

Improving Requirements Specifications in Model-Driven Development Processes

Jordi Cabot and Eric Yu

Department of Computer Science, University of Toronto
{jcabot,eric}@cs.toronto.edu

Abstract: Understanding the organizational context and rationales (the “Whys”) that lead up to system requirements (the “Whats”) help us to analyze the stakeholders’ interests and how they might be addressed, or compromised, by the different design alternatives. These aspects are very important for the ongoing success of the system but are not considered by current Model-Driven Engineering (MDE) methods. In this paper we argue for the necessity of extending MDE methods with improved requirements techniques based on goal-oriented techniques for the analysis and specification of the organization context and discuss the benefits and challenges of such integration.

1. The Challenge

Understanding the purpose, goals, and intentions of a software system is a necessary condition for its successful design and implementation. By understanding the goals, the needs of the organization can be better aligned with the functionality provided by the system. However, organizations are complex in nature and in general, when attempting to design a system, there are usually many alternatives, each with different implications for the many parties (stakeholders) that may have an interest in the system. To identify, evaluate, and select the best alternatives is a considerable challenge but a key aspect for the ongoing success of the system.

Unfortunately, such an organizational-based analysis of the system’s requirements has not yet been adopted by current model-driven engineering (MDE) methods. In general, MDE methods limit the requirements specification to the elicitation of the functional requirements of the system (i.e. the behaviour/services/tasks the system-to-be must provide) but neither consider the organizational context of the system nor its non-functional requirements (NFRs) [10] (as usability, extensibility and so on).

We believe this restricted requirements analysis challenges the ability of current MDE methods to provide an accurate system representation at the end of the development process. As a result, the generated system may not satisfy the stakeholder’s expectations because their goals have not been sufficiently considered.

Therefore, in our opinion, providing a better support for the requirements specification and analysis is still a research challenge for MDE and one of the reasons that may explain the low adoption of MDE methods among software professionals.

In the next sections, we present the goal-oriented methods (Section 2) as a possible response to this challenge by means of integrating these methods within MDE approaches (Section 3). Finally, we discuss a list of open problems that must be solved to make this integration feasible (Section 4) and end up with the conclusions.

2. Goal-Oriented Requirements Engineering Methods

The requirements engineering community has largely addressed and recognized the leading role played by goals in the requirements engineering process. Nowadays, beliefs and goals (the “Whys” of the system) have been incorporated into most requirements acquisition frameworks, commonly known as goal-oriented requirements engineering (GORE) methods. A goal is an objective to be achieved by the system under consideration. Goals may refer to either functional concerns or NFRs.

It has been proven that an explicit goal modelling representation helps stakeholders in coming up with the initial requirements in the first place, always one of the most problematic tasks in any development process. By focusing on goals instead of specific requirements, analysts enable stakeholders to communicate using a language based on concepts (e.g. goals) with which they are both comfortable and familiar [2].

Other well-known benefits of using goal-based methods are their higher level view of the system requirements as compared with traditional requirements specifications; their link with business goals and decisions; the stability of goals compared with the requirements that implement them; the ability to consider alternative solutions and the potential to improve the traceability of the system by establishing clearer links between process design decisions and technical system alternatives [12], which also facilitates the system evolution and maintenance. Besides, GORE methods allow verifying and validating the requirements early in the development cycle thus avoiding propagating the errors and misunderstandings through the later phases of the process.

3. Integrating GORE and MDE methods

Given the characteristics and potential benefits of current GORE methods, we believe that they are a strong candidate to cover the gap of current MDE approaches when it comes to requirements specification and analysis.

In this sense, we think that an integrated software development process, where the typical phases of MDE methods (analysis, design and code-generation) are preceded by the early and late requirements phases of GORE methods (e.g. [7]), could overcome the limitations of MDE methods pointed out in the introduction. For specifying the two first phases we propose to use the specialized i^* modelling framework [16]. UML (or similar DSLs) are the logical option for the rest.

It is important to note that in order to maximize the benefits of such integration, goals should be taken into account in all phases of the development process: functional goals must be “realized” by the static and dynamic behaviour of analysis and design models while non-functional ones drive the selection of possible design alternatives when moving from one phase to the following. This is a distinction from

other recent works that have tried to bridge the gap between GORE and MDE methods by (partly) generating excerpts of an initial UML-based specification from the goals information [4],[11],[9,14].

In what follows, we provide a short description of each phase:

- **Early requirements phase.** Intentions (goals) of the actors (i.e. stakeholders) in the organizational context are modelled and analyzed. In this phase we explore why the system is needed, what alternatives might exist to attain each goal and what are the implications of each alternative for the stakeholders (i.e. what is their effect in terms of its contribution to the desired NFRs).
- **Late requirements phase.** The system is added as a new actor that can be in charge of fulfilling some of the desired goals. An analysis of the different alternative designs is performed and a final decision on the set of responsibilities to delegate to the system is made. From this assignment we may derive the list of functional requirements (tasks) for the system to be.
- **Analysis phase.** The static (i.e. structural) and dynamic (i.e. behavioural) aspects of the domain required by the software system to perform its functions are specified. Goals must still be taken into account when specifying these models. Most aspects can be specified using different semantically-equivalent alternatives, each one with its trade-offs. The selection of the most suitable one for the specific system under development should be influenced by the functional and non-functional goals that the system must fulfil.
- **Design phase.** The design phase adapts the previous models to the technical features of the technological platform where the system is going to be implemented. This implies defining a sequence of transformations that refine the models until all elements have a direct correspondence with the programming primitives available in the production environment. In general, all transformations are non-deterministic, in each refinement we have a number of possible design alternatives implying different non-functional qualities. The goals specified in the GORE models, in particular the non-functional ones, can help us to drive the transformation process and select the best alternative according to the requested NFRs.
- **Code Generation.** All software elements of the system (code, data structures,...) are generated from the design models.

4. Open Research Problems

Despite the potential benefits of mixing GORE and MDE methods, there are a number of open problems that must be solved to make their integration feasible in practice. In particular, the main challenge is to ensure that the benefits outweigh the extra work implied by the explicit definition of the goals at the beginning of the MDE process.

Therefore, our list of research challenges goes in the direction of smoothing the application of our approach by automating and simplifying as much as possible the transition between the different phases. A partial list is the following:

- Development of a common (meta)modelling framework. To bridge the gap between GORE and MDE models and ease the definition of model-to-model transformations between them, both kinds of models should be specified using “compatible” modelling languages. Current efforts try to formalize the i* language as a MOF-compliant metamodel (see [15], [3] and [1]).
- Reusing the information in the goal models to partially generate the static and dynamic aspects of the system. Existing approaches focus on the generation of preliminary use case diagrams, activity diagrams and class diagrams (see [8] [4],[11],[9, 14] [11] for more information). This generation must also consider the representation of the non-functional requirements. See [5, 13] for examples.
- Linking and traceability techniques between the business models (and business processes) and the analysis and design models and the architectural decisions. For instance, we should be able to detect, for each system goal, the subset of the design models that ensures the fulfilment of that goal.
- Incremental model synchronization techniques between all kinds of models.
- Consistency analysis between the different models involved in the process. Low-level models must be a consistent refinement of the higher-level ones.
- Propose a systematic way of dealing with NFRs in MDE (and in UML).
- Reusing the information about the system NFRs to influence in the analysis phase as, for instance, during the selection of the right analysis pattern to model certain aspects of the domain (e.g. to represent the *price* of a product we could just specify an attribute of type *Real* or use more complex patterns that facilitate recording the price in different money units; if *internationalization* is one of the desired NFRs for the system we should stick to the second option).
- Defining NFR-based model transformations to automatically transition from the analysis to the design phase. This transformation is not deterministic. Each modelling construct in the analysis models could be expressed using different design alternatives. Each alternative results in a different contribution to the non-functional qualities of the system. In our integrated approach, we could automate the transformation process by selecting the design alternatives that better match the specified NFRs in the requirements phases. This way, we also ensure that the resulting system is better aligned with the stakeholder’s interests.
- Establishing methods for adjusting MDE processes to the reality of the development team. An analysis of the knowledge and skills of team members and team structures will help to tune the development process so that we can maximize the team abilities and facilitate the process application.

5. Conclusions

We have identified the limitations of current MDE methods when it comes to understand the needs and goals of the organization and its stakeholders regarding the software system to be developed. This limited requirements analysis in most common MDE methods negatively impacts the quality of the generated system. We believe that improving this requirements support is still an important MDE research challenge.

As a possible solution we have proposed to extend MDE methods with goal-oriented requirements engineering techniques. We think this enriched development process may be better suited to produce software systems that satisfy the stakeholders' expectations. It may happen that using our approach only pays off for certain types of systems and/or domains and/or organizations. This needs to be empirically validated.

Acknowledgements

Work supported by the Spanish Ministry of Education and Science (project TIN2005-06053), the grant 2007 BP-A 00128 (Catalan Government) and the Natural Science and Engineering Research Council of Canada.

References

1. Amyot, D. URN metamodel proposal. Available: <http://jucmnav.softwareengineering.ca/twiki/bin/view/UCM/DraftZ151Metamodel>
2. Anton, A. I.: Goal Identification and Refinement in the Specification of Software-Based Information Systems. PhD Thesis. Georgia Institute of Technology (1997)
3. Ayala, C. P., Cares, C., Carvallo, J. P., Grau, G., Haya, M., Salazar, G., Franch, X., Mayol, E., Quer, C.: A Comparative Analysis of i*-Based Agent-Oriented Modeling Languages. In: Proc. SEKE'05 (2005) 43-50
4. Bertolini, D., Delpero, L., Mylopoulos, J., Novikau, A., Orler, A., Penserini, L., Perini, A., Susi, A., Tomasi, B.: A Tropos Model-Driven Development Environment. In: Proc. CAiSE Forum 2006, (2006)
5. Cysneiros, L. M., Leite, J. C. S. d. P.: Nonfunctional Requirements: From Elicitation to Conceptual Models IEEE Trans. Software Eng. 30 (2004) 328-350
6. Forward, A., Lethbridge, T. C.: Problems and Opportunities for Model-Centric Versus Code-Centric Development: a Survey of Software Professionals. In: Proc. MiSE'08 (ICSE Workshop), (2008)
7. Giorgini, P., Kolp, M., Mylopoulos, J., Pistore, M.: The Tropos Methodology: an overview. In: F. Bergenti, M.-P. Gleizes, and F. Zambonelli, (eds.): Methodologies And Software Engineering For Agent Systems. Kluwer Academic Publishing (2003)
8. Jiang, L., Topaloglou, T., Borgida, A., Mylopoulos, J.: Incorporating Goal Analysis in Database Design: A Case Study from Biological Data Management. In: Proc. RE'06, (2006) 196-204
9. Martínez, A., Pastor, O., Estrada, H.: Closing the Gap between Organizational Modeling and Information System Modeling. In: Proc. WER'03, (2003) 93-108
10. Mylopoulos, J., Chung, L., Nixon, B. A.: Representing and Using Nonfunctional Requirements: A Process-Oriented Approach. IEEE Trans. Softw. Eng. 18 (1992) 483-497
11. Perini, A., Susi, A.: Automating Model Transformations in Agent-Oriented Modelling. In: Proc. AOSE'05, LNCS, 3950 (2005) 167-178
12. Regev, G., Wegmann, A.: Where do Goals Come from? The Underlying Principles of Goal-Oriented Requirements Engineering. In: Proc. RE'05, (2005) 339-349
13. Salazar-Zarate, G., Botella, P., Dahanayake, A.: Introducing non-functional requirements in UML. In: UML and the unified process. IGI Publishing (2003) 116-128
14. Santander, V. F. A., Castro, J.: Deriving Use Cases from Organizational Modeling. In: Proc. RE'02, (2002) 32-42
15. Susi, A., Perini, A., Mylopoulos, J., Giorgini, P.: The Tropos Metamodel and its Use. Informatica 29 (2005) 401-408
16. Yu, E.: Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. In: Proc. RE'97, (1997) 226-235