Artifact-Centric Compliance Demonstration for ISO 26262 Projects Using Model-Based Design

Mirko Conrad

The MathWorks GmbH
mirko.conrad@mathworks.de

Abstract: Automotive software components are frequently engineered using Model-Based Design. For software that needs to comply with the ISO 26262 standard, OEMs and suppliers look for efficient ways to demonstrate compliance with the software-related requirements of this functional safety standard. To demonstrate process compliance, the objectives and requirements of ISO 26262-6 need to be mapped onto Model-Based Design approaches and tools. Since Model-Based Design projects typically use a tailored version of the ISO 26262 reference phase model for software development, this is not a straightforward task.

This paper discusses an artifact-centric compliance demonstration approach intended to streamline ISO 26262 compliance documentation for software developed using Model-Based Design with code generation. Using templates and distinguishing between application-specific and application-agnostic compliance arguments, the approach facilitates partial re-use of the compliance documentation artifacts across Model-Based Design projects using the same or updated tools and processes.

1 Model-Based Design for Automotive Software Components

The increasing utilization of embedded software has resulted in a staggering complexity that has proven difficult to manage with conventional design approaches. Because of its capability to address the corresponding software complexity and productivity challenges, **Model-Based Design** [HKM05, CFG05, CD06, MBD] is quickly becoming the preferred software engineering paradigm for the development of embedded software components across application domains. The **Simulink®** modeling environment [SL, SF] supports different technologies that comprise Model-Based Design.

With the advent of Model-Based Design, an artifact of great value along many axes of use has been introduced into the software life cycle between requirements and code: the **model**. In a typical Model-Based Design workflow, an initial executable graphical model represents the software component under development by including pertinent design behavior while omitting the detailed complexity of the software implementation. The model is then refined and augmented **(model elaboration)** till it is sufficiently detailed to serve as the blueprint for the final implementation through **automatic code generation** [CFG+05]. This way, application software development is characterized by successive refinement of models followed by code generation, compilation and linking as shown in Fig. 1.

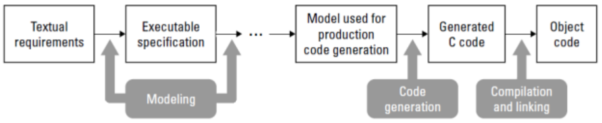

Fig. 1: Model-Based Design workflow

The modeling notations offered by Simulink and Stateflow® (i.e. block-diagrams, state-transition diagrams, flow charts, truth tables, imperative code written in MATLAB®) are used throughout these consecutive modeling stages. Different verification and validation activities can be carried out at the model level; i.e., early in the software lifecycle before source code becomes available.

This development paradigm exhibits certain specifics; e.g., the existence of new executable software engineering artifacts (i.e. models) and the partial blending of different lifecycle phases.

2 Model-Based Design and ISO 26262

ISO 26262 'Road vehicles - Functional safety' [ISO 26262], the functional safety standard for the automotive domain, was published in 2011. ISO 26262-6 discusses 80+ specific methods intended to increase the level of confidence in achieving compliance with the software related requirements and recommendation of the standard. These methods are presented in 16 'methods tables', where they are mapped to the four automotive safety integrity levels, ASIL A to ASIL D.

ISO 26262-6 explicitly discusses requirements and recommendations related to Model-Based Design and code generation. Appendix B outlines the concept of model-based development of in-vehicle software and its implications on product development at the software level. However, the guidance is scattered across ISO 26262-6 and needs to be combined and glued together. As an example, it remains open how to map the requirements and recommendations onto new software development artifacts (namely executable models) and how to deal with the partial blending of different lifecycle phases in Model-Based Design.

As a result - prior to applying the standard in a Model-based Design project - ISO 26262 needs tailored towards Model-Based Design, i.e. to be mapped onto the specific development artifacts, processes, and tools used during modeling, code generation and related verification activities.

To comply with ISO 26262-6, the applicant needs to demonstrate that each applicable requirement is satisfied or to provide a rationale that the non-compliance is acceptable. To facilitate this demonstration, applicants usually compile **compliance documentation**.

Even for traditional software development processes this is not a straightforward activity. Efficiency and effectiveness of process audits depend on appropriately prepared compliance documentation. This paper proposes an **artifact-centric, template-based, compliance demonstration approach** intended to streamline ISO 26262-6 compliance documentation for software developed using Model-Based Design. The approach is illustrated by using the ‘model used for production code generation’ as an example. The proposed artifact centric compliance demonstration is compared with the traditional approach using method-centric compliance tables.

3 Demonstrating Compliance with ISO 26262-6

Regardless of the compliance demonstration approach (internal audit, external audit, audit by OEM), the applicant needs to **document compliance with the relevant set of ISO 26262 requirements**. For software, this traditionally involves creating tailored instances of the ‘methods tables’ that explain how each recommended method was interpreted and applied to the project at hand.

3.1 Using Tailored Methods Tables

The objectives and requirements outlined in ISO 26262-6 target the quality of the software engineering processes. **Tailored software methods tables** are frequently used to document process compliance based on the applied methods within the context of a particular project (see Tab. 1 for an example). ISO 26262-6 provides 16 methods tables that list the techniques and measures recommended for each ASIL (cf. columns #1 and #2 in Tab. 1). These tables are organized according to the different software lifecycle phases as outlined in the standard (**phase/method-centric compliance documentation**).

In order to demonstrate ISO 26262-6 compliance, these tables are extended by a new column to document if and how a given method was interpreted and applied for the application under consideration (cf. column #3 in Tab. 1). If a particular highly recommended technique or measure is not used, then a justification should be provided.

Practitioners refer to the resulting set of tailored tables as **compliance matrix**. The compliance matrix is used within the compliance demonstration process as evidence that the requirements and objectives of the standard have been met. The completed compliance matrix usually contains a mixture of application specific and application agnostic compliance arguments. Compliance matrices are often recreated from scratch for each application.

Table 7 – Notations for software unit design

Methods		ASIL C	Interpretation in this application
1a	Natural language	++	Used to describe behavior of application software components and basic software
1b	Informal notations	+	Not used
1b	Semi-formal notations	++	Used for all application software components. In particular time-based block diagrams as provided by Simulink 7.5, and event-based state machines as provided by Stateflow 7.5
1d	Formal notations	+	Only exceptionally, for application software components X and Y

Tab. 1: Example of a tailored software method table (cf. ISO 26262-6, Table 7)

Compliance matrices structured as in the given example, have proven helpful in supporting the compliance demonstration process for individual, and self-contained applications. However, they are less tailored to automotive engineering organizations, which develop multiple, related applications following similar processes. The structuring of the tables is also poorly tailored for Model-Based Design projects.

3.2 An Artifact Centric Approach for ISO 26262 Compliance Documentation

3.2.1 Artifact Centric Compliance Demonstration Tables

In Model-Based Design, core software development artifacts such as Simulink and Stateflow models evolve through multiple lifecycle phases (cf. section 1). In addition, more than one development team may work on the same artifact.

In order to provide optimal support for the development teams responsible for a particular artifact, all information relevant to assess standard compliance w.r.t. this artifact should be easily accessible through a single entry point for the given artifact. Such an **artifact-centric structure** is not very well supported with the compliance demonstration means presented thus far in this article.

The issue can be addressed by using an **artifact centric approach for ISO 26262 compliance documentation** comprising the following steps:

1) All major software engineering artifacts are gathered. As outlined in section 1, a typical Model-Based Design process may comprise the following core artifacts: textual requirements, executable specification model, model used for production code generation, generated C source code, object code.
2) For each core artifact, a complete list of the associated ISO 26262-6 requirements is added. Note that this is not necessarily a 1-to-1 mapping, some requirements may belong to more than one artifact. Others may not directly apply.

   The first three columns of Tab. 2 illustrate the structure of an artifact centric compliance table by using the model used for production code generation as an example. For usability reasons, the table divided is into three sections, ‘Constructive Activities’ (i.e. software engineering activities), ‘Verification and Validation Activities’ (quality engineering / assurance activities) and ‘Supporting Activities’.

   For each applicable section of ISO 26262, the requirements derived from the textual part of the standard are listed first. After that, the recommended methods from the corresponding methods tables are listed. To ease discoverability of the tables, table related entries are prefaced with the table symbol ▦. Methods from the software methods tables are referenced as M<tab#>-<meth#>, whereas <Tab#> represents the table number and <meth#> the method number within the table. Notes and examples from the standard are captured using the ⓘ symbol, as is illustrated in the case of clause 8.4.2.

   The standard itself does not always clearly separate the requirements in the textual part from the recommended methods in the tables. Some aspects are covered both in the text and in the tables, others are only covered once. This results in certain redundancies within the table.

3) The requirements and methods are mapped onto Model-Based Design processes and tools as they are used for the application under consideration. Some aspects of the mapping information may be application independent; other parts may need to be augmented with evidence demonstrating the fulfilment of the corresponding requirements. If application specific information is required, this should be clearly identified in the table.

   This is shown in the rightmost column of Tab. 2. The table shows an exemplary mapping of the constructive activities pertaining to model development. Further information on the verification and validation activities for the model can be found in [CS09, Con12].

3.2.2 Using Templates to Facilitate Reuse

Organizations that run multiple software projects with similar development processes and tool chains look for compliance matrices that facilitate re-use and sharing of best practices (cf. [CD06, Con07]). Therefore it is desirable to factor out the application-independent parts and make them available for reuse by means of templates.

As a first step to avoid starting from scratch every time a new project starts, columns 1 … 3 could be provided as templates. Changes are minimal as long as the list of artifacts remains unchanged.

Different organizations may have different interpretations on applicable requirements, recommendations, and methods for a given artifact. Therefore the applicant should get buy-in from relevant stakeholders (incl. certification authority) for this list.

Column 4 also contains significant application-independent information that can be reused across projects that share the same development processes and tool chains. This is appealing to larger development organizations and product lines. Under such circumstances, the central elements of the tool chain used in function development are usually stable for an extended period of time.

It may be worth replacing or refining the information in ‘Model-Based Design processes and tools; Interpretation in this application, Evidence’ with information specific to the release of the Simulink product family [SL, SF] that is currently used for function development. In other words, one should tailor and select the techniques and tools to apply the standard for a specific project.

Significant effort savings are expected if projects reuse those pre-filled templates as basis for their compliance demonstration process. Only project / application specific information such as references to actual work products and corresponding status information need to be modified or included.

The proposed approach also results in ‘standardized’ compliance matrices. If an applicant using such an approach repeatedly works with the same certification authority, the assessors will get more familiar with the information provided over time. Efforts to clarify issues and miscommunications can be reduced greatly. The overall compliance assessment effort is also reduced.

Based on this information, each artifact that is in the scope of the project team can be assessed with respect to its associated requirements. Compliance with these requirements could be made part of the artifact's sign-off procedure.

4 Summary and Conclusion

ISO 26262 is increasingly being used to develop high-integrity systems in the automotive industry. The objectives and requirements of the standard apply to hand-coded software as well as to software developed using Model-Based Design. The fusion of software lifecycle phases in Model-Based Design and the important role of executable models within these phases motivate a specific tailoring of the standard if applied in the scope of Model-Based Design projects.

This tailoring affects the ISO 26262 compliance demonstration process as well. Carrying out traditional compliance documentation efforts; i.e., by using and extending the ISO 26262-6 software methods tables, has various disadvantages.

The proposed artifact-centric and template-based ISO 26262 compliance demonstration approach in the context of Model-Based Design provides an engineering solution to capture the requirements associated with a given software development artifact and to assess their realization in a given project. The proposed structure also facilitates a department or company wide deployment if multiple projects standardize on the same processes, tool chains and versions. It also supports developers and quality managers in their day-to-day work.

The approach was inspired by a similar approach for the IEC 61508 [IEC 61508] standard [CF09], but had to be tailored to the specifics of ISO 26262-6.

It shall be noted, that the compliance demonstration discussed here is not sufficient to establish the overall safety of the system. It needs to be augmented by other documents and artifacts including a system-specific argumentation such as a safety case (see e.g. [RDG07]).

Worked Example

Applicant:	<application specific>
Application / Project:	<application specific>
Document Version:	<application specific>
Date:	<application specific>
Tool Versions Used:	Simulink®, Stateflow®, Embedded Coder™, Simulink® Verification and Validation™, Simulink® Design Verifier™, PolySpace® Products for C, IEC Certification Kit (R2011b)

	Technique/Measure, Associated Requirements	Ref. to ISO 26262-6	Model-Based Design processes and tools; Interpretation in this application, Evidence
Model used for production code generation Model name: <application specific> (see: <add link/reference>)	CONSTRUCTIVE ACTIVITIES		
	Initiation of product development at the software level – Requirements and recommendations	5.4	
	To support the correctness of the design and implementation, the design guidelines for the modelling, languages, shall address the topics listed below ⓘ ⓘ ⓘ	5.4.7	
	▦ Topics to be covered by modelling guidelines	Tab. 1	<application specific>The guideline subset used for the project has been reviewed to address a combination of topics applicable for the ASIL under consideration (review log: <add link/ reference>).
	Enforcement of low complexity	M1-1a	The Modeling Guidelines for High-Integrity Systems (R2011b) and the MathWorks® Automotive Advisory Board - Control Algorithm Modeling Guidelines Using MATLAB®, Simulink, and Stateflow® (R2011b) are used to address topics listed in this table.
	Use of language subsets ⓘ	M1-1b	
	Enforcement of strong typing ⓘ	M1-1c	
	Use of defensive implementation techniques	M1-1d	
	Use of established design principles	M1-1e	
	Use of unambiguous graphical representation	M1-1f	
	Use of style guides	M1-1g	
	Use of naming conventions	M1-1h	
	...	...	...
	Software architectural design – Requirements and recommendations	7.4	
	...	...	...
	Software unit design and implementation – Requirements and recommendations	8.4	
	The requirements of this subclause shall be complied with if the software unit is safety-related. ⓘ	8.4.1	ASIL level: <application specific>
	The software unit design shall be described using the following notations: ⓘ	8.4.2	
	▦ Notations for software unit design	Tab. 7	The selected combination contains 1 highly recommended method for ASIL A and 2 highly recommended methods for ASIL B-D
	Natural language	M7-1a	Simulink 'Model Info block' or 'DocBlock' blocks are used to add natural language or descriptions of a unit design to a model.
			Simulink Verification and Validation – Requirements Management Interface (RMI) is used to link models

	Technique/Measure, Associated Requirements	**Ref. to ISO 26262-6**	**Model-Based Design processes and tools; Interpretation in this application, Evidence**
			representing unit designs to descriptions in Microsoft Word, Microsoft Excel, ASCII text, or PDF files.
	Informal notations	M7-1b	Not used
	Semi-formal notations	M7-1c	Semi-formal notations provided by Simulink and Stateflow are used for all application software components.
	Formal notations	M7-1d	Not used
	The specification of the software units shall describe the functional behaviour and the internal design to the level of detail necessary for their implementation. ⓘ	8.4.3	The model used for production code generation contains the information needed by the code generator to create the source code
	Design principles for software unit design and implementation at the source code level as listed below shall be applied to achieve the following properties: a) correct order of execution of subprograms and functions within the software units, based on the software architectural design; b) consistency of the interfaces between the software units; …	8.4.4	
	▦ Design principles for software unit design and implementation	Tab. 8	<application specific>The chosen combination of methods has been reviewed to be suitable for the ASIL under consideration (review log: <add link/ reference>).
	One entry and one exit point in subprograms and functions ⓘ	M8-1a	Adherence can be facilitated by applying modeling guidelines in combination with analyzing generated code. MAAB guideline jc_0511 provides corresponding modeling recommendations. Polyspace can assess compliance with MISRA–C:2004 rule 14.7.
	No dynamic objects or variables, or else online test during their creation ⓘ ⓘ	M8-1b	Embedded Coder has been configured to generate C code that does not include dynamic objects. <application specific> (Configuration set used: <add link/ reference>).
	Initialization of variables	M8-1c	Simulink 'IC' blocks are used to specify the initial condition for signals (if applicable). The **Underspecified initialization detection** diagnostic is set to `Simplified` to improve consistency of simulation results for models that do not specify initial conditions for conditional subsystem output ports or have conditionally executed subsystem output ports connected to S-functions.
	No multiple use of variable names ⓘ	M8-1d	…

	Technique/Measure, Associated Requirements	**Ref. to ISO 26262-6**	**Model-Based Design processes and tools; Interpretation in this application, Evidence**
	Avoid global variables or else justify their usage ⓘ	M8-1e	…
	Limited use of pointers ⓘ	M8-1f	…
	No implicit type conversions ⓘ ⓘ	M8-1g	…
	No hidden data flow or control flow ⓘ	M8-1h	…
	No unconditional jumps ⓘ ⓘ ⓘ	M8-1i	…
	No recursions	M8-1j	…
	…	...	…
	VERIFICATION AND VALIDATION ACTIVITIES		
	...	…	…
	Software unit design and implementation – Requirements and recommendations	8.4	
	The software unit design and implementation shall be verified in accordance with ISO 26262-8 Clause 9, and by applying the verification methods listed below, to demonstrate: a) the compliance with the hardware-software interface specification (in accordance with ISO 26262-5, 6.4.10); b) the fulfilment of the software safety requirements as allocated to the software units (in accordance with 7.4.9) through traceability; …	8.4.5	
	▦ Methods for the verification of software unit design and implementation	Tab. 9	<application specific>The chosen combination of methods has been reviewed to be suitable for the ASIL under consideration (review log: <add link/ reference>).
	Walk-through ⓘ	M9-1a	Not used
	Inspection ⓘ	M9-1b	Unit design inspections are carried out based on Web View of the model generated by Simulink Report Generator. Unit design inspections are supported by ISO 26262, and MAAB checks in Model Advisor. <application specific> The Model Advisor check configuration<add link/ reference>) defines a set of checks to pass as a prerequisite for entering the model inspection.
	Semi-formal verification	M9-1c	…
	Formal verification	M9-1d	...
	Control flow analysis ⓘ ⓘ	M9-1e	…
	Data flow analysis ⓘ ⓘ	M9-1f	…
	Static code analysis	M9-1g	…

	Technique/Measure, Associated Requirements	**Ref. to ISO 26262-6**	**Model-Based Design processes and tools; Interpretation in this application, Evidence**
	Semantic code analysis ⓘ	M9-1h	...
	…	…	…
	SUPPORTING ACTIVITIES		
	...	...	…
	….	...	…

Note: … indicate information left out of this example for brevity or clarity purposes.

Tab. 2: Example of an artifact centric compliance demonstration table

References

[Con07] M. Conrad: Using Simulink® and Real-Time Workshop® Embedded Coder for IEC 61508 Applications. White Paper, Safety Users Group, 2007 http://www.safetyusersgroup.com/documents/AR070002/EN/AR070002.pdf

[Con12] M. Conrad: Verification and Validation According to ISO 26262: A Workflow to Facilitate the Development of High-Integrity Software. Proc. Embedded Real Time Software and Systems (ERTS^{2}2012), Tolouse, FR, Feb. 2012

[CD06] M. Conrad, H. Dörr: Deployment of Model-based Software Development in Safety-related Applications - Challenges and Solutions Scenarios. Proc. Modellierung 2006, Innsbruck, Austria, März 2006, LNI Vol 82, p. 245-254

[CF09] M. Conrad, I. Fey: Demonstrating IEC 61508 Compliance in Model-Based Design. Proc. Dagstuhl-Workshop Modellbasierte Entwicklung eingebetteter Systeme (MBEES09), Schloß Dagstuhl, DE, April 2009, pp. 171-181

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

[CS09] M. Conrad, G. Sandmann: A Verification and Validation Workflow for IEC 61508 Applications. SAE Techn. Paper #2009-01-0271, SAE World Congress 2009

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

[HKM05] A. Helmerich, N. Koch, L Mandel, L. et al.: Study of worldwide trends and R&D programs in embedded systems in view of maximising the impact of a technology platform in the area. Report for the European commission, Brussels, Belgium.

[IEC 61508] IEC 61508:2010. Int. Standard Functional safety of electrical/ electronic/ programmable electronic safety-related systems. 2010

[ISO 26262] ISO 26262:2011. Int. Standard Road vehicles - Functional safety. 2011

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

[MBD] Model-Based Design web page. The MathWorks Inc., www.mathworks.com/applications/controldesign/description

[RGD07] W. Ridderhof, H.-G. Gross, H. Dörr: Establishing Evidence for Safety Cases in Automotive Systems – A Case Study. SafeComp 2007

[EC] Embedded Coder^{TM} product page. The MathWorks Inc., www.mathworks.com/products/ecoder

[SF] Stateflow® Product Page. The MathWorks Inc., www.mathworks.com/products/stateflow

[SL] Simulink® Product Page. The MathWorks Inc., www.mathworks.com/products/simulink