07241 – Summary
Tools for the Model-based Development of Certifiable, Dependable Systems

June 10th to June 15th 2007
— Dagstuhl Seminar —

Michaela Huhn[1], Hardi Hungar[2], and Doron Peled[3]

[1] Technische Universtät Braunschweig, Carl-Friedrich-Gauß-Fakultät
D-38106 Braunschweig, Mühlenpfordtstr. 23, Germany
M.Huhn@tu-braunschweig.de
[2] OFFIS, Safety Critical Systems
D-26121 Oldenburg, Escherweg 2, Germany
hungar@offis.de
[3] University of Warwick, Department of Computer Science
Coventry, CV4 7AL, United Kingdom
doron@dcs.warwick.ac.uk

Abstract. This paper summarizes the objectives and structure of a seminar with the same title, held from June 10th to June 15th, 2007 at Schloss Dagstuhl, Germany.

Keywords. Dependable systems, safety, security, certification, formal methods, modelling, verification, tools.

1 Introduction

The development of software for dependable systems on which the safety or security of individuals, organizations, and property may rely has become an important application and research field. In many cases, law enforces certification as a prerequisite for the introduction of a technical, dependable system. The certification formally assures that the systems and its development process meet the technical standards and all efforts have been made to reduce the risks. The complexity and discrete nature of software makes an assurance of the required properties of the software components of a dependable system extremely difficult. This applies even more as the software often constitutes the control in a so-called embedded system where coupling electronic and mechanical components is still a challenge. The relevant national and international standards (e.g. IEC 61508 [Int00]) recommend formal methods for the development of systems that shall be certified for the higher safety integrity levels. However, deriving constructive design guidelines and formally verifiable software constraints from safety requirements on the system level is still a challenging problem.

In many industries developing high-assurance products, model-based design is already well established and a number of tools used in safety-critical software design are founded on formal semantics and support in parts automated code generation, formal analysis, verification or error detection. But certification requires more than formal semantics of a modelling notation or the formal verification of the artefacts of a particular development step. Formal methods and tools have to be embedded in a seamless design process which covers all development phases and which includes an efficient construction of a safety case for the product. Moreover, whereas most (semi-)formal modelling approaches focus on functional issues additional concepts are required for dependable systems like fault tolerance, timing or security. Even if these concepts are addressed – as several UML profiles do – they are supported at most rudimentarily by the tools.

1.1 Some Statistics

The workshop aimed at bringing together people actually working on the certification of dependable systems with researchers who develop and validate (semi-) formal methods and tools for modelling and verification. About 30 researchers and practitioners followed the invitation. The program featured about 25 presentations, 14 research presentations from academia, 4 reports on experiences in the certification processes of particular products and 6 about practical issues from industrial participants. In addition, a discussion session on the status quo of formal methods in the certification of dependable systems was arranged.

1.2 The Railway Level Crossing Case Study

To focus discussions and to directly compare different approaches a case study on the design of the software control for a simple railway level crossing [HHLZ07] was given to the participants beforehand. Several participants addressed the case study in their presentation. So it was shown that some immediate shortcomings in the requirement specification could be detected with simple static analysis techniques applied on an abstract design model. This observation was independent of formal modelling notations actually used. A more subtle ambiguity was detected by those approaches that employed formal verification on a behavioural model of the level crossing. Another participant used a generalized version of the case study to prove scalability of a verification approach. Last but not least, the case study has set up the discussion to what extend implicit assumptions are revealed when transferring an informal requirements specification into a formal model and how to deal with different categories of implicit assumptions methodologically.

2 Discussion on the Status Quo of Formal Methods and Certification of Dependable Systems

On each of the issues raised during the discussion, there was a wide spectrum of partly divergent opinions and observations. A summary of each of the main themes follows below.

2.1 Mathematical Rigour

On the one hand, it was reported, that all necessary techniques were available to treat the subject satisfactorily. By a relevant number of pilot projects it has been proven that systems of impressive size can be verified working with theorem proving or model checking on the basis of a cleanly and well defined semantics for systems and problem statements. A decomposition into well-built layers, each verified itself, and dedicated construction of relations between layers, was considered to be a successful approach. This goes very well with the model-based design paradigm, because most model-based development processes provide modelling guidelines for different levels of abstraction. The proponents of a rigorous formal approach were convinced that even the popular counter-arguments - as ongoing problems with scalability of formal verification, the high investments needed to develop the initial foundation of a formal process (e.g. verified layers), and the severe shortage of experts who have profound knowledge on formal methods and the application domain - would merely be surmountable obstacles.

This positive view was not shared by everybody. First of all, a number of participants considered the mentioned obstacles to be substantial. Additionally, the maturity of formal methods and tools was challenged: So far, a relevant portion of the success stories was achieved with prototype tools in a research environment. These are not easily transferred into the fields of practice, not only because matters like tool support, robustness and scalability, and domain specific adaptations have to be solved for industrial development. The most important barrier for introducing academic tools in dependable system industries is that certification imposes high requirements not only on the products and the process but also on each tool used in the development and in particular for quality assurance.

2.2 Verification and Validation Tools

There was a general agreement that tool support for verification and validation (V&V)is indispensable and will grow in its importance in the following years. There were of course, again, differing viewpoints.

One observation concerned the introduction of new analysis methods like hybrid model checking into the field of (control) engineering. Up to now, these model checkers have not yet shown much promise to be able to handle even medium size designs. Are, henceforth, the "classical" engineering techniques in this field going to be replaced in the foreseeable future? And how are they, then, going to be integrated with formal specifications and reasoning?

A similar integration concern has been raised with respect to the growing usage of automatic simulation or test generation tools. Though considerable progress has been made in automation techniques, it is difficult to profit from a tool which does not guarantee complete coverage. Related to this issue is the question of how to find the "interesting" points via simulation or testing? Hence, these techniques have to be equipped with means to direct the search to those parts of the state space where complex and error-prone behaviour occurs, i.e. interactions between components from different suppliers, or mode transitions in the presence of exceptional behaviour. Moreover, the technique should be able to identify dead code or unreachable parts of the state space or other anomalies indicating potential errors.

To summarize, it shows more promising to specifically tailor a tool to a particular purpose explicitly required by regulations than (naively) introducing an unspecific test generation tool into the V&V process. For example the avionics standard DO-178B requires MC/DC coverage, which can be addressed by a dedicated procedure relying on abstract interpretation.

If requirements are not that strict, as for instance it is often the case in the automotive domain, it is somewhat easier to profit from the availability of high-level models. This argument gets detailed in the section on processes.

2.3 Specification

The question of whether specification has to be formal is controversial. Clearly, formality helps to uncover errors and to avoid misinterpretation, and eases subsequent automation and application of formal techniques. Additionally, the ubiquitous UML [Obj05] carries the promise of an emerging de-facto standard of becoming a lingua franca, understood across domains.

There are obvious objections concerning the communicability of formalisms and the difficulty of checking consistency and completeness. These are more severe deficiencies in a formal than in an informal specification, since the latter appeals more to the developer's experience and competence. Also, it was said that specification is "inherently an iterative process", where some aspects cannot be fixed before at least a prototypical exploration of the design is applied. That specification cannot be fixed up front is experienced even in cases where standards require this explicitly, like the EN 50128 [CEN01]. Often, customers require late changes to a product. To incorporate such changes into a specification is usually more difficult when the specification is formal.

Finally, UML is not naturally suited for every kind of application: Parts of the UML are not unambiguously defined, others presume a particular model of execution as for instance statecharts [HK04] with event pools and run-to-completion semantics, that does not match the semantic world of every application. For modelling the hardware architecture appropriately, which is an integral part of most design and analysis tasks in dependable system development, the UML adoption SysML [Obj07] plus profiles like MARTE [Obj06] are needed.

There was wide agreement that there is no one language for formal specification, nor could there be one. Application-specific languages certainly have their place, as well as weak formalisms like the UML for communication purposes. Whether one should strive for full formality depends at least today on the means available and on the requirements.

2.4 Process

The specification is only one of the artefacts to be produced in a design process. The seminar's focus was on dependability and safety, so their role was prominently discussed. In several domains standards prescribe a process skeleton; and a number of well established industrial processes for developing dependable and certifiable systems instantiate and refine these skeletons.

Typically, certification and thus development processes for dependable systems necessitates construction of an argument of safety in a formalised document – a so-called *safety case*. It is considered advisable to take specific care so that the safety case construction can be done hand-in-hand with the development, i.e. integrate safety arguments with specification refinement and implementation. Today, this is not yet done satisfactorily efficient. Improvements will most likely be possible on a basis of formal development artefacts. Some safety concerns give rise to additional functional requirements, e.g., when redundancy or watchdog constructs are introduced as safety measures. Here, a tight integration of the design steps and the construction of the safety case is particularly recommended.

Another issue is the structure in the line of arguments in the safety case itself. Approaches like *Goal Structuring Notation* [Kel98] improve the conciseness and understandability of reasoning. Moreover, context information and constraints, limiting the applicability of a particular argument or reasoning method, can be made explicit. However, full integration with model-based design methods or formal V&V methods is missing so far.

In the automotive area, a domain-specific standard for the development of safety related electronic programmable systems is still pending (e.g. ISO WD 26262). Here, the development of automotive systems is performed according to an iterative, approximative process. First a core control model is developed on the basis of an ideal plant model. In the next step the normal behaviour of the control model is explored and validated in the real plant. Then exceptional behaviour and specific case treatment is added to the control model. Then a fine-tuning and optimization phase follows. Finally, the control model is transformed into target specific code, which is validated against the runs of the ideal model. In all phases, design proceeds based on the division of labour between OEM and suppliers.

3 Conclusion

It was commonly agreed that formal specification has already reached the stage of being an effective support in the development of software-intensive dependable systems and that its role will increase in the future.

Technical progress in verification, for instance in component-oriented reasoning (assume-guarantee proofs) and (semi-)automated abstraction techniques, significantly expand the potential for applying formal methods on complex systems.

A tighter integration of models for design and models for verification has already begun and proven to be a key factor for the introduction of formal methods into the industrial practice. A number of verification approaches directly start with design models from UML or Matlab/Simulink as analysis input and offer seamless tool integration. However, these methods are often restricted to one particular verification goal that is considered relevant in one design phase or to a single concern like functional correctness or timing. Thus, a major challenge for the future will be to integrate formal approaches dealing with the different concerns that contribute to safety like functional correctness, reliability, timing, et cetera.

References

CEN01. CENELEC. EN 50128: Railway applications - Communications, Signalling and Processing Systems - Software for Railway Control and Protection Systems. European Standard., March 2001.

HHLZ07. Hardi Hungar, Michaela Huhn, Oliver Lemke, and Axel Zechner. Tools for the model-based development of certifiable, dependable systems: Case study railway level crossing, June 2007.

HK04. David Harel and Hillel Kugler. The Rhapsody semantics of statecharts (or, on the executable core of the UML). In H. Ehrig et al., editor, *Integration of Software Specification Techniques for Applications in Engineering*, number 3147 in LNCS, pages 325 – 354, 2004.

Int00. International Electrotechnical Commission. IEC 61508 Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems – Part 7: Overview of Techniques and Measures, March 2000.

Kel98. Timothy Kelly. *Arguing Safety – A Systemic Approach to Managing Safety Cases*. PhD thesis, University of York, September 1998.

Obj05. Object Management Group. Unified Modeling Language: Superstructure, August 2005.

Obj06. Object Management Group. UML Profile for Modeling and Analysis of Real-Time and Embedded Systems (MARTE) rfp, 2006.

Obj07. Object Management Group. SysML specification v. 1.0, August 2007.