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.