



CASE STUDY: MODEL-BASED DEVELOPMENT USING A CERTIFIED CODE GENERATOR WITHIN A SAFETY-CRITICAL (SIL-4) PROJECT

Thomas Liedtke¹, Paul Linder²

¹ ICS AG, Business Unit Methods, Processes & Tools

Address: Sonnenbergstraße 13, D-70184 Stuttgart, Germany

Phone: (+49) (0)711-2 10 37 39, Fax: (+49) (0)711-2 10 37 53, E-mail: thomas.liedtke@ics-ag.de

² ICS AG, Business Unit Methods, Processes & Tools

Address: Sonnenbergstraße 13, D-70184 Stuttgart, Germany

Phone: (+49) (0)711-2 10 37 52, Fax: (+49) (0)711-2 10 37 53, E-mail: paul.linder@ics-ag.de

Abstract: In this paper, we will present benefits of a model-based software development approach using automatic code generation (certified to EN 50128) evaluated in a real-life project in direct comparison to conventional software development techniques. The project was conducted in the area of transportation (railway control center) under SIL-4 requirements. We could proof that for module design, implementation and testing the effort can be decreased by half while increasing the quality of the whole software in terms of efficiency, effectivity and maintainability in parallel.

Keywords: model-based development, safety-critical software

1. INTRODUCTION

Software for railway control and protection systems must be in line with European Standard EN 50128. To ensure safety, the EN 50128 describes extensive requirements regarding the software development process, the methods and techniques to be applied, especially for safety-critical applications with high safety integrity level. Among others, the EN 50128 demands following software verification activities, which have to be performed in a thorough and comprehensive manner in order to achieve approval by the assessor:

1. Verification of software requirements, software architectural design and software module design
2. Static analysis of the source code against coding standards
3. Module test achieving full C0 coverage
4. Software integration test, software/hardware integration test and software requirements test achieving full coverage of the requirements and the architecture.

Our goal was to apply a modern model-based development approach to develop the software for a safety-critical railway protection system of the highest safety integrity level SIL-4. In this context, we had to face the question, how to integrate modern model-based development

techniques into an EN 50128 conforming development process in an effective manner.

The structure of the paper is as follows. In Chapter 2, the software project regarded is outlined. In Chapters 3 and 4, the approach to integrate the model-based development techniques into an EN 50128 conforming development process is presented. In Chapter 5, the benefits of using the model-based development approach are evaluated. Finally, Chapter 6 will present concluding remarks.

2. SCOPE OF THE CASE STUDY

Content of the software project regarded was the software development for a SIL-4 railway protection system. The system shall realize the safe transmission of interlocking signals and consists of two interconnected units, each of them consisting of several 2oo2 subsystems, each subsystem consisting again of two 16 bit micro-controllers.

The functionality of the software is to perform a safe duplex transmission of interlocking signals between the two units in conformance to the European Standard EN 50159-1 for safe signal transmission. The signal transmission includes the collection of the signals of the different 2oo2 subsystems on the sending unit and the

distribution of the signals between the different 2oo2 subsystems of the receiving unit. In this context, the functionality of the software may be summarized as following:

1. Protection of the signal transmission between the two units.
2. Protection of the signal transmission between the 2oo2 subsystems of each unit.
3. Communication between the two micro controllers of each 2oo2 subsystem.
4. Cyclic monitoring of the hardware of each 2oo2 subsystem including online-tests of CPU, RAM, ROM, power supply and temperature.

For the development of the software, the modelling, simulation and code generation tool SCADE (version 5.1) was chosen (Esteler SA, 2006). As modelling notation, SCADE uses synchronous, i.e. time discrete, data flow diagrams and state machines. The SCADE 5.1 code generation is certified against the EN 50128, i.e., the generated source code is regarded as equivalent to the corresponding source models. This implies that the generated source code do not have to be verified or tested against the source models.

3. SOFTWARE DEVELOPMENT

A major issue for the software development approach was the need for a hybrid development process regarding both, the model-based development of software with SCADE as well as the conventional development of software. The reason for this was the nonexistence of a framework or middleware for the micro controller used to easily integrated the generated source code. Hence, additional to the model-based development of the software application with SCADE, an elementary middleware had to be developed to realize a lightweight hardware abstraction layer containing the low-level functions needed to control the micro controller. This middleware, called "wrapper", had to be developed in a conventional manner using the programming language C. In consequence, a software development process was adopted following the V standard but splitting into two different branches after the software architectural design, namely into a model-based branch and a conventional branch. These branches joined up together again before software integration.

An important aspect of the development process adopted is the rigorous separation of model-based and conventionally developed soft-

ware parts into different modules with defined interfaces. Accordingly, the application modules (about 80% of the software functionality) were developed purely model-based with merely no manual coding, while the wrapper modules were developed conventionally in C without SCADE. This rigorous separation enabled a clear and stringent selection of development methods and techniques for every module, making it easy to check the compliance of the selection against the requirements of the EN 50128 and making the software development procedure as well as the software verification strategy comprehensible and transparent for Validation, Quality Management, RAMS Management and Assessment.

4. SOFTWARE VERIFICATION

The strict distinction made by the development process between the purely model-based development of application modules and the purely conventional development of wrapper modules enabled the effective usage of model-based testing techniques as essential part of the overall software verification strategy. While conventional testing techniques were used for the module test of the wrapper modules as well as for integration and system tests, a model-based testing approach was used for the module test of the application modules.

The model-based test of the application modules was performed in the simulation environment of SCADE while using the test coverage criterion MTC (Model Test Coverage) provided by SCADE to check the thoroughness of the testing. MTC is a data flow oriented coverage criterion that checks, which values are adopted by the ingoing and/or outgoing data flows of the single function blocks of the SCADE data flow diagrams during testing, e.g. if the output of a logical block adopts "true" and "false" during testing corresponding to a decision coverage of that block.

Essentially for the effectiveness of the used model-based testing approach was the fact that the SCADE code generator is certified for EN 50128. In consequence, the generated C source code needs not to be tested against the SCADE models. An interesting consequence of this is the fact that the model-based tests performed correspond actually to two conventional software verification activities, namely to the module test and to the module design verification against the architectural design. This issue has

impact on the specification of the model test cases, which proved to be more demanding as the specification of conventional module test cases, as the expected behavior must be derived from a rather abstract architectural design while, on the same time, the test cases must satisfy the thorough coverage criteria of module tests.

Additionally to the model-based testing, the SCADE models of the application modules were analyzed statically as compensation for the eliminated source code verification of the application modules due to the use of a certified code generator. The static analysis of the SCADE models comprised the computer-aided check of the data consistency of the models with SCADE and the manual review of the SCADE models against self-defined modeling rules. Analogous to coding standards, the modeling rules define, which modeling elements may be used and which elements must not be used, e.g. because they are not covered by the code generation certificate and will compromise the software verification strategy.

5. LESSONS LEARNED

Due to the software development process adopted with conventionally developed wrapper modules and model-based developed application modules, the benefits of a model-based development approach could be evaluated in direct comparison to the conventional development of the wrapper modules. Thereby, experiences as they were observed are described in the following subchapters. Lessons Learned were focused on effort and quality aspects mainly. All aspects evaluated applying our model-based approach as

described could be evaluated in a very positive way.

5.1 Less Effort and Better Quality for Detailed Design and Implementation

Taking into account the amount of functionality provided by the application modules versus the functionality of the wrapper modules, the effort to be spent to design, implement and test the application modules is much lower compared to the effort spent for the wrapper modules.

First, modeling and code generation of modules requires less effort than designing and coding modules. However to mind is the fact that modeling a module requires more effort than only designing a module.

Second, the model-based test of a module, resp. of its corresponding model, proved to require less effort than a conventional module test as well as for the execution of the test cases in the simulation as for the implementation of the test scripts. To test the models, no effort consuming implementation of test driver or test stubs are necessary. However, one point to mind is that the specification of model test cases proved to require more effort than the specification of conventional module test cases. This is, however, not surprising, as the model-based test corresponds to two conventional software verification activities, namely the module test and the verification of the module design versus the architectural design.

Figure 1 shows the distribution of effort spent related concerning two dimensions. Firstly: conventional (effort normalized to 100% (design and test)) vs. model-based development. Secondly: design vs. test.

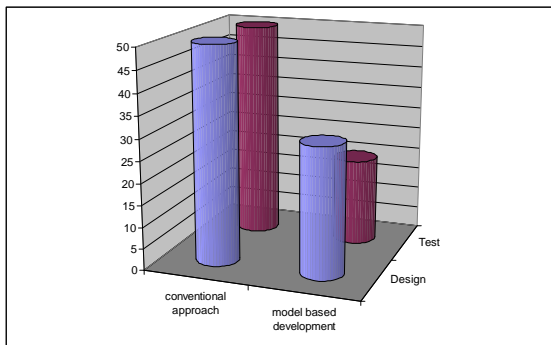


Fig. 1. Effort comparison model-based vs. conventional

If you have to spend typically 50% for design and coding and 50% for test in a conventional development approach, you can decrease the effort of design and coding by 40% and the effort for test by 60%. In total the effort can be decreased by 50% overall for the design, implementation and test of software modules. However to mind is the fact, that module development is only a single phase in the overall software development process.

5.2 Less Effort for Testing and Static Analysis

The static analysis of the model sources of the application modules within SCADE requires significant less effort compared to the verification of the source code of the wrapper modules. The crucial point allowing substituting the verification of the source code by static analysis of the models is the usage of a certified code generator to generate certified code out of the SCADE models.

5.3 Model-based Techniques as Components of Development and Verification Strategy

The approach adopted, to strictly distinguish between model-based developed modules and conventional developed modules, and to regard in consequence the model-based development and testing techniques used on module level just as clearly outlined components of the overall development and verification process, has proven in use. It led to a transparent and comprehensive development and verification strategy, which is vital for a SIL-4 project.

5.4 Quality Aspects

To enable judgment about the quality of model-based development, different aspects have to be considered:

1. *Effectivity*: During qualification test and field trials much less failures occurred. This leads to the assumption that fewer faults were introduced during the design phase (no coding faults due to code generator).

2. *Efficiency*: Due to the very low number of bugfixes (in fact only 1 until today) effort could be saved compared to conventional development

3. *Maintainability*: Due to the high transparency of requirements linked to the source models, readability and changeability in case of errors or changes could be increased significantly.

4. *Human motivation*: All team members feel much more comfortable to deal with models in stead of large code.

6. CONCLUDING REMARKS

In this paper, we presented experiences and lessons learned made regarding the application of model-based development and testing techniques for a safety-critical railway protection system of level SIL-4.

The model-based testing approach could be applied successfully for the model test of the application modules. The effort for design and test was much lower compared to the conventional module test for the hardware related wrapper modules programmed in C. This statement is taking into account the amount of functionality of these modules. The model test can be seen as a single but very important part of the software verification strategy applied with a clearly defined scope and role. Overall we could prove that approximately 50% of design, implementation and test of software modules can be saved comparing to a conventional software development approach increasing in parallel the quality of the software.

ACKNOWLEDGEMENT

The project has been supported by the Informatik Consulting Systems AG (ICS AG).

REFERENCES

- EN 50128 (2001). Railway applications – Communications, signalling and processing systems – Software for railway control and protection systems, *CENELEC*
- EN 50159-1 (2001). Railway applications – Communication, signalling and processing systems. Part 1: Safety-related communication in closed transmission systems, *CENELEC*
- Esterel SA (2006). SCADE User Manual. *Esterel Technologies SA*