Verification and Validation According to ISO 26262: A Workflow to Facilitate the Development of High-Integrity Software

Mirko Conrad

The MathWorks, Inc.
Natick, MA, USA
mirko.conrad@mathworks.com

ABSTRACT

Model-Based Design with production code generation has been extensively utilized throughout the automotive software engineering community because of its ability to address complexity, productivity, and quality challenges. With new applications such as lane departure warning or electromechanical steering, engineers have begun to consider Model-Based Design to develop embedded software for applications that need to comply with functional safety standards such as ISO 26262.

For the development of high-integrity in-vehicle software, ISO 26262 is considered state-of-the-art or generally accepted rules of technology (GART) [Fal02, Lov06]. Developers of in-vehicle software need to understand and implement the standard’s requirements pertaining to software development. Due to the widespread utilization of Model-Based Design to develop automotive E/E systems, it is of particular importance to set-up ISO 26262 compliant Model-Based Design processes and tool chains.

This paper discusses a verification and validation workflow for developing in-vehicle software components which need to comply with ISO 26262-6 using Model-Based Design. It discusses tool support by using a Simulink® family tool chain for Model-Based Design as an example.

KEYWORDS: Functional Safety, ISO 26262, Model-Based Design, Reference Workflow, Verification and Validation of Models and Generated Code, Simulink, Embedded Coder, IEC Certification Kit

INTRODUCTION

The increasing application of embedded software in passenger cars has resulted in a staggering complexity that has proven difficult to negotiate with conventional design approaches and processes. In addition, more and more projects need to comply with functional safety standards, because modern software-based electronic control units (ECUs) control or interact with critical systems such as brakes and steering.

Development approaches and processes play a decisive role in addressing the complexity, productivity, and quality challenges. Because of its ability to address these challenges, Model-Based Design has been extensively used throughout the automotive software engineering community. More recently, automotive OEMs and suppliers have begun to consider and adopt Model-Based Design for the development of embedded software for high-integrity applications that need to meet requirements of the ISO 26262 standard.

ISO 26262 “Road Vehicles - Functional Safety” [ISO 26262] is the adaptation of IEC 61508 [IEC 61508] to comply with needs specific to the application sector of electric / electronic systems (E/E systems) within road vehicles. This adaptation applies to all activities during the safety lifecycle of systems composed of electrical, electronic, and software elements that provide safety-related functions. ISO 26262-6 “Product development: software level” [ISO 26262-6] is concerned with software development, verification, and validation.

Model-Based Design tools such as Simulink and Stateflow® [Simulink] can be used throughout multiple phases of the software life cycle. Model-Based Design enables early simulation and testing of the functional behavior as well as automatic code generation for production systems. Production code generation within Model-Based Design has replaced manual coding in various vehicle domains and generated code is increasingly being deployed in high-integrity applications [JSB+08, FMC08].

The use of Model-Based Design for high-integrity applications requires extra consideration and rigor because additional requirements imposed by functional safety standards such as ISO 26262 have to be met. To improve quality and functional safety, models and generated code should be subjected to a combination of quality assurance measures. Further, compliance with ISO 26262 needs to be demonstrated.

This paper discusses how Model-Based Design and associated verification and validation techniques can be leveraged to help develop automotive applications that comply with ISO 26262. It covers the mapping of selected ISO 26262-6 objectives onto Model-Based Design.

A WORKFLOW FOR VERIFICATION AND VALIDATION OF MODELS AND GENERATED CODE

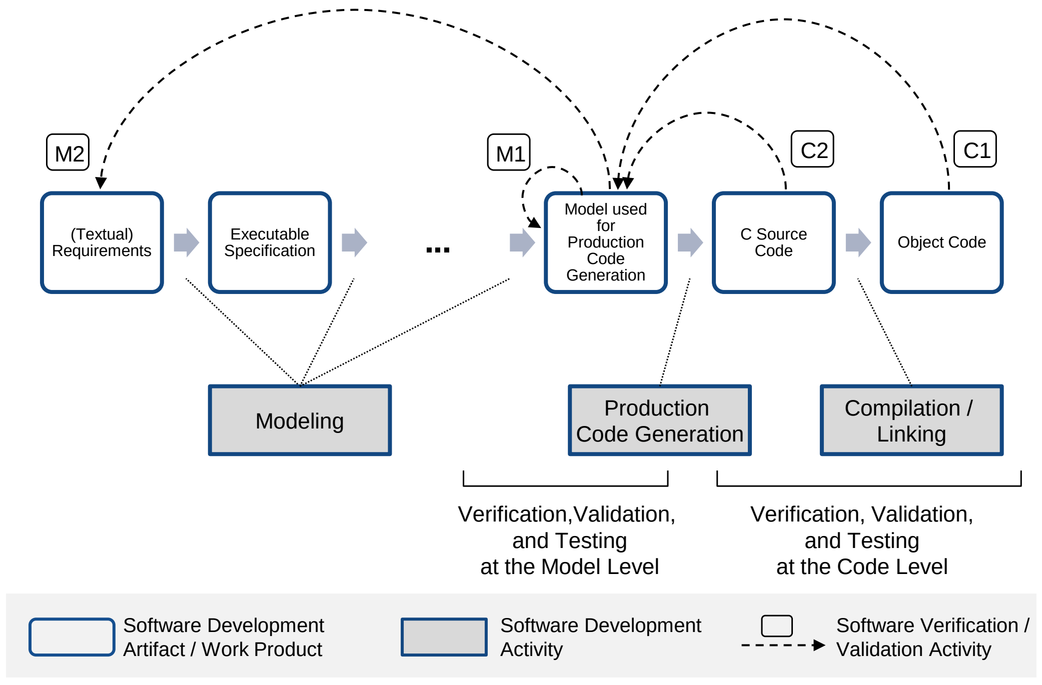
ISO 26262 calls for application-specific verification and validation regardless of the tool chain and the development process used. When using Model-Based Design with production code generation, application-specific verification and validation needs to answer the following questions:

- Does the model implement its (textual) requirements?
- Does the object code to be deployed in the ECU implement the design embodied in the model?

To facilitate a seamless use of Model-Based Design in the context of ISO 26262, a reference workflow that describes the verification and validation activities necessary for ISO 26262-6 compliance [Cert Kit] has been developed by MathWorks and audited by the certification authority TÜV SÜD. The ISO 26262 workflow is derived from a similar reference workflow for the generic IEC 61508 standard [Con08, Con09]. Following the suggested division of the verification and validation activities for applications developed using Model-Based Design and production code generation, the workflow can be divided into the following steps (Fig 1):

1. Design Verification: Demonstrate that the model satisfies its specification and does not contain unintended functionality.
2. Code Verification: Show that the generated code matches the model and does not contain unintended functionality.

Utilizing similar workflows for IEC 61508 and ISO 26262 streamlines the software development activities of companies that need to comply with different standards for different projects.



M2
M1
C2
C1
(Textual) Requirements
Executable Specification
...
Model used for Production Code Generation
C Source Code
Object Code
Modeling
Production Code Generation
Compilation / Linking
Verification,Validation, and Testing at the Model Level
Verification, Validation, and Testing at the Code Level
Software Development Artifact / Work Product
Software Development Activity
Software Verification / Validation Activity

M1	Reviews and static analyses at the model level
M2	Module and integration testing at the model level
C1	Back-to-back comparison testing between model and code
C2	Measures to prevent unintended functionality

Fig. 1: Reference workflow for application-specific verification and validation of models and generated code

DESIGN VERIFICATION — VERIFICATION, VALIDATION, AND TESTING AT THE MODEL LEVEL

The goal of design verification is to gain confidence in the model that is being used for production code generation. Design verification takes place at the model level, before the code is generated.

The design verification part comprises a combination of reviews, static analyses, and comprehensive functional testing activities at the model level [Con08, Con09]. These activities together help provide confidence that the design satisfies the associated requirements, does not contain unintended functionality and complies with defined modeling guidelines.

[M1] Reviews and Static Analyses at the Model Level

Model components, i.e. model subsystems considered as modules, should be reviewed. If feasible, these manual model reviews should be supported by automated static analyses of the model [Beg07].

Modeling guidelines should be used and adherence with the guidelines should be assessed. Modeling constructs that are not suited or not recommended for production code generation should not be used. In addition, modeling guidelines addressing specific ISO 26262 objectives should be employed.

[M2] Module and Integration Testing at the Model Level

Model components should be functionally tested using test cases systematically derived from the software requirements specification. The objective of module testing is to demonstrate that each model component performs its intended function and does not perform unintended functions.

As model component testing is completed, components can be integrated. The model integration stages shall be tested in accordance with the dedicated integration tests. These tests should show that all model components interact as specified to perform their intended function and do not perform unintended functions.

A more detailed discussion on testing at the model level can be found in [CF09].

The result of the design verification process is a golden model, i.e., a validated and verified, well-formed model that implements the requirements and does not contain unintended functionality. The golden model serves as a reference for the subsequent code verification activities.

CODE VERIFICATION — VERIFICATION, VALIDATION, AND TESTING AT THE CODE LEVEL

A combination of verification, validation, and testing activities at the code level shall demonstrate that the semantics of the golden model is being preserved during code generation, compilation, and linking. Furthermore, it needs to be shown that no unintended functionality is being introduced during code generation.

[C1] Back-to-back Comparison Testing Between Model and Code

The reference workflow utilizes translation testing [Con09, Con11], i.e. translation validation using systematic testing, to demonstrate numerical equivalence between the model and the generated C code. Translation testing in this context means that each model-to-code-translation (i.e. a run of the code generation tool chain) is augmented with a back-to-back testing phase (cf. ISO 26262-6, sections 9 and 10) to verify that the executable compiled from the generated C code (object code) matches the golden model.

Back-to-back testing for numerical equivalence is well suited to automation, because the expected outputs for the test vectors do not have to be provided [Ald01]. Detailed discussions of back-to-back testing procedures are available in [SM05, SCD+07].

[C2] Measures to Prevent Unintended Functionality

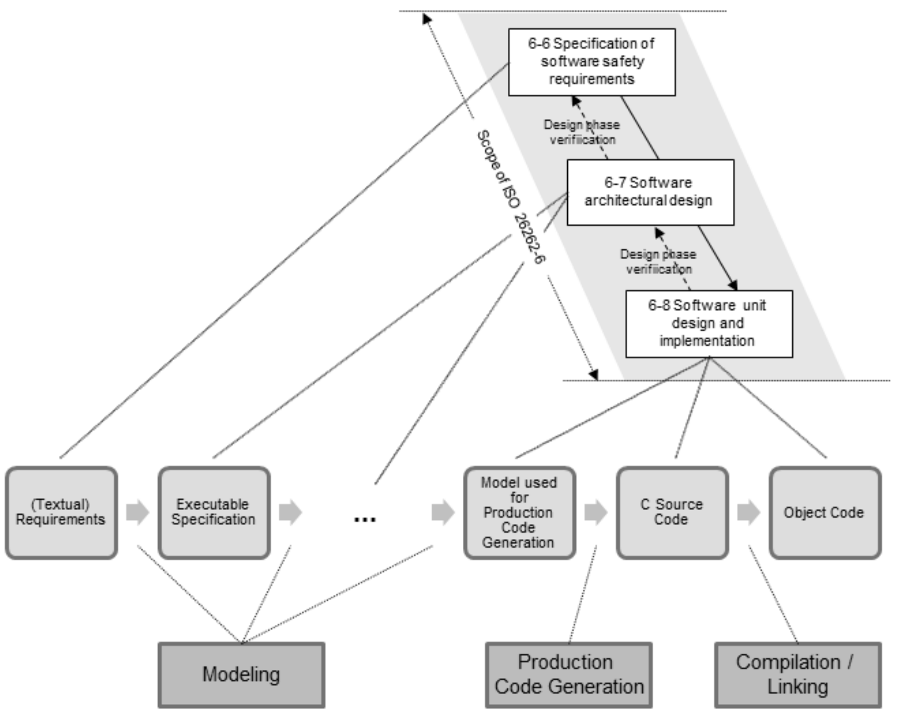
Back-to-back testing alone may not be sufficient to demonstrate the absence of unintended functionality in the generated source code. Therefore, back-to-back testing to assess numerical equivalence needs to be augmented with additional measures to demonstrate the absence of unintended functionality. This second activity in the code verification process shall demonstrate that the generated C code does not perform unintended functions, i.e., the code does not contain functionality not embodied in the model.

Alternative techniques are available to achieve this objective including (a) model versus code coverage comparison and (b) traceability analysis. These techniques serve the purpose of demonstrating structural equivalence between the model and the source code. If model versus code coverage comparison is being used to show the absence of unintended functionality, model and code coverage are measured during back-to-back testing and compared against each other. Discrepancies with respect to comparable coverage metrics need to be assessed. Alternatively, a traceability analysis can be performed to demonstrate that all parts of the generated C source code can be traced back to the golden model. In this case the generated code is subjected to a limited review that exclusively focuses on traceability aspects. Non-traceable code shall be flagged and assessed. Generation of a traceability matrix to aid traceability analysis is provided in the IEC Certification Kit [Cert Kit].

MAPPING ONTO ISO 26262-6 AND COMPLIANCE DEMONSTRATION

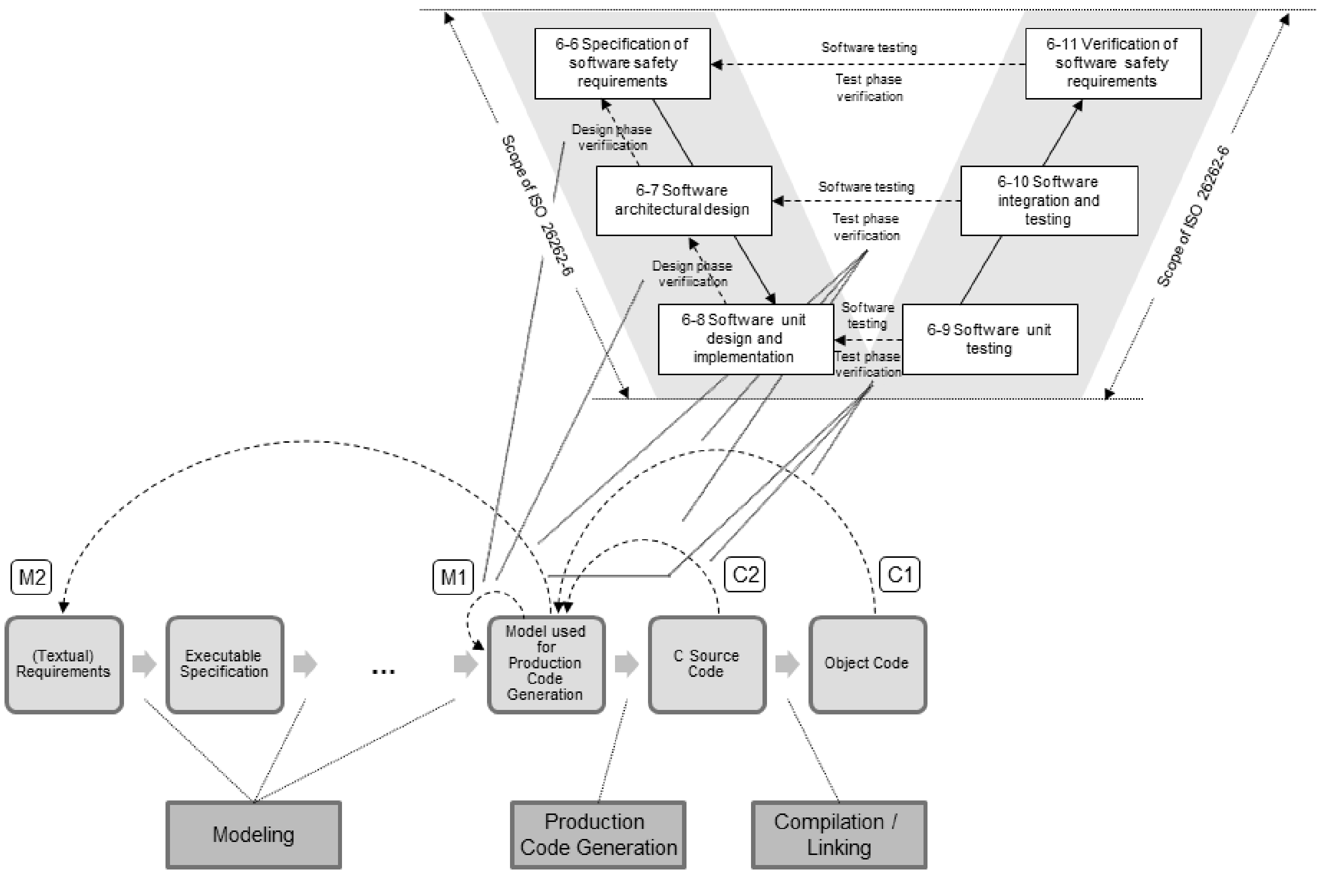
The design and code verification activities described in the reference workflow provide core activities to implement requirements of ISO 26262-6 when utilizing executable models and generated code as part of the software life cycle. These activities need to be augmented with additional measures to address other objectives of the ISO 26262 standard.

Fig. 2 and 3 map the reference workflow and the corresponding activities in ISO 26262-6 onto each other. Fig. 2 provides a mapping between the ISO 26262-6 design phase activities and the software development artifacts from the reference workflow. Fig. 3 maps design phase verification and test phase verification activities of the standard onto their counterparts in the reference workflow.



6-6 Specification of software safety requirements
Design phase verification
6-7 Software architectural design
Design phase verification
6-8 Software unit design and implementation
Scope of ISO 26262-6
(Textual) Requirements
Executable Specification
...
Model used for Production Code Generation
C Source Code
Object Code
Modeling
Production Code Generation
Compilation / Linking

Fig. 2: Mapping of software development artifacts and activities



6-6 Specification of software safety requirements
Software testing
Test phase verification
6-11 Verification of software safety requirements
Design phase verification
6-7 Software architectural design
Software testing
Test phase verification
6-10 Software integration and testing
Design phase verification
6-8 Software unit design and implementation
Software testing
Test phase verification
6-9 Software unit testing
Scope of ISO 26262-6
Scope of ISO 26262-6
M2
M1
C2
C1
(Textual) Requirements
Executable Specification
...
Model used for Production Code Generation
C Source Code
Object Code
Modeling
Production Code Generation
Compilation / Linking

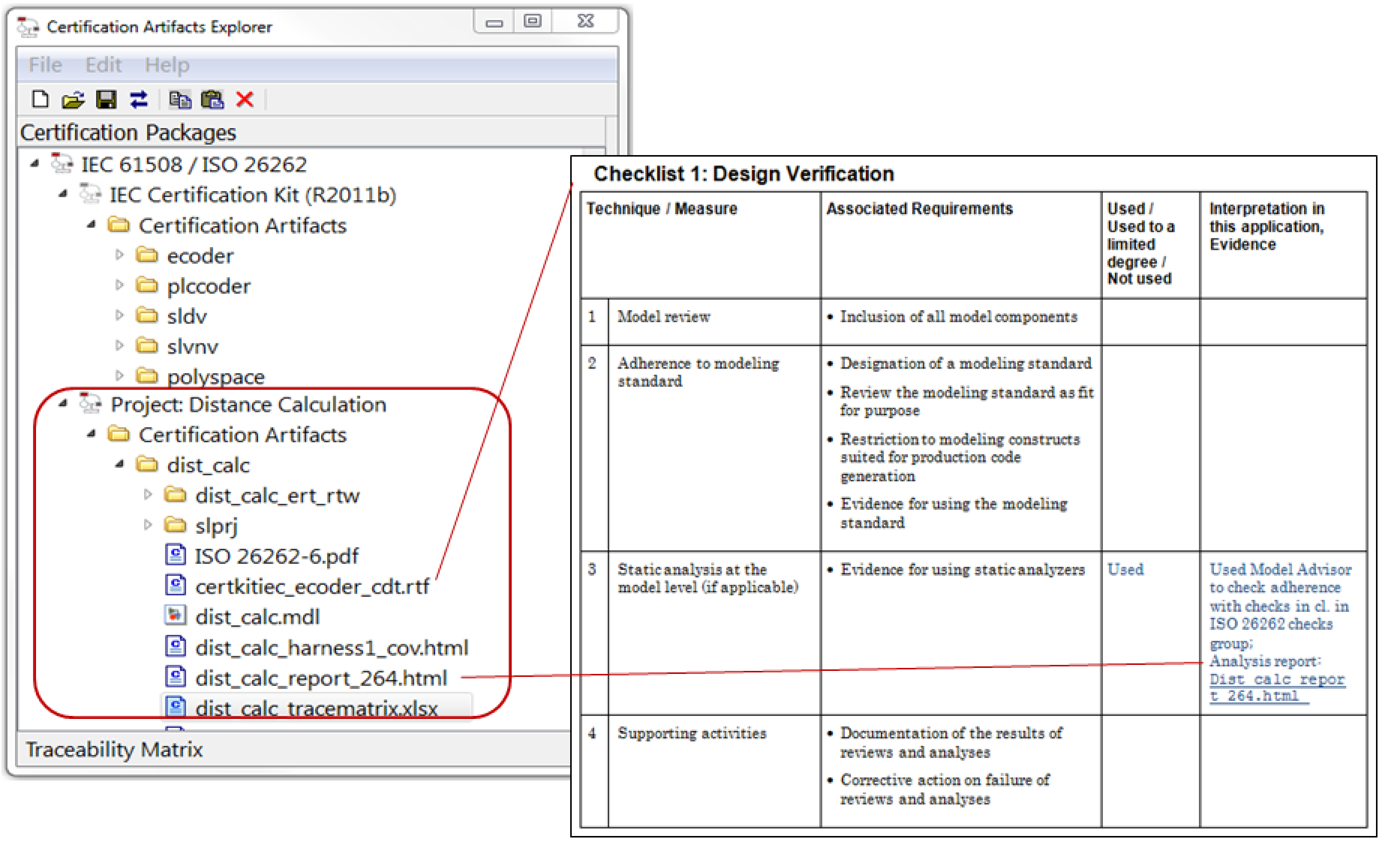
Fig. 3: Mapping of software verification, validation, and testing activities

Examples for carrying out the above mentioned verification, validation and testing activities can be found in [Con11].

When using the reference workflow, compliance demonstration with the applicable requirements of ISO 26262-6 can be facilitated by

- Checklists that document the realization of the workflow activities and reference the work products created and
- Artifact management systems supporting the compilation of items for the safety case and traceability between them.

Fig. 4 illustrates the checklist-based compliance demonstration process and the management of associated artifacts with the Artifacts Explorer [CertKit].



- IEC 61508 / ISO 26262
  - IEC Certification Kit (R2011b)
    - Certification Artifacts
      - ecoder
      - plccoder
      - sldv
      - slvnv
      - polyspace
  - Project: Distance Calculation
    - Certification Artifacts
      - dist_calc
        - dist_calc_ert_rtw
        - slprj
        - ISO 26262-6.pdf
        - certkitiec_ecoder_cdt.rtf
        - dist_calc.mdl
        - dist_calc_harness1_cov.html
        - dist_calc_report_264.html
        - dist_calc_tracematrix.xlsx

Checklist 1: Design Verification

	Technique / Measure	Associated Requirements	Used / Used to a limited degree / Not used	Interpretation in this application, Evidence
1	Model review	• Inclusion of all model components		
2	Adherence to modeling standard	• Designation of a modeling standard • Review the modeling standard as fit for purpose • Restriction to modeling constructs suited for production code generation • Evidence for using the modeling standard		
3	Static analysis at the model level (if applicable)	• Evidence for using static analyzers	Used	Used Model Advisor to check adherence with checks in cl. in ISO 26262 checks group; Analysis report: Dist_calc_report_264.html
4	Supporting activities	• Documentation of the results of reviews and analyses • Corrective action on failure of reviews and analyses		

Fig. 4: Artifact management tool and checklist-based compliance demonstration (example)

SUMMARY AND CONCLUSION

ISO 26262-6 is the main functional safety standard with respect to software development for high-integrity in-vehicle applications. It defines requirements and constraints for software development, verification and validation processes. These requirements apply to both Model-Based Design with code generation and traditional software development. However, implementing these requirements within Model-Based Design requires special consideration. To aid this process, the author discussed a reference workflow for Model-Based Design and tool considerations to address the objectives of ISO 26262-6.

The proposed workflow for verification and validation of models and generated code can be viewed as instantiation of the verification and validation requirements outlined in ISO 26262-6. This instantiation exploits the advantages of production code generation as an integral part of Model-Based Design.

The reference workflow can also help to meet the tool qualification requirements of ISO 26262-8 [ISO 26262-8, section 11]. In particular, it can provide the foundation for a pre-qualification of the Model-Based Design tools used in the implementation of this workflow [CSM11, HKW+11]. Such a procedure was used for the in-context certification by TÜV SÜD of the Embedded Coder code generator and other tools for Model-Based Design.

A detailed workflow description, together with a compliance demonstration template, and templates for the ISO 26262-8 tool qualification work products are made available as part of the IEC Certification Kit [Cert Kit]. The kit also contains evidence of the independent assessment carried out by TÜV SÜD (certificates and corresponding certification reports), the artifacts explorer, and the traceability matrix generation tool.

REFERENCES

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

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

[Cert Kit] IEC Certification Kit (for ISO 26262 and IEC 61508). www.mathworks.com/products/iec-61508.

[CF09] Conrad, M., Fey, I.: Testing Automotive Control Software. In: Navet, N., Simonot-Lion, F. (Eds): Automotive Embedded Systems Handbook, CRC Press, 2009.

[Con08] Conrad, M.: Model-Based Design for IEC 61508: Towards Translation Validation of Generated Code. Proc. Workshop Automotive Software Engineering: Forschung, Lehre, Industrielle Praxis, co-located with Software Engineering 2008, Munich, February 2008.

[Con09] Conrad, M.: Testing-based translation validation of generated code in the context of IEC 61508. Form. Methods Syst. Des. 35, 3 (Dec. 2009), 389-401

[Con11] Conrad, M.: Testing-Based Translation Validation of Generated Code. In: Zander, J., Schieferdecker, I., Mostermann, P. J. (Eds): Model-Based Testing for Embedded Systems (Computational Analysis, Synthesis, and Design of Dynamic Systems), CRC Press, 2011

[CSM11] Conrad, M., Sandmann, G., Munier, P.: Software Tool Qualification According to ISO 26262. SAE 2011 World Congress, Detroit, MI, USA, April 2011. SAE Techn. Paper #2011-01-1005

[Fal02] Faller, R.: The Evolution of European Safety Standards. exida.com, 2002.

[FMC08] Fey, I., Müller, J., Conrad, M.: Model-Based Design for Safety-Related Applications. Proc. Convergence 2008, Detroit, MI, USA, Oct. 2008

[IEC 61508] IEC 61508 ‘Functional Safety of Electrical/Electronic/ Programmable Electronic Safety-Related Systems’. International Standard, 2nd edition, 2010.

[ISO 26262] ISO 26262 ‘Road vehicles - Functional safety’. International Standard, 2011.

[ISO 26262-6] ISO 26262 ‘Road vehicles - Functional safety - Part 6: Product development: software level’. International Standard, 2011.

[ISO 26262-8] ISO 26262 ‘Road vehicles - Functional safety - Part 8: Supporting processes’. International Standard, 2011.

[JSB+08] Jablonski, T., Schumann, H., Busse, C., Haussmann, H., Hallmann, U., Dreyer, D., Schöttler, F.: Die neue elektromechanische Lenkung APA-BS. ATZelektronik 01/2008 Vol. 3 (2008) 01, pp. 30–35.

[Lov06] Lovric, T.: Sicherheitsanforderungen an Fahrerassistenzsysteme. 2. Sachverständigentag (SVT 2006), Sept. 2006.

[Simulink] Simulink®. www.mathworks.com/products/simulink

[SM05] 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.

[HKW+11] Hamann, R., Kriso, S., Williams, K., Klarmann, J., Sauler, J.: ISO 26262 Release Just Ahead: Remaining Problems and Proposals for Solutions. Proc. SAE World Congress 2011, Detroit, MI, USA, April 2011.

CONTACT

Dr. Mirko Conrad
Development Manager - Simulink Certification and Standards; The MathWorks, Inc. Mirko.Conrad@mathworks.com

Mirko Conrad manages MathWorks certification and standards team. He oversaw the certification of Embedded Coder, Simulink Verification and Validation, and Simulink Design Verifier for usage in development processes that need to comply with IEC 61508, ISO 26262 related standards. His team develops and maintains the IEC Certification Kit.

Mirko started his professional career in the automotive industry in 1995 and joined The MathWorks in 2006. He holds a Ph.D. in engineering (Dr.-Ing.) and a M.Sc. in computer studies (Dipl.-Inform.) from Technical University Berlin, Germany. Mirko is also a visiting lecturer at Humboldt University in Berlin and Technical University Munich. His publication record includes more than 80 papers on automotive software engineering, functional safety, and Model-Based Design. He is a member of the managing team of the Special Interest Group for Automotive Software Engineering in the German Computer Society (GI-ASE) and was a member of the ISO 26262 standardization sub-working group on software.