Using model-based analysis in certification of critical software-intensive systems

Frank Ortmeier, Simon Struck and Michael Lipaczewski
Computer Systems in Engineering
Otto-von-Guericke University of Magdeburg

Abstract: Software is taking over more and more functionality in most technical systems, which leads to the term software-intensive or cyber-physical systems. Although this offers many exciting new opportunities, it also makes precise analysis of safety and reliability goals much more complicated. Well-known traditional techniques often reach their limits. Model-based approaches on the other hand can be useful for solving some of these problems.

However, in industrial practice answering the question alone is often not sufficient. It is also necessary to explain how answers were found. In this paper, we will show some of the capabilities of modern model-based analysis methods and highlight how they possibly could be used in safety engineering resp. what obstacles need to be avoided.

1 Introduction

Safety critical applications need certification prior to system launch. Based on a safety analysis the certification ensures that the system will not cause any harm to human beings. Considering modern software-intensive systems the safety analysis is a challenging task. Research developed new analysis methods to cope with the increasing complexity. However these methods and techniques face some difficulties in the traditional certification process. In this paper we point out the current state, advantages and difficulties of the application of modern model-based safety-analysis techniques in the certification process of software-intensive safety-critical systems. Section 2 introduces software intensive systems. Section 3 gives an overview of the certification process and Section 3.1 points out how the safety analysis is used during certification. After that Section 3.2 introduces chances and challenges of modern model-based safety analysis techniques. Finally Section 4 summarizes the content presented in the paper.

2 Software-intensive systems

A *software-intensive system* is a technical system in which important parts are realized in terms of software and thus the software interacts with other systems, the environment and human beings [WH06]. This especially implies embedded computer applications. Such systems often consist of sensors, a programmable processing unit and actors. Besides the

basic physical model of the sensors and actors the behavior of the system is heavily dependent on the programming of the control unit.

Along with decreasing costs and increasing capabilities of semiconductor technology, software-intensive systems become more and more important in modern engineering. An important aspect of many software-intensive systems is their nature as reactive systems. Based on sensor values the system controls its actors so that it properly reacts on the input values. If the system also covers an internal state its output not only depends on the current input, but also on a history of input values. Reactive systems often cannot be analyzed in isolation, because their output typically influences their inputs via the systems environment. Therefore it is mandatory to analyze the system together with its surrounding components. The environment may cover other system components as well as the surrounding nature.

3 Certification process

In general, the goal of certification is to ensure the correctness and safety of a system. For successful certification a comprehensive a-priori statement about the system safety is necessary. This means the safety of the system is demonstrated and documented in an objective way prior to the system launch. Depending on the stakeholder there are different reasons why (safety critical) systems require certification:

- The producer of a system (i.e. the company that develops and builds the system) needs the certification as permission to sell/install/use the resulting product. Of course a company is also interested in the safety of their products (as this may have impact on the reputation), but basically they get paid for the product and not its safety. So the main goal of producers is to minimize the risk of failing to successfully certify their system.
- The certification body (usually in terms of a national agency) has to ensure the safety of the society and especially all human beings. The main interest of the certification body is in certifying *only* solidly safe systems. The secondary (but very important) interest is to make an *objective* decision. In particular, a-posteriori (after an accident has happened) certification bodies need to show how and why they approved the system. So their main goal is to make legally solid and objective arguments as claims for certification.

These are somehow contradictory interests from which different requirements can be derived. The company has to demonstrate the safety of their products and wants to do this by minimal costs. On the other hand there is the certification body which wants to see a comprehensive and objective demonstration about the safety of the product in question.

Mathematically spoken, verification of software can be used to ensure the functionality of a software system with respect to a specification (for example Hoare logic). This proves the correctness of the system in a mathematical way and is thus absolutely correct. However,

deducing a mathematical proof is often very complicated and time consuming. In the case of complex systems it even might not always be feasible. In any case, it is a very expensive task and only done for small but highly critical system components.

In addition, mathematical verification of a system is often hard to understand. The lack of traceability as well as high complexity of the (mathematical) reasoning makes understanding of this type of safety arguments hard. This is especially a problem when the safety case needs to be accepted by the certification body. Imagine that an accident with the system happened and an a-posteriori analysis showed an error in the mathematical analysis. Who takes responsibility then? The analyst (for making the error), the safety manager (for not finding the error) or the certification body (for allowing hard to understand/evaluate arguments). Things become even worse if arguments have been calculated (semi-) automatically, so that the designer/producer of the tool can be held responsible (for distributing erroneous software).

3.1 Safety analysis and certification

Safety analysis is the process of a-priori measuring/determining the safety of a system. Certification bodies often rely on standardized tools and analysis techniques. Different standards, with the focus on specific needs form different domains, exist. The IEC 61508 [Lad08] as a generic norm, DO178B/C [RTC92] for avionics and the upcoming ISO26262 [Int09] for automotive. However, in practice safety analysis is not only done by certification bodies nor only during certification.

Figure 1 depicts a certification process from the railroad domain. The system is developed by the system engineers (not depicted in the figure) led by a technical project leader. The quality manager observes the overall design and implementation process and organizes the (software) tests. The validator evaluates the results of the (software) tests and compares the results with hardware/system in the loop tests. More important for the context of this paper is the safety manager. He coordinates the safety related processes within the project. The technical project leader exchanges design documents and implementation plans with the safety manager and in return gets early feedback on the system's safety aspects. On the other hand, the safety manager combines the quality management report and the design related documentation into a safety case that is passed to the safety assessor. The safety assessor evaluates the overall systems safety by examining the validation plan and validation report from the validator and the safety case from the safety manager. As a result the safety assessor creates a safety assessment that is passed to the certification body. If the safety assessment is accepted, the system gets certified.

The interaction between the technical project leader, the safety manager and the safety assessor relates to the two different interests for certification mentioned in the last section. The safety manager gives feedback to the system engineers and thereby helps to pass the safety certification. On the other side the safety manager provides important input for the safety assessment and thus fulfills the requirements from the certification body.

For the safety assessment to be accepted it must provide comprehensive information about

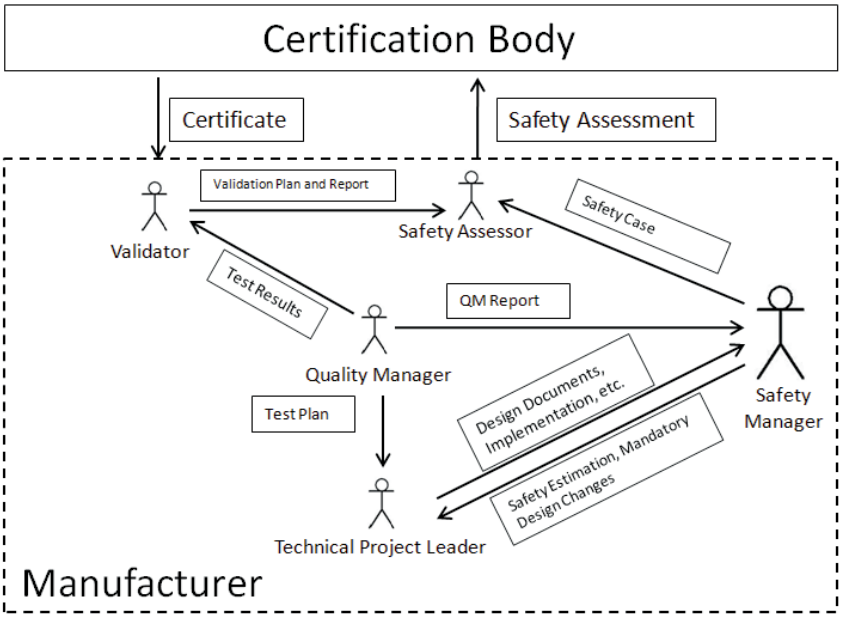

Abbildung 1: Interaction between all participants in the certification process

the systems safety. This implies that the used analysis techniques are comprehensible, even to people who are not experts in safety analysis domain. Therefore safety engineers often rely on common safety analysis techniques like fault tree analysis (FTA) [VDF+02] and failure modes and effects analysis (FMEA) [MMB96]. Besides being standardized, these analysis techniques have proved their use in numerous of applications and are easy to understand.

3.2 Chances and challenges for model-based methods

However, traditional safety analysis methods have some disadvantages. They were mostly developed decades ago. Thus they were designed for mechanical systems with only few or no software and might no longer fit the needs of modern software-intensive systems.

Model-based safety analysis methods where developed to cope with software-intensive systems. There are several modeling languages and safety analysis tools available, that were already successfully applied during system development: COMPASS/SLIM[BCK+09] and SCADE/Lustre[1] and some others.

A clear advantage of the model-based analysis methods is that they are highly automated. Thus they are performed very fast, and cost efficient. This is a great advantage in modern

[1]Feb. 2012: `http://www.esterel-technologies.com/products/scade-suite/`

software development processes, where the analysis/evaluation is not only applied once at the end of the development. As a highly automated process, model-based analysis is also less error prone and also leads to more accurate results. Finally the close relation between the model and the design/implementation helps to improve the maintainability of the model.

If model-based analysis methods are used during the system design time, they can give very early and very precise feedback about the system safety and thus assist the design process. Imagine a design related safety issue. If the issue is only found during the certification (where safety analysis must be done), it is often very expensive to correct a failure. However if model-based safety analysis points out that there is an issue, this can be considered in the design process and thus will not lead to complications in the certification process. This fits with the interests of the project/company which wants to pass the certification on the first approach.

The main challenge for the application of model-based analysis techniques is the demonstration of its cost-benefit ratio. Due to the deterministic and highly automated process more accurate results are possible. These can even be gained at lower costs than with traditional analysis methods. In contrast to the certification body the overall costs are one of the main issues for the company developing the system.

Objectively measuring the safety of a system, also meets the demands of the certification body. The main challenge for the application of model-based analysis during the certification process is that they are hardly accepted by the certification body. On a long term the model-based analysis techniques should be qualified as tool for safety certification. This, however, is a difficult and long process. On shorter term, the model-based analysis can be used in combination with traditional analysis methods. Imagine a FTA, where the safety engineer has to decide how detailed the analysis is performed. This can be a serious source of failure in the analysis. If the results of the FTA are evaluated with the results from a model-based analysis, this could point out where the safety engineer abstracted to far from the system. The FTA can then be corrected, and thus becomes more reliable. In this scenario the certification body still gets a traditional analysis as safety case.

4 Conclusion

Safety analysis is used during the design time and for the certification of safety critical systems. Traditional analysis techniques do not cope with the complexity of modern software-intensive systems. Thereto model-based analysis techniques where created. However, these methods are barely used in industrial practice.

Model-based analysis techniques can assist during the design of a system. Possible design failures can be detected and corrected early in the development process. Thus the system can pass certification without expensive corrections (in a late development phase). This decreases production costs, and thus suits the economic interests of a company that is producing safety-critical systems.

After the development of a safety critical system, certification is required as permission to

sell/launch the system. The certification process uses safety analysis to measure/determine the system safety. Despite the advantages of model-based analysis techniques, they are even less used in the certification process than during the system design. Qualifying model based analysis is a long and difficult process. An intermediate approach is to combine model-based and traditional analysis methods. Thus the accuracy of the model-based analysis can be expressed in terms of traditional analysis methods, which are accepted by the certification body.

Acknowledgments

Michael Lipaczewski is sponsored by the Deutschen Ministerium fr Bildung und Forschung in the ViERforES project (BMBF, project-Nr.: 01IM08003C).

Simon Struck is sponsored by the German Research Foundation (DFG) within the ProMoSA project.

Literatur

[BCK+09] M. Bozzano, A. Cimatti, J.P. Katoen, V. Nguyen, T. Noll und M. Roveri. The COMPASS approach: Correctness, modelling and performability of aerospace systems. *Computer Safety, Reliability, and Security*, Seiten 173–186, 2009.

[CCG+02] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani und A. Tacchella. NuSMV Version 2: An OpenSource Tool for Symbolic Model Checking. In *Proceedings of the* 14^{th} *International Conference on Computer Aided Verification (CAV 2002)*, Jgg. 2404 of *LNCS*. Springer, 2002.

[Est11] Esterel Technologies, http://www.esterel-technologies.com/?id=13281, accessed Feb. 28. 2011. SCADE Suite, 2011.

[GO10] Matthias Güdemann und Frank Ortmeier. A Framework for Qualitative and Quantitative Model-Based Safety Analysis. In *Proceedings of the* 12^{th} *High Assurance System Engineering Symposium (HASE 2010)*, Seiten 132–141, 2010.

[HCRP91] N. Halbwachs, P. Caspi, P. Raymond und D. Pilaud. The synchronous data-flow programming language Lustre. *Proceedings of the IEEE*, 79(9):1305–1320, September 1991.

[Int09] International Organisation for Standardization. ISO 26262: Road Vehicles-Functional Safety, Draft International Standard (DIS), 2009.

[KNP02] Marta Kwiatkowska, Gethin Norman und David Parker. PRISM: Probabilistic Symbolic Model Checker. In T. Field, P. Harrison, J. Bradley und U. Harder, Hrsg., *Proceedings of the* 12^{th} *International Conference on Modelling Techniques and Tools for Computer Performance Evaluation (TOOLS 2002)*, Jgg. 2324 of *LNCS*, Seiten 200–204. Springer, 2002.

[Lad08] Peter B. Ladkin. An Overview of IEC 61508 on E/E/PE Functional Safety, 2008.

[MMB96] Robin E. McDermott, Raymond J. Mikulak und Michael R. Beauregard. *The Basics of FMEA*. Quality Resources, 1996.

[ORS06] F. Ortmeier, W. Reif und G. Schellhorn. Deductive Cause-Consequence Analysis (DCCA). In *Proceedings of the* 16^{th} *IFAC World Congress*. Elsevier, 2006.

[RTC92] RTCA. DO-178B: Software Considerations in Airborne Systems and Equipment Certification, December, 1st 1992.

[VDF+02] Dr. W. Vesley, Dr. Joanne Dugan, J. Fragole, J. Minarik II und J. Railsback. *Fault Tree Handbook with Aerospace Applications*. NASA Office of Safety and Mission Assurance, August 2002.

[WH06] M. Wirsing und M. Holzl. Software intensive systems. *Draft Report on ERCIM âBeyond the Horizon Thematic Group*, 6, 2006.