* and *recall* for each answer. Precision measures the fraction of items in the answer that are correct, while Recall measures the fraction of expected items that are in the answer. Since the two above metrics measure two different concepts we added a derived measure, $F{-}measure$, which is a standard metrics defined as the harmonic mean of *precision* and *recall*:

$$F{-}measure_{s,i} = \frac{2 \cdot precision_{s,i} \cdot recall_{s,i}}{precision_{s,i} + recall_{s,i}}$$

To measure the comprehension levels of the models we used the mean of the F-measure over all the questions in a particular category.

The time to answer to the questionnaire was recorded directly by subjects noting down their start and stop time.

3.2 Experiment Design

The experiment population included 19 Management Information Systems (MIS) bachelor students in the last semester of their studies at the University of Haifa, Israel. The experiment took place in the elective course "Human Aspects of Software Engineering". The students in the course had already studied in previous years all the basic IS courses including "IS Analysis". In general, the students

Table 1: Experimental design.

Group	Lab 1	Lab 2
Group 1	mss-UC	eRest-Tropos
Group 2	mss-Tropos	eRest-UC
Group 3	eRest-Tropos	mss-UC
Group 4	eRest-UC	mss-Tropos

completed all core mandatory MIS courses, hence they can be considered having the same theoretical level of knowledge of requirements analysis. The experiment was conducted in two lab sessions, lasting 1 hour and a half each with half an hour break between the two sessions. Two different (but similar in complexity) software systems were considered, in application domains the subjects are familiar with. The first is a meeting scheduler system (called "mss" in short) that assists the user in planning and organizing meetings, the second is a system that automates several tasks in a restaurant (eRestaurant system, "eRest" in short), such as automatic ordering of dishes from customers or ingredients from the chef.

To avoid learning effects, the experiment design ensures that each subject worked on different application domains in the two labs, receiving each time a different treatment. Also, the design permits us to consider different combinations of application domains and treatment in different order across labs. Table 1 summarizes the experimental design. The subjects were split randomly into the four groups since they have similar level of knowledge and experience in requirements modelling.

3.3 Experiment material and procedure

Prior to the lab sessions, two lessons of 90 minutes reminded notions of software requirements and conceptual modeling in scenario-based approaches (Use Case modeling) and introduced conceptual modeling in goal-oriented approaches (with Tropos). Practical examples about how to build models and use them for analyzing software requirements were also provided using application domain different from those used for the experiment objects, but of similar complexity. Moreover, notions about the purpose of empirical studies in software engineering have been given.

For each of the two labs the experimental package that has been distributed included: a story describing the intended use of the software system, the Tropos/UC models of the software requirements, a questionnaire about model comprehension, and the slides of the lessons (they can be consulted for answering the questionnaire).

The *questionnaire* about the models is composed, similarly to [8], of 14 open questions on the assigned system. Most of the questions are realistic scenarios in requirements engineering tasks. Questions span from assessing the consistency of the visual model with the textual description of the system story, to understanding of the models for gathering typical information (such as which are the actors in the domain, what are their interactions, objectives and tasks, and so on), to some possible modifications to the models to capture changes or revisions of the knowledge about the domain and of the system requirements.

The procedure we followed for each lab session of the experiment has been: 1. we introduced the experiment describing to the subjects the material that will be given to them and the fact that they will have to play the role of requirements analysts; 2. we distributed the experimental package to the subjects; 3. we requested the subjects to specify surname, name, start-time in the questionnaire, before starting working with the questionnaire; 4. at the end we invited the subjects to mark the end-time of the task and collected all the questionnaires.

4. PRELIMINARY RESULTS

This section reports preliminary analyses aimed at testing the hypotheses formulated in Section 3.1 and threats to validity. Detailed analysis of confounding factors and survey questionnaires, as well as other analyses (e.g., paired analysis per subject), will be reported in future works. Results of statistical tests are intended to be significant for a significance level of 95%.

4.1 Data Analysis

We tested the presence of a significant difference between the two different treatments (*Tropos* and *UC*), using for both hypotheses ($H_0 1$ and $H_0 2$) the two-tailed Wilcoxon test. We selected a non-parametric test because it is very robust and sensitive when a small number of data points, as in our case (number of subjects = 19), is considered. Moreover we used a two-tailed statistical test due to the non-directionality of the hypotheses.

The left diagram in Figure 2 depicts boxplots of the F-measure for *Tropos* and *UC* models. The plot shows that subjects using *Tropos* models answered better to the questionnaire given that the median values of F-measure are higher for *Tropos* than for *UC*. This difference is significant as the Wilcoxon test results in a p-value = 0.04[2]. Thus, the first null hypothesis can be rejected (see also the "Overall" raw in table 2).

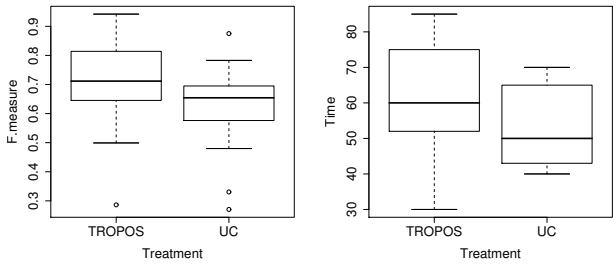

Figure 2: Boxplots of F-measure (left) and boxplots of time needed to perform the tasks (right).

We took a closer look at the results, separating the three different comprehension levels (the first 3 rows in table 2). For consistency, *Tropos* outperforms *UC* while for understandability and modifiability the differences are not significant. Thus, we decided to take a closer look at the two different cases separately (results presented in table 3). We can see that indeed usually *Tropos* outperforms *UC*, but for the individual systems the differences are not significant. Turning to the comprehension levels we see that with the "mss"

[2]Measuring separately the statistical significance of precision and recall we obtained a p-value = 0.23 for precision and a p-value = 0.017 for recall.

Table 2: Experiment results.

	F-Tropos	F-UC	p-value
Consistency	**0.72**	**0.58**	**<0.01**
Understandability	0.79	0.70	0.13
Modifiability	0.56	0.57	0.90
Overall	**0.71**	**0.62**	**0.04**

case *Tropos* outperforms *UC* for the first two levels while with the "eRest" case the differences are not significant. For the modifiability level, the results are mixed: in one case *Tropos* seems better and in the other *UC*, but in both cases the differences are not significant. Still in general, *Tropos* seems indeed to outperform *UC*, but more in-depth analysis is required for better understanding the results before carrying out a large scale experiment. Note that the small number of participants may also be a reason for insignificant results in this case.

However, what seems to be a better performance of *Tropos* comes with a cost. The right diagram in Figure 2 reports boxplots of the time needed to answer the questionnaire in both methods. It is clear that the time to answer the questionnaire is larger with *Tropos* than with *UC*. Considering the medians the difference in time between the two methods is 10 minutes. However, the difference is not significant as the Wilcoxon test results in a p-value = 0.07. Thus, the second null hypothesis can not be rejected.

4.2 Threats to validity

As usual in controlled experiments, we identified the main threats to the validity [17] of our results: construct, internal, conclusion, and external validity threats.

Construct validity threats concern the relationship between theory and observation. They are mainly due to the method used to assess the outcomes of tasks. The measurements we conceived — comprehension questions in different categories — are as objective as possible.

Internal validity threats concern external factors that may affect a dependent variable. We used two systems similar in complexity and domains well-known to all subjects. Questions were carefully chosen and models double-checked by two of the authors. Moreover the design is a one-factor balanced experiment design with random assignments that balances individual factors and learning effects. Another internal threat to the validity may be the background of the subjects. Even though the subjects are supposed to have the same (theoretical) level of knowledge, there may be differences in their performance due to work experience. However, the impact of these differences could not be assessed and since the experiment within subjects, this should not have impacted the results, but may need to be considered in the future.

Conclusion validity concerns the relationship between the treatment and the outcome. The statistical analysis is performed mainly using non-parametric tests that do not assume data normality.

External validity concerns the generalization of the findings. The main threat in this area stems from the type of subjects. They are bachelor students with not much (but same level) experience in requirements. Only further studies may confirm whether the results obtained can be generalized to more experienced subjects (e.g., professionals).

5. RELATED WORK

The benefits and limitations of both goal-oriented and scenario-based requirements engineering approaches have been discussed in different works (e.g., [15]). The main strengths of goal-oriented approaches mentioned in the literature are: (1) they make the requirements engineers focus on the problem domain and the needs of the stakeholders, (2) they cover both functional and non-functional requirements, and (3) they address various kinds of conflicts, as well as correctness and completeness concerns. However, goals are sometimes hard to elicit, since stakeholders may have difficulties expressing them in an abstract form. Scenario-based approaches, on the other hand, are rich, informal, and used in other fields as well. Nevertheless, several problems with scenarios in the context of requirements engineering have been identified, most notably: (1) they are inherently partial, raising a coverage problem, (2) they may raise a combinatorial explosion, as scenarios are specified at the instance level, and (3) scenarios leave required properties about the intended system implicit and neglect non-functional requirements.

Evaluation of approaches in the two modelling paradigms has been made using different comparison criteria. In [9], a comparative study has been made, in order to help practitioners choose an appropriate technique for their project. The main conclusion in this paper is that the most appropriate requirements engineering paradigm would be a "golden combination" of both goal-oriented and scenario-based approaches, since they are complementary to each other. [13] also goes in this direction, suggesting definition of bi-directional coupling between scenarios and goals. Using an exemplar from the health care domain, [5] compares an agent-/goal-oriented approach, i*/Tropos, and an object-oriented (scenario-based) method, UML/RUP. The conclusions here are that object-orientation is not sufficiently effective for representing and analyzing complex situations, whereas agent-/goal-orientation suffers from a number of pragmatic concerns, such as learning curve and availability of efficient tools.

Our work aims at empirically justifying the aforementioned benefits and limitations of both requirements modelling paradigms, using representative methods from both categories. From the first pilot experiment, reported in this paper, we can already see that the general comprehension of Tropos models is better than that of Use Case models. In particular, the consistency between textual descriptions of the intended system and requirements models, which is in the core of different requirements engineering tasks, is significantly better with Tropos.

6. CONCLUSION AND FUTURE WORK

In this paper we presented the design and preliminary results of an experiment to compare two state-of-the-art visual methods for requirements modelling of software systems: Use Cases and Tropos Late Requirements. We adopted a one-factor balanced experiment design with two treatments.

We focused on two variables: the *comprehensibility level* of the requirements models in the two methods and the *time* required for consistency evaluation, understanding and updating. The experiment was conducted with 19 bachelor students acting as requirements analysts who worked on the models related to two different exemplar software systems.

Table 3: Detailed experiment results separated according to systems.

	eRest			mss		
	F-Tropos	F-UC	p-value	F-Tropos	F-UC	p-value
Consistency	0.71	0.57	0.15	**0.73**	**0.59**	**<0.01**
Understandability	0.84	0.86	0.49	**0.75**	**0.52**	**<0.01**
Modifiability	0.73	0.58	0.22	0.41	0.55	0.20
Overall	0.76	0.67	0.07	0.66	0.57	0.08

Results show that Tropos seems more effective than Use Cases in terms of comprehensibility. This result is further strengthen by the fact that the subjects had been familiar with UML in general and Use Cases in particular prior to the experiment, as opposed to Tropos which they learned towards the experiment only. However, a closer look reveals some interesting issues regarding possible differences between the different levels of comprehensibility as well as the two cases that resulted in different scores. This may also be attributed to the small number of participants (when looking at the two cases individually). In any case, what seems as improved performance comes with a cost: participants spent more time performing the required tasks using Tropos. This may be attributed to the fact that they had considerably less experience in using Tropos compared to UC.

As future works we plan to complete the analysis of the collected data considering also the confounding factors and to extend the experiment to a larger number of subjects. Moreover, it would be interesting to further refine the design of the experiment in order to explore in detail the comprehensibility when the subjects have to perform different tasks of analysis and construction of the requirements models.

7. REFERENCES

[1] G. Booch, I. Jacobson, and J. Rumbaugh. *The Unified Modeling Language User Guide (Second Edition)*. Addison-Wesley, 2005.

[2] P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, and A. Perini. Tropos: An Agent-Oriented Software Development Methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, July 2004.

[3] J. Castro, M. Kolp, and J. Mylopoulos. Towards requirements-driven information systems engineering: the tropos project. *Information Systems*, 27(6):365–389, 2002.

[4] A. Dardenne, A. van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Science of Computer Programming*, 20(1–2):3–50, 1993.

[5] L. M. C. Filho, V. Werneck, J. Amaral, and E. S. K. Yu. Agent/goal orientation vs object orientation for requirements engineering: A practical evaluation using an exemplar. In *WER'2005*, pages 123–134, 2005.

[6] E. Kavakli. Goal-oriented requirements engineering: A unifying framework. *Requirements Engineering journal*, 6(4):237–251, 2002.

[7] M. Kim, S. Park, V. Sugumaran, and H. Yang. Managing requirements conflicts in software product lines: A goal and scenario based approach. *Data & Knowledge Engineering*, 61(3):417–432, 2007.

[8] L. Kuzniarz, M. Staron, and C. Wohlin. An empirical study on using stereotypes to improve understanding of UML models. In *12th IEEE International Workshop on Program Comprehension (IWPC'04)*, pages 14–23. IEEE CS, 2004.

[9] S. Misra, V. Kumar, and U. Kumar. Goal-oriented or scenario-based requirements engineering technique - what should a practitioner select? In *Canadian Conference on Electrical and Computer Engineering*, pages 2288–2292, 2005.

[10] J. Mylopoulos. Information Modeling in the Time of the Revolution. *Inf. Syst.*, 23(3-4):127–155, 1998.

[11] B. Nuseibeh and S. Easterbrook. Requirements Engineering: a roadmap. In *Proceedings of the Conference on The Future of Software Engineering (ICSE'00)*, pages 35–46, 2000.

[12] F. Ricca, M. Di Penta, M. Torchiano, P. Tonella, and M. Ceccato. The role of experience and ability in comprehension tasks supported by UML stereotypes. In *Proceedings of the 29th International Conference on Software engineering (ICSE'2007)*, pages 375–384. IEEE Computer Society, 2007.

[13] C. Rolland, G. Grosz, and R. Kla. Experience with goal-scenario coupling in requirements engineering. In *Proceedings of IEEE International Symposium on Requirements Engineering*, pages 74–81, 1999.

[14] A. Sutcliffe. Scenario-based requirements engineering. In *Proceedings of the 11th IEEE International Conference on Requirements Engineering (RE'2003)*, pages 320–329, 2003.

[15] A. van Lamsweerde. Requirements engineering in the year 00: a research perspective. In *Proceedings of the 22nd International Conference on Software engineering (ICSE'2000)*, pages 5–19, 2000.

[16] A. van Lamsweerde. Goal-oriented requirements enginering: a roundtrip from research to practice [enginering read engineering]. In *Proceedings of the 12th IEEE International Conference on Requirements Engineering (RE'2004)*, pages 4–7, 2004.

[17] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in Software Engineering - An Introduction*. Kluwer Academic Publishers, 2000.

[18] E. Yu. *Modelling Strategic Relationships for Process Reengineering*. PhD thesis, University of Toronto, Department of Computer Science, University of Toronto, 1995.