Model-Based Design for IEC 61508: Towards Translation Validation of Generated Code

Mirko Conrad
The MathWorks, Inc.

Abstract: Production code generation with Model-Based Design has replaced manual coding in various vehicle subsystems and generated code is increasingly being deployed in safety-related applications as well. To validate the translation process and to ensure quality and functional safety, generated application software components shall be subjected to a combination of quality assurance measures. Further, compliance with safety standards needs to be demonstrated.
On principle, translation validation of generated code could be carried out in the same way as for hand code. However, from a process efficiency point of view, this would leave something to be desired. Therefore, engineering methods and tools to validate and verify generated code are highly desirable. As a step towards this goal, a workflow for verification and validation of models and generated code is outlined in this paper and mapped onto the objectives of IEC 61508-3.

1 Introduction

The increasing application of **embedded in-vehicle software** has resulted in a staggering complexity that has proven difficult to negotiate with conventional design approaches. Due to its capability to address the software complexity and productivity challenges, **Model-Based Design** [CFG+05, ADL07, MBD] is becoming the preferred software engineering paradigm for the development of application software components in various vehicle subsystems. The core idea is that an initial **executable graphical model** representing the application software component to be developed is refined and augmented until it becomes the blueprint for the final implementation through automatic **code generation**. Simulink products have become popular for Model-Based Design. Graphical modeling with time-based block diagrams and event-based state machines is supported by **Simulink**® [SL] and **Stateflow**® products; production code generation is supported by the **Real-Time Workshop® Embedded Coder™** [RTW-EC] product.

Regardless of the software development paradigm applied, it is crucial to subject the software being developed to an appropriate combination of **verification and validation** (V&V) activities to detect errors and produce confidence in the correct functioning of the embedded application software, i.e. to carry out a **translation validation** [PSS98]. V&V of software developed using Model-Based Design could be carried out in the same way as for hand code. However, the presence of executable graphical models opens up new, more efficient translation validation approaches. Models provide an excellent source of information that could be exploited for various V&V techniques. Considering the question of process efficiency, one should make the most of these possibilities. Therefore, engineering approaches for the translation validation of generated code tightly integrated with the construction activities of Model-based Design are highly desired.

In the recent past, Model-Based Design with code generation has been successfully employed to produce software for **safety-critical applications**, see e.g., [Pot04, TMW05, ADL07]. In such application contexts, additional requirements imposed by safety **standards** have to be met and the objectives and recommendations outlined therein must be mapped onto Model-

Based Design. This could become difficult because various safety standards including **IEC 61508** were assembled in the 1990s, i.e. before Model-Based Design was broadly adopted.

Contribution: This paper proposes a **workflow for verification and validation of models and generated code** that is intended to be used as an IEC 61508-3 compliant translation validation procedure. The workflow carries the advantages of Model-Based Design forward to safety-critical applications. Available tool support is also discussed.

2 A Workflow for Application-specific Verification and Validation of Models and Generated Code

IEC 61508-3 and other safety standards call for **application-specific verification and validation** regardless of the tool chain and the development paradigm used. When using Model-Based Design it seems natural to carry out a large share of the necessary V&V activities on model level. Unfortunately, more detailed guidance for the translation validation is rare: IEC 61508-3 dates back to 1998 and therefore does not have a notion of model testing and code generation. Its objectives need to be interpreted and mapped onto Model-Based Design [Con07, EC07]. A model-based approach to application-specific V&V is to divide the problem into two steps: First to demonstrate that the model is correct and meets all requirements, and then to show that the generated code is equivalent to the model [Ald01]. The first step, referred to as **design verification**, combines suitable V&V techniques on model level. The second step, **code verification**, uses equivalence testing and complementary techniques to demonstrate equivalence between the model and the generated code (Fig 1).

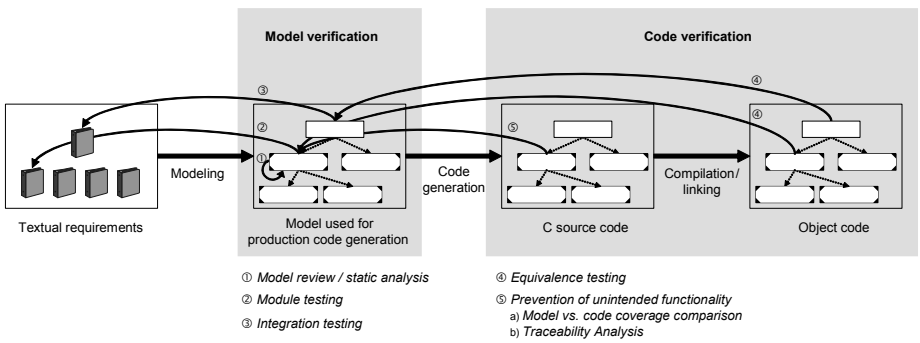

Fig. 1: Workflow for application-specific verification and validation

2.1 Verification and Validation on Model Level (Design Verification)

Design verification aims at gaining confidence in the model used as input for production code generation. It takes place before the actual code generation will be carried out. A **combination of reviews, static analyses and testing activities on model level** provides assurance that the design satisfies its associated requirements, i.e. it is verified. Requirements for design verification can be derived from IEC 61508-3 clauses 7.4.6 – 7.4.8.

Reviews and Static Analyses on Model Level: Model subsystems considered modules (model components) should be reviewed. Manual **model reviews** should be supported by automated **static analyses**. **Modeling guidelines** should be used and adherence with these guidelines should be assessed. Modeling constructs that are not suited or not recommended for production code generation should not be used. Model reviews are expected to provide an

efficiency gain over code reviews. [Kuf07] e.g. reports that model reviews could be about 5 times more efficient than source code reviews.

Tool support: The Simulink Block Support tables for Real-Time Workshop Embedded Coder identify blocks not suited for production code generation. The MAAB V2.0 guidelines [MAAB] contain general modeling guidelines for control algorithms. Adherence to guidelines and safety standards can be partially enforced by using modeling standard checks in Model Advisor [Beg07]. Model reviews can be facilitated by using reports or web views generated with Simulink Report Generator™ product.

Module and Integration Testing on Model Level: Model components should be functionally tested using systematically derived test vectors. These **module tests** should demonstrate that each module performs its intended function and does not perform unintended functions. As the module testing is completed then module **integration testing** should be performed to show that all model components interact correctly.

Tool support: The Simulink Verification and Validation product supports various facets of model testing. Model level test can be carried out within the Simulink environment.

2.2 Verification and Validation on Code Level (Code Verification)

By using **equivalence testing** between the model used for production code generation and the executable derived from the generated C code and augmenting measures to demonstrate the **absence of unintended functionality**, code verification demonstrates that the execution semantics of the model is preserved during code generation, compilation and linking. Successful code verification indicates that the outputs of Real-Time Workshop Embedded Coder and the compiler / linker tool-chain are correct for the design instance being checked. It's more difficult to derive requirements for the code verification part from IEC 61508-3, because such comparative testing is not addressed directly there. So, the standard's gist needs to be carried over.

Equivalence Testing: The core part of code verification is comparative testing between the model used for production code generation and the resulting object code. **Equivalence testing** (back-to-back testing) demonstrates numerical equivalence between the two artifacts. It's carried out by stimulating both the model used for code generation and the executable derived from the generated code with identical test vectors. Since complete testing is impossible, stimuli (test vectors) should sufficiently cover the different structural parts of the model. IEC 61508-3 doesn't call out specific coverage goals, but secondary sources such as [SS05] recommend that some test coverage metric should be visible at SIL 2 and above. Discussions of equivalence testing procedures can be found in [SC03, SC05, SCD+07]. Testing for numerical equivalence is unique in that the expected outputs for the test vectors do not have to be provided a-priori [Ald01]. This makes in-depth equivalence testing ideally suited to automation.

Tool support: The Simulink Design Verifier product helps create (additional) test vectors for structural testing. Model coverage information can be collected by using the Model Coverage Tool. Processor-in-the-loop (PIL) testing can be used to execute the object code on a target-like hardware.

Prevention of Unintended Functionality: Multiple techniques are available to demonstrate that the generated C code does not perform any **unintended function.** These include:

Model and code coverage are measured during equivalence testing and compared against each other **(model vs. code coverage comparison**). Discrepancies w.r.t. comparable

coverage metrics should be assessed. If the code coverage achieved is less than the model coverage, unintended functionality could have been introduced.

A **traceability analysis** of the generated code can be performed to ensure that all parts of the generated source code can be traced back to the model.

Tool support: Code coverage information can be derived from the IDE or by integrating a code coverage tool. Real-Time Workshop Embedded Coder provides a traceability report with mapping information between model blocks and the generated code.

3 Related work

Earlier work on V&V of production code generation includes [Edw99, Bur04, SWC05]. Typically a combination of translation validation and translator validation approaches is discussed. However, none of the papers discusses the proposed measures or techniques in the context of IEC 61508. Translator validation can be considered orthogonal to translation validation discussed in this paper. A popular means of translator validation is validation testing. MISRA-C:2004 [MISRA-C] suggests documented validation testing as a possible approach to gain confidence in the tools used. This approach has been applied to Model-Based Design by major automotive companies and the TÜV. A production code generation tool chain including Embedded Coder successfully passed the resulting Automotive Code Validation Suite (AVS), a test suite to validate quality and robustness of the code generator, compiler, linker tool chain [TÜV05; PSB07]. Underlying concepts on such test suites are laid out for example in [SC03, Stü06, ZPC+06, SCD+07]. The interplay between model and code coverage has been examined analytically by [Ald01] and statistically by [BCS+03].

4 Conclusions

The increasing usage of Model-Based Design with production code generation calls for new methods and tools for verifying and validating generated code. Model-Based Design allows for new V&V approaches utilizing the existence of executable graphical models as well as multiple executable artifacts which could be checked for equivalence. For safety-critical developments existing V&V approaches need to be reconciled with safety standards such as IEC 61508. The proposed workflow for verification and validation of models and generated code could be viewed as instantiation of the V&V requirements of IEC 61508-3, which exploits the possibilities of code generation as an integral part of Model-Based Design. Users of Simulink and Real-Time Workshop Embedded Coder can use this workflow as a reference to create their own project and SIL specific variants of this workflow. The presented workflow targets typical safety-critical in-vehicle applications, i.e. SIL 2 or SIL 3.

While adhering to the recommendations in this document will reduce the risk that an error is introduced in the development and not be detected, it is not a guarantee that the system being developed will be safe. Conversely, if some of the recommendations in this document are not followed, it does not mean that the system being developed will be unsafe.

References

[ADL07] Kosten-/Nutzenanalyse der modellbasierten Softwareentwicklung im Automobil. Arthur D. Little, 2007.

[Ald01] Aldrich, W.: Coverage Analysis for Model-Based Design Tools. TCS 2001.

[BCS+03] Baresel, A.; Conrad, M.; Sadeghipour, S.; Wegener, J.: The Interplay between Model Coverage and Code Coverage. EuroSTAR ‘03, Amsterdam, Netherlands.

[Beg07] Begic, G.: Checking Modeling Standards Implementation. The MathWorks News & Notes, June 2007.

[Bur04] Burnard, A.: Verifying and Validating Automatically Generated Code. IAC ’04, Stuttgart, Germany.

[CFG+05] Conrad, M.; Fey, I.; Grochtmann, M.; Klein, T.: Modellbasierte Entwicklung eingebetteter Fahrzeugsoftware bei DaimlerChrysler. Inform. Forsch. Entwickl. 20(1-2): 3-10 (2005).

[Con07] Conrad, M.: Using Simulink and Real-Time Workshop Embedded Coder for Safety-Critical Automotive Applications. MBEES '07, Dagstuhl, Germany.

[EC07] Erkkinen, T.; Conrad, M.: Safety-Critical Software Development Using Automatic Production Code Generation. SAE World Congress 2007, Detroit, USA.

[Edw99] Edwards, P.D.: The Use of Automatic Code Generation Tools in the Development of Safety-Related Embedded Systems. ERA Report 99-0484, 1999.

[IEC61508-3] IEC 61508-3:1998. Int. Standard Functional safety of electrical/ electronic/ programmable electronic safety-related systems - Part 3: Software requirements. 1998.

[Kuf07] Kuffner, W.: Umsetzung der Anforderungen aus der funktionalen Sicherheit in der Fahrwerkselektronik. 11. Int. Fachkongress Fortschritte in der Automobil-Elektronik 2007; Ludwigsburg, Germany

[MAAB] Control Algorithm Modeling Guidelines Using MATLAB®, Simulink®, and Stateflow® V2.0, 2007. www.mathworks.com/industries/auto/maab.html

[MBD] Model-Based Design: www.mathworks.com/applications/controldesign/description

[MISRA-C] MISRA-C:2004. Guidelines for the use of the C language in critical systems. MIRA, 2004.

[Pot04] Potter, B.: Use of The MathWorks Tool Suite to develop DO-178B certified code. ERAU/FAA Software Tools Forum 2004, Daytona Beach, USA.

[PSB7] Pofahl, E., Sauer, T., Busa, O.: AVS - A Test Suite for Automatically Generated Code. MAC ‘07, Dearborn, USA.

[PSS98] Pnueli, A.; Siegel, M.; Singerman, E.: Translation Validation. TACAS'98, Lisbon, Portugal.

[RTW-EC] Real-Time Workshop® Embedded Coder™: www.mathworks.com/products/rtwembedded

[SC03] Stürmer, I.; Conrad, M.: Test Suite Design for Code Generation Tools. ASE '03, Montreal, Canada.

[SC05] Stürmer, I.; Conrad, M.: Ein Testverfahren für optimierende Codegeneratoren. Inform. Forsch. Entwickl. 19(4): 213-223 (2005)

[SCD+07] Stürmer, I.; Conrad, M.; Dörr, H.; Pepper, P.: Systematic Testing of Model-Based Code Generators. IEEE Transactions on Software Engineering, Sep 2007, pp. 622-634

[SL/SF] Simulink®, Stateflow®: www.mathworks.com/products/simulink

[SS05] Smith, D.J.; Simpson, K.G.L.: Functional Safety – A straightforward guide to applying IEC 61508 and related standards. Elsevier Butterworth-Heinemann, 2005

[Stü06] Stürmer, I.: Systematic Testing of Code Generation Tools - A Test suite-oriented Approach for Safeguarding Automatic Code Generation. TU Berlin, Germany, 2006.

[SWC05] Stürmer, I.; Weinberg, D.; Conrad, M.: Overview of Existing Safeguarding Techniques for Automatically Generated Code. SEAS '05, St. Louis, USA.

[TMW05] The MathWorks: Alstom generates production code for safety-critical power converter control systems. 2005. www.mathworks.com/products/rtwembedded/userstories.html?file=10591

[TÜV05] Validation of the MathWorks code generator Real-Time Workshop® Embedded Coder with the Autocode Validation Suite v3.0. Report No. 968/EL 211.02/05, TÜV Rheinland, 2005.

[ZPC+06] Zelenov, S.V.; Petrenko, A.K.; Conrad, M.; Fey, I.: Automatic Test Generation for Model-Based Code Generators. ISoLA ’06, Paphos, Cyprus.