From Solution to Problem Spaces: Formal Methods in the Context of Model-Based Development and Domain-Specific Languages

Bernhard Schätz

fortiss GmbH
Guerickestr. 25, 80805 München, Germany
{schaetz}@fortiss.org

Abstract—With the increased use of model-based techniques and the provision of domain-specific languages, the focus of the development process is shifting from the implementation to the analysis and the design phase. With this shift from the general-purpose, technical-oriented solution space to the application-specific, domain-oriented problem space, new possibilities of application open up for rigorous engineering techniques, both on the analysis and on the synthesis side of applications.

I. MODEL-BASED DEVELOPMENT AND DOMAIN-SPECIFIC LANGUAGES

In many application domains, model-based development – the use of explicit models throughout the complete development process rather than the focus on code in the implementation phase – and domain-specific languages – the use of models for an application domain and specific purpose rather than a general-purpose language – has become the central software engineering paradigm. For example, recent studies in the automotive domain have shown that 96% of all participants use model-based development in both software design and implementation phase; the other 4% have just begun using model-based development and use it in the SW-design. In average 73% of a total application functionality is already developed model-based. Furthermore, 73% of the developed application software is being generated using auto-coding mechanisms.

These studies also show that model-based development substantially adds to cost improvements: The study participants report of cost savings around 27% and time savings around 36%, depending on the degree of expertise and intensity of usage of model-based development.

A closer look on these finding shows that these improvements are due to two important advantages of model-based techniques: First, the possibility of front-loading quality assurance techniques, i.e., the shifting of these techniques to the earlier phases due to the availability of explicit specification models in those early phases. In case of the automotive development process, for example, data flow models of the control algorithms in the early phases are suited for simulation or execution, allowing to validate models via rapid control prototyping. The second possibility of model-based development is the support for automated design space exploration, i.e., the automation of specific design decision due to the separation and provision of individual views of the system under development. Again taking the case of the automotive development process as an example, separate views of the logical software architecture and the physically distributed hardware are used, allowing to generate suitable bus configurations for the deployed software components.

II. ENHANCED POSSIBILITIES BY FORMAL DEVELOPMENT TECHNIQUES

The use of several, explicit models throughout the development process and the restriction of the expressiveness of these models to specific views and purposes allows to structure and segment the development process in specific development tasks and to provide dedicated techniques for these individual tasks. Here, techniques used in the area of formal methods can be applied to further substantially improve the efficiency and effectiveness of the development process beyond simulation and auto-coding.

A light-weight but nevertheless very helpful approach consists in exploiting the fact that domain-specific models – like in the rail domain – provide a rather rich conceptual structure, reflecting the concepts – like track, signal, gate – of their application domains. Here, structural well-formedness conditions, capturing the modeling constraints of this specific application domain in a language like OCL, can help to avoid unreasonable models by automatically detecting violations of these constraints.

Besides on the structural – or syntactic – level, dedicated analysis techniques can also be provided for specific semantic constraints. Here, again, the explicit representation of domain-specific aspects – for example signaling properties distinguishing between event and state signals – can be exploited to automatically check the consistency – for example in the form of contracts between the sender and receiver – of these different aspects using model checking.

Besides providing specific analysis techniques, formally-based techniques can also be used to support design-space exploration. Here, by consistently combining the different views of the system – like the logical view of a system with timing requirements for its components and the physical view with its computation and communication resources – additional views – like the deployment of these components to the resources – can be synthesized using SMT solvers, often ensuring optimality of the solution.

Similarly, these formally-based techniques can also be used to automatically synthesize new models for specific purposes. Again, by making use of the explicitly available models – like the architecture of a system and the behavior
of its components – solver techniques can be used to deduce specific aspects – like a system-level test case for a component defect – from these models.

In total, the application of formal techniques to a model-based development process with its explicit, overlapping models restricted to specific purposes, offers substantial potential to improve the efficiency and quality of the development process.