Technical Report soft-11-01, Chair for Software Engineering, University of Konstanz

The QuantUM Approach in the Context of the ISO Standard 26262 for Automotive Systems

– Extended Abstract –

Florian Leitner-Fischer and Stefan Leue

University of Konstanz and

Steinbeis Transfer Center for Complex Systems Engineering

Florian.Leitner@uni-konstanz.de, Stefan.Leue@uni-konstanz.de

February 28, 2011

Abstract

The forthcoming standard ISO 26262 defines processes and techniques in support of a safe design and implementation of automotive systems. We comment on the recommendations that this standard provides with respect to the use of semi-formal and formal methods, including formal verification, during various stages of the proposed safety process. We illustrate how the QuantUM method and tool that we have developed in order to open UML-type system architecture models to formal analysis using stochastic model checking can be applied in support of the safety requirements imposed by the standard.

1 Introduction

The forthcoming standard ISO 26262 [12] defines processes and techniques in support of a safe design and implementation of passenger automobile systems containing safety relevant or safety critical programmable electronic components. It is hence the standard applicable to automotive systems and software engineering in the domain of passenger cars as far as safety relevant aspects of these systems are concerned. ISO 26262 can be viewed as an application domain specific specialization of the standard IEC 61508 [11] that is applicable to general programmable, safety-critical electronic systems. ISO 26262 will be considered to represent the state of the art with respect to functional safety processes and techniques in the automotive system engineering area, and is hence likely to become a point of reference for litigation in this domain.

The objective of this extended abstract is to discuss and illustrate how semi-formal and formal design methods can be used to support the design process of safety-critical automotive systems. We briefly present the QuantUM [14, 13] approach and tool that permits UML based system models to be annotated with quantitative failure characteristics. The annotated models will be automatically analyzed using the stochastic PRISM model checker [8]. The results will be

displayed as fault trees (FTs) [17] as well as sequence diagrams. The synthesized PRISM model can be used as a basis for probabilistic Failure Mode and Effects Analysis (pFMEA) [7, 1]. We then review the ISO 26262 standard, and discuss how semi-formal and formal methods and in particular QuantUM can support the safe design of systems in keeping with the process requirements imposed by this standard.

Related Work. Work described in [5] and [6] discusses how model-based software development using semi-formal modeling languages like the UML or the Matlab/Simulink modeling language can be used in a ISO 26262 development process. Other than that, we are not aware of any other work that relates ISO 26262 to concrete formal or semi-formal techniques. The use of formal methods in the context of the forthcoming avionics standard DO-178C is being discussed in [4].

2 The QuantUM Approach

The QuantUM approach allows for the quantitative safety analysis of UML models. It can be used to answer questions such as "does the system satisfy its requirements in 99.99% of the time?" In QuantUM all inputs of the analysis are specified at the level of a UML model. To facilitate the specification of reliability and dependability characteristics, we propose a quantitative extension of the UML. This extension is defined in terms of a UML Profile and takes advantage of the UML concept of a stereotype. A comprehensive description of the QuantUM profile can be found in [13].

Due to the fully automated nature of our approach we were able to implement a tool that completely hides the analysis model and the formal methods used during the analysis from the user. Our approach is depicted in Figure 1. It can

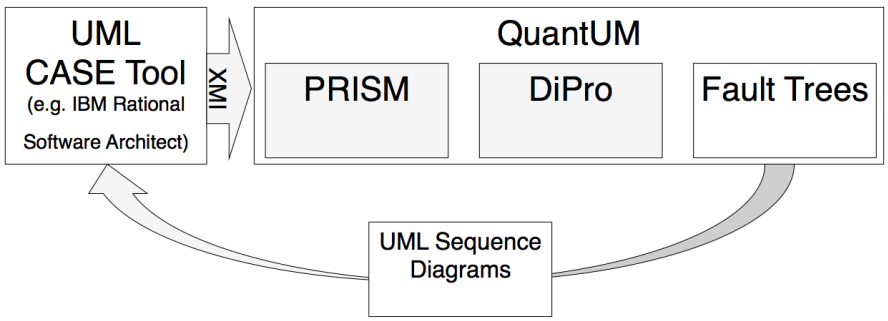

Figure 1: Overview of the QuantUM Approach.

be summarized by identifying the following steps:

- The QuantUM extension is used to annotate the UML model with all information that is needed to perform a dependability analysis.
- The annotated UML model is then exported in the XML Metadata Interchange (XMI) format [15], which is the standard format for exchanging UML models.

- Subsequently, our QuantUM Tool parses the generated XMI file and generates an analysis model in the input language of the probabilistic model checker PRISM. The tool also generates a formal specification of the properties to be verified using a probabilistic temporal logic.
- For the analysis we use the probabilistic model checker PRISM together with our counterexample generation tool DiPro [2] in order to compute probabilistic counterexamples representing paths leading to a hazard state.
- The resulting counterexamples can then be transformed into a fault tree that can be interpreted at the level of the UML model. Alternatively, they can be mapped onto a UML sequence diagram which can be displayed in the UML modeling tool containing the original UML model.

The QuantUM extension of the UML allows for a direct annotation of UML models within the case tool that is used to design the system. This allows for an convenient integration of the QuantUM approach into the development process of safety-critical systems. The following section is devoted to an overview on possible application scenarios of the QuantUM approach in an ISO 26262 compliant development process.

3 QuantUM and ISO 26262

We discuss how the ISO 26262 standard is related to the use of formal methods, and how the QuantUM approach can support the proposed functional safety objectives and techniques. The standard is structured into nine parts that address different stages of the system design process.

- Part 26262-1 presents a *definition of terms* used in the standard. In the context of our paper it is most interesting how terms related to formal methods are defined. A *formal notation* is defined as a description technique that possesses completely defined syntax and semantics. A notation is defined to be *semi-formal* if only its syntax is precisely defined. The UML would hence qualify as a semi-formal notation. *Formal verification* is defined as a method that allows one to prove correctness of a system against its specification. A verification method is *semi-formal* if it is based on specification given in a semi-formal notation. Following this terminology, our paper uses both semi-formal notations (UML and its derivative QuantUM) as well as a formal notation (the PRISM model checker input language) and formal verification techniques (stochastic model checking.)
- Part 26262-2 defines requirements on the *management* of functional safety. The standard defines different confirmation measures to be performed including confirmation reviews, functional safety audits and functional safety assessments. These process steps are designed to compile a safety case, which is defined to be an argument that the safety goals for an aspect of the system[1] are satisfied based on the functional safety assurance measures carried out. Amongst others, Table 1 in 26262-2 requires confirmation reviews using Fault Tree Analysis (FTA) [17] and Failure Modes and Effects

[1]Instead of "aspect" the standard uses the term *item* to denote a system, a collection of systems or a function to which ISO 26262 is applied.

Analysis (FMEA) [10] to be performed. For ASIL-A and ASIL-B this is merely required, for ASIL-C this analysis shall be performed by a person not belonging to the development team, and for ASIL-D by a person not belonging to the same organization or department within an organization. Note that both FTA and FMEA avail themselves of formal analysis methods and that QuantUM supports the automated synthesis of models required to carry out FTA and FMEA. In general, an automation of the compilation of safety cases can significantly enhance the analysis since it makes the compilation less error prone and more cost effective.

- Part 26262-3 is devoted to the *concept phase* of the system development life cycle. This stage is pivotal for the functional safety of the system since it involves the planning of all safety assurance activities. In particular, it comprises the definition of the different system aspects that are safety relevant, the initiation of the safety process, a hazard analysis and a risk assessment. Tables 1 to 4 of 26262-3 define a qualitative method to determine the Automotive Security Integrity Level (ASIL) of a system aspect, which ranges from A (lowest safety-criticality) to D (highest safety-criticality). 26262-3 finally requires the derivation of a functional safety concept from previously elicited functional safety goals, consisting of a list of functional safety requirements. We contend that the hazard analysis and risk assessment activities can be well supported by a formal analysis approach, such as the one using QuantUM that we describe below. Clause 8.4.5 of 26262-3 requires the resulting functional safety requirements to be evaluated with regard to their effectiveness using tests, trials, prototyping and simulation. We maintain that a definition of safety requirements based on semi-formal and formal techniques will greatly enhance this activity, for instance by performing model checking runs.

- Part 26262-4 is concerned with the *system design* and with ensuring that the system design satisfies the technical safety requirements. Formal verification techniques such as model checking and theorem proving are prime candidates to establish such a satisfaction relation between a design and a set of requirements. 26262-4, however, does not mention these techniques. Table 1 of 26262-4 proposes the use of deductive analysis techniques, such as FTA, and qualifies them as highly recommended for ASIL C and D. Inductive Analysis techniques, such as FMEA, Event Tree Analysis (ETA) and Markov modeling are highly recommended for all ASIL levels. We note that hence QuantUM based probabilistic analysis is well suited to support system design, in particular if it is extended to system design languages such as SysML [16]. We also note that QuantUM allows for the automatic generation of fault trees from design models, and provides support for carrying out probabilistic FMEA.

- Part 26262-5 is devoted to *hardware design*. While we note that hardware analysis can be carried out by model checking [3], our UML-based approach is not really suitable to describe hardware architectures. We nonetheless believe that there is great potential for supporting hardware level functional safety using formal analysis methods. Clause 9.4.3 requires a probabilistic metric for random hardware failures. If the component level of the hardware architecture is modeled with UML (or SysML),

our QuantUM approach can be used to compute such probabilistic metrics automatically.

- Part 26262-6 defines the *software development process* including the definition of software safety requirements and their verification. Figure 2 of 26262-5 defines the V-model [9], adapted to accommodate software safety requirements, as the applicable software process and lifecycle model. Software safety requirements "are derived from the technical safety concept and the system design." Further objectives of this stage are a refinement of the software-hardware interface, and an assurance mechanism showing that the software safety requirements are "consistent with the technical safety concept and the system design specification." Tables 7 and 8 of 26262-6 recommend formal verification for the higher two ASIL levels and the use of formal notations for software unit design at all ASIL levels. The QuantUM approach uses a semi-formal notation and synthesizes a formal model of individual subsystems, which is a fit with the former two requirements. Table 10 specifies methods to be used in unit design and implementation verification, and requires formal verification for the higher two ASIL levels. Semantic code analysis, which can be accomplished by functional model checking, is recommended for all ASIL levels. We notice that the way in which QuantUM links model based design and formal analysis is consistent with the recommendations regarding software verification in this part of 26262, even though the QuantUM approach is restricted to analyzing quantitative safety requirements.

- Part 26262-7 addresses safety relevant issues in the *production, operation and maintenance* of automotive systems and is not directly relevant in our context.

- Part 26262-8 is devoted to specifying requirements on *processes supporting* the system development, including the management of safety requirements, the process of system verification, the qualification of software tools and the qualification of software components. In the specification of safety requirements, semi-formal and formal notations are again recommended for all ASIL levels. Verification is defined as a means to ensure that products meet their specification, which is interpreted as saying that they are "correct, complete and consistent." The process is required to include a planning of the verification activities, which also addresses the verification method to be used. Most notably, note 1 of clause 9.4.1.1 expressly mentions model checking as one such applicable verification method, even thought this remains the only reference to this verification technique throughout the standard. QuantUM supports the application of model checking to system design models, and hence fits into a verification process according to 26262-8.

  A further relevant issue addressed in 26262-8 is the qualification of software tools used in the development process. It requires that software tools used during system development will not entail the violation safety requirements allocated to some system aspect, or that if a violation of a safety requirement is caused, there is high confidence in revealing this later on in the development process. This entails a significant hurdle for the practical qualification of tools, since proving software tools correct is an

utterly resource consuming task that is not guaranteed to lead to a result at all, not at least since the underlying software correctness problem is undecidable. With respect to model checking tools it should be pointed out that the only sound usage of model checkers on models of realistic size, for which a complete state space exploration is likely to be impossible, is to use them as error detectors. They are suitable to increase the confidence in the correctness of some software artifact, but cannot prove it correctness unless the state space is so small that it can be completely explored. However, if we are able to produce abstractions of the system to be built that allow for a complete state space exploration, as it was the case when we used QuantUM and PRISM on significant case studies [13], then we may actually assume that the use of these tools is compliant with the software tool qualification requirements of 26262-8.

- Part 26262-9 finally addresses *safety-oriented analysis* techniques. Clause 8.2 mentions qualitative FMEA as well as qualitative FTA and Event Tree Analysis (ETA) as qualitative analysis techniques. As quantitative analysis techniques, quantitative FMEA, quantitative FTA, ETA and Markov models are being recommended, amongst others. Our QuantUM approach clearly supports the quantitative analysis, since it directly provides for quantitative FTA and enables probabilistic FMEA, as we illustrated in [1].

4 Conclusion

We have briefly sketched the QuantUM approach and discussed how it, along with other formal methods, can be used to support the software process requirements imposed by the standard ISO 26262. Our findings were that QuantUM is a good match for many of those requirements, even though the use of formal methods in support of this standard should not be extended to other semi-formal and formal techniques.

References

[1] H. Aljazzar, M. Fischer, L. Grunske, M. Kuntz, F. Leitner-Fischer, and S. Leue. Safety analysis of an airbag system using probabilistic FMEA and probabilistic counterexamples. In *QEST '09: Proceedings of the Sixth International Conference on Quantitative Evaluation of Systems*, pages 299–308, Los Alamitos, CA, USA, 2009. IEEE Computer Society.

[2] H. Aljazzar and S. Leue. Debugging of Dependability Models Using Interactive Visualization of Counterexamples. In *QEST '08: Proceedings of the Fifth International Conference on the Quantitative Evaluation of Systems*, pages 189–198. IEEE Computer Science Press, 2008.

[3] C. Baier and J.-P. Katoen. *Principles of Model Checking*. The MIT Press, 2008.

[4] D. Brown, H. Delseny, K. Hayhurst, and V. Wiels. Guidance for using formal methods in a certification context. In *Proc. Embedded*

Real Time Software and Systems (ERTS 2010), 2010. Available from http://www.erts2010.org/Default.aspx?Id=973&Idd=980.

[5] M. Broy. Seamless model driven systems engineering based on formal models. In K. Breitman and A. Cavalcanti, editors, *Formal Methods and Software Engineering*, volume 5885 of *Lecture Notes in Computer Science*, pages 1–19. Springer Verlag, 2009.

[6] M. Conrad and H. Dörr. Deployment of model-based software development in safety-related applications: Challenges and solutions scenarios. In *Proceedings of Modellierung 2006, 22.-24. März 2006, Innsbruck, Tirol, Austria*, pages 245–254, 2006.

[7] L. Grunske, R. Colvin, and K. Winter. Probabilistic model-checking support for FMEA. In *QEST '07: Proceedings of the Fourth International Conference on Quantitative Evaluation of Systems*, pages 119–128, Washington, DC, USA, 2007. IEEE Computer Society.

[8] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. PRISM: A tool for automatic verification of probabilistic systems. In *TACAS '06: Proceedings of the 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Lecture Notes in Computer Science, pages 441–444. Springer, 2006.

[9] IABG Information Technology. V-model lifecycle process model. Technical report, 1995. Available from http://www.v-modell.iabg.de/kurzb/vm/k_vm_e.doc.

[10] International Electrotechnical Commission. Analysis techniques for system reliability - procedure for failure mode and effects analysis (FMEA), IEC 60812.

[11] International Electrotechnical Commission. Functional safety of electrical/electronic/programmable electronic safety-related systems, IEC 61508, 2004.

[12] International Organization for Standardization. Road vehicles – functional safety, ISO 26262, 2008.

[13] F. Leitner-Fischer. Quantitative Safety Analysis of UML Models. Master's thesis, University of Konstanz, 2010.

[14] F. Leitner-Fischer and S. Leue. QuantUM: Quantitative safety analysis of UML models. In *Proc. Ninth Workshop on Quantitative Aspects of Programming Languages (QAPL 2011)*, 2011.

[15] Object Management Group. XML Metadata Interchange (XMI), v2.1.1. http://www.omg.org/technology/documents/formal/xmi.htm, 2007.

[16] Object Management Group. SysML. Specification v1.2. http://www.sysml.org, 2010.

[17] U.S. Nuclear Regulatory Commission. *Fault Tree Handbook*, 1981.