ISO 26262 – Exemplary Tool Classification of Model-Based Design Tools

Mirko Conrad[1], Ines Fey[2]

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

[2] samoconsult GmbH
Berlin, Germany
ines.fey@samoconsult.de

Abstract: Tool classification is an important part of the tool qualification process required by ISO 26262 since it determines the required confidence level for each tool in use. To cover the variety of tools used by practitioners, the standard only provides a framework for tool classification and leaves it up to the applicant to instantiate this framework.

To illustrate the ISO 26262 tool classification procedure, this paper provides an exemplary tool classification for the Model Advisor, a static analysis tool used in Model-Based Design,

By putting this example into the context of a practical tool qualification approach for COTS tools, the author's report their experiences in instantiating the ISO 26262-8 tool qualification framework.

1 Tool Qualification According to ISO 26262

According to ISO 26262, a software tool can support or enable tailoring of the lifecycle through tailoring of activities required by the standard. In these cases, confidence is needed that the usage of a tool:

1. Decreases the risk of systematic faults in the system under development caused by malfunctions of this tool or its erroneous output.
2. Promotes the adequacy of the development process w.r.t. compliance with ISO 26262, if activities or tasks required by ISO 26262 rely on the correct functioning of this tool.

Chapter 11 of *ISO/FDIS 26262-8* defines a *process to gain confidence in the usage of a software tool*, a.k.a. *software tool qualification*. The required level of confidence in the tool is determined by:

- The possibility that the malfunctioning software tool or its erroneous output can either introduce errors or fail to detect errors in the system/item being developed.
- The confidence in preventing or detecting such errors in the tool's output.

To evaluate the confidence in prevention or detection measures, tool-internal measures (e.g. monitoring) as well as tool-external measures (e.g. guidelines, tests, reviews) implemented in the software lifecycle can be taken into account.

If required by the tool confidence level determined, appropriate qualification methods need to be applied to comply with both the tool's confidence level and the maximum ASIL among the safety requirements allocated to the system / item to be developed using the software tool.

2 Two-stage Qualification of COTS Tools

The knowledge of the actual tool usage and its embedding into the software development process used in the actual project is crucial to the ISO 26262 tool classification and qualification approach. Since only the tool user has this knowledge at their disposal, the responsibility for the final tool classification and potentially tool qualification in the context of the application lies with the tool user.

However, tool classification and qualification also encompass generic aspects and activities that are common across various users. In case of commercial-off-the shelf (COTS) tools, the tool vendor can facilitate the tool qualification process by documenting generic aspects, conducting generic activities, and providing the resulting information to its tool users to simplify their tool qualification activities.

Similar as for other standards (e.g. DO-178B), users of COTS tools expect their tool vendors to provide *ISO 26262 tool qualification packages* that can be adapted and instantiated with limited effort.

Tool qualification in such a situation can be achieved by a *two-stage qualification approach* that is characterized by sharing the tool qualification activities between the tool vendor and the tool user [Con10, CSM10, HKW+11].

Stage I, the *application-agnostic pre-qualification*, comprises the following steps:

I.a. Generic tool classification [Tool vendor]

I.b. Generic pre-qualification up to max. required TCL [Tool vendor]

I.c. Independent assessment of generic tool classification and pre-qualification [External organization]

The pre-qualification results in an *ISO 26262 tool qualification kit* as mentioned above. Even not required by ISO 26262, an independent assessment by a certification authority or a consulting firm with sufficient experience and reputation can be used as a means to increase the confidence in the pre-qualification and the tool qualification kit.

Stage II, the *application-specific adaptation*, encompasses a

II.a. Review / adaptation of the tool qualification kit [Tool user]

The application-specific adaptation results in the final *ISO 26262 tool qualification documentation*.

The generic, *application-agnostic* pre-qualification, i.e. stage I, is based on *common, typical tool use cases*. It can be augmented by a *reference workflow* to be utilized by the tool user when using the tool for developing or verifying safety-related software. In terms of ISO 26262, the reference workflow describes error prevention and detection measures employed when using the tool.

In the generic tool classification, the tool is classified assuming that it is used as specified by the typical use cases and tool usage is supported by the error prevention and detection methods described in the reference workflow. The maximum required tool confidence level (TCL) resulting from the generic tool classification is used to determine the necessary tool qualification methods.

An example generic tool classification will be discussed in the next section.

If an independent assessment is utilized, the tool qualification artifacts created by the tool vendor are submitted to the external organization for review and approval. The assessor states the adequacy of the classification and the pre-qualification artifacts in an *assessment report*. If the assessment is carried out by a notified body such as the TÜV, a corresponding *certificate* can be issued.

To further support tool users, the tool vendor can also create a *tool qualification package* (TQP) containing pre-filled templates for the tool qualification artifacts as described in ISO 26262-8.

An example of a tool qualification kit created using this approach is MathWorks *IEC Certification Kit* [IECCertKit]. The ISO 26262 portion of the kit[1] contains ISO 26262 tool qualification kits for the Embedded Coder™ code generator [ECoder] and the Polyspace® code verifiers [Polyspace]. The independent assessments for these products were carried out by TÜV SÜD.

For stage II, the tool user needs to review the template documents for applicability to the system / item under consideration and to instantiate or adapt the information provided.

If the pre-qualified tool is used within the perimeter of the use cases and the reference workflow, the user can directly leverage the pre-qualification by referencing the certificate and the certificate report.

If the user uses the tool differently, the tool classification needs to be carried out according to the actual use case(s). However, if the required TCL derived from the actual use cases is equal or lower than the TCL that resulted from the typical use cases, the generic tool qualification could still be leveraged. [CMS10] provides an example for an application-specific adaptation of an ISO 26262 tool qualification kit.

[1] The IEC Certification Kit also contains documents, artifacts, and tools to support certification and qualification activities according to the IEC 61508 base standard and additional domain-specific derivative standards.

3 Tool Classification Example: Model Advisor

Model-Based Design of Automotive Application Software Components

Because of its capability to address software complexity and productivity challenges, *Model-Based Design* [HKM+05, CFG+05, CD06] is quickly becoming a widespread software engineering paradigm for the development of embedded application software components across industries. For example, practitioners report productivity gains of 4 to 10 times when using graphical modeling techniques [HKM+05, KHJ07]. In a typical Model-Based Design workflow, an initial *executable graphical specification* representing the application software component under development is refined and augmented until it becomes the blueprint for the final implementation through automatic *code generation*.

Model-Based Design is one of the software development paradigms directly addressed in ISO 26262. Annex B of ISO 26262-6 describes this development paradigm in more detail.

Simulink® [Simulink, Mos06] is a widely used Model-Based Design environment. It supports the description of both discrete- and continuous-time models in a graphical block-diagram language. Embedded code generation from Simulink models is supported by *Embedded Coder™* [ECoder]. The code generator is capable of translating Simulink models into C or C++ code that can be deployed onto embedded systems.

The Simulink environment is extensively used in different application domains. According to [HKM+05], 50% of the behavioral models for automotive controls are designed using this environment. Because of the popularity of this environment, an analysis tool for Simulink models is used in this paper to illustrate the realization of the ISO 26262 tool classification approach.

Static Analysis at the Model Level

When using Model-Based Design for ISO 26262 applications, the detailed design of application software components can be implemented as a model, in accordance with a set of modeling guidelines (a.k.a. design and coding guidelines for the modeling language). The detailed design needs to be verified before proceeding to the next development phase [ISO 26262-6]. Part of this verification process is the static analysis of the Simulink model against the applicable modeling guidelines. Model checks can be carried out by using the Model Advisor [MdlAdv] a static analysis tool integrated into the Simulink environment. Additional Model Advisor checks focused on specific standards, such as ISO 26262,

are provided by Simulink® Verification and Validation™. Fig. 1 illustrates the usage of Model Advisor to check a set of modeling guidelines.

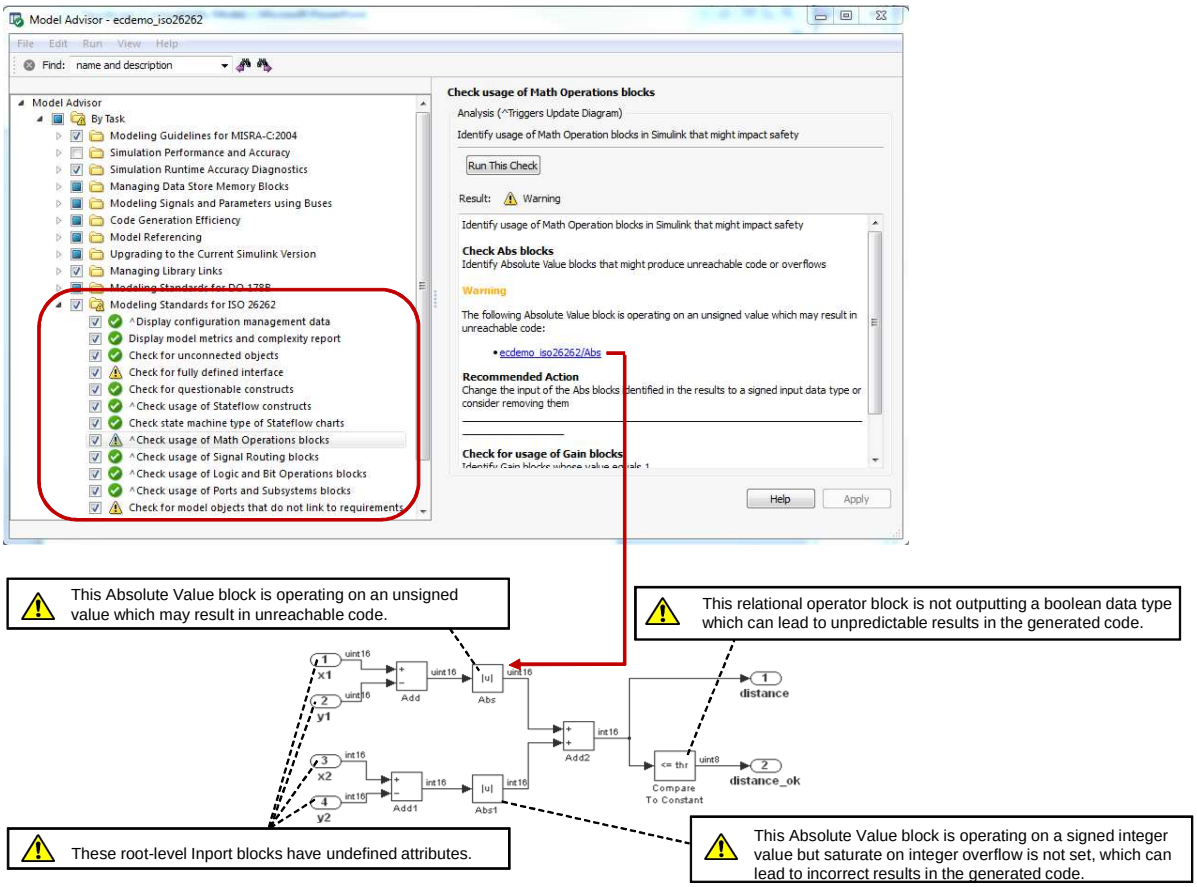

Figure 1. Example of using modeling standard checks in Model Advisor.

I.a. Generic Tool Classification of Model Advisor

The software tool classification is crucial for the ISO 26262 tool qualification approach. The tool user needs to carry out the tool classification for all tools used in the software lifecycle. The outcome of the analysis determines whether or not a formal tool qualification is necessary.

If the two-stage tool qualification approach outlined in the previous chapter is utilized, the actual (i.e. project-specific) classification can be based on a generic tool classification provided by the tool vendor.

Tool Interfaces

A prerequisite for the tool classification is a description of the inputs and outputs to the tool.

Example: Model Advisor uses the inputs and creates the outputs subsequently listed:

Tool Inputs

- Simulink model to be analyzed
- Check configuration (selection of checks to be applied)
- Input parameters for parameterized checks [if applicable]

Tool Outputs

- Analysis results with Hyperlinks to the Simulink model (Model Advisor check report)
- Corrected model [if applicable]

Tool Use Cases

The generic tool classification depends on typical tool use cases (a.k.a. reference use cases) used.

Example: The above description of Model Advisor leads to the following use cases:

[UC1] Static analysis of Simulink models to verify compliance with specified modeling guidelines

[UC2] Automatic fixing of reported issues

The tool classification continues with analyzing the tool use cases

1. To evaluate the possibility that a malfunction of the tool can introduce or fail to detect errors in the system under development
2. To determine the confidence in measures to prevent or detect these errors.

The tool classification facilitates the determination of the necessary *tool confidence level* (TCL) and follows the schematics provided in Fig. 2. The TCL in conjunction

with the Automotive Safety Integrity Level (ASIL) of the safety-related software developed using the software tool, determines the selection of the appropriate tool qualification methods.

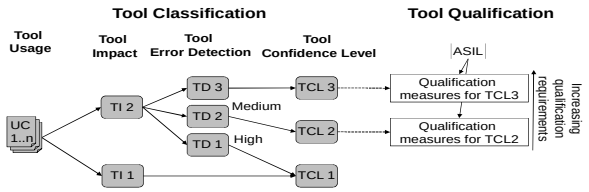

Figure 2. Tool Classification Scheme acc. to ISO 26262-8

Malfunctions / Erroneous Tool Output

The generic use cases for the tool are analyzed to evaluate if a malfunction of the tool can introduce or fail to detect errors in the system under development. If it can be argued that there is no such possibility, tool impact TI1 shall be chosen. Otherwise, the tool impact is TI2.

To facilitate a systematic analysis, the malfunctions or erroneous tool outputs considered should be listed.

Example: The following potential malfunctions or erroneous outputs of the Model Advisor are taken into account for the generic tool classification:[2]

(E1) False negative: The modeling guideline checker incorrectly marks model as compliant

(E2) False positive: The modeling guideline checker incorrectly marks model as non-compliant

(E3) Non interference: The modeling guideline checker contains an error, but model to be analyzed does not invoke the erroneous portion of the tool

(E4) Incorrect hyperlinks: Hyperlinks in the analysis results contain errors.

(E5) Incorrect fixing of reported issues: Automatic fixing of reported issues does not work correctly.

(E6) Misinterpretation of results: User interprets correct analysis results incorrectly

Tool Error Detection (TD)

The intended software tool use cases are analyzed to determine the probability of preventing or detecting that the software tool is malfunctioning or producing erroneous output. The degree of confidence, that a malfunction or an erroneous output from the tool can be prevented or detected, determines the tool error detection (TD) as outlined in Fig. 3.

Degree of confidence	Tool error detection
high	TD1
medium	TD2
all other cases	TD3

Figure 3. Tool Error Detection Levels

[2] The actual tool classification contains additional error cases.

Tool Confidence Level (TCL)

If TI and TD have been chosen, the tool confidence level (TCL) can be determined following the schematics provided in Fig. 2.

Having multiple use cases for a tool can potentially result in multiple TCLs. To determine the required tool qualification methods, the *maximum TCL required (TCL_{REQ})* to support these use cases needs to be established.

Example: A table format (cf. [Mai09]) can be used to document the determination of TI, TD and TCLs. Table 1 summarizes this process for the Model Advisor example.

I.b. Generic Pre-qualification of Model Advisor

As per the tool classification summary in Table 1, the determined maximum TCL for Model Advisor is $TCL_{Max}2$. To achieve the necessary confidence in the modeling guideline checker when used within the perimeter of the generic tool classification, Model Advisor needs to be qualified for TCL2. This can be achieved for example by a test suite that validates the modeling standard checks used for modeling guideline checking.

For the shipping Model Advisor checks, this test suite could be provided by MathWorks.

The generic tool classification and pre-qualification result in an *ISO 26262 tool qualification kit.* Optionally, an independent assessment can be used as a means to increase the confidence in the pre-qualification and the tool qualification kit. In the above example, the tool vendor would –among other things- provide the assessor with information on the validation test suite.

II.a Review / Adaptation of the Tool Qualification Kit for Model Advisor

The tool user's activities within the two-stage qualification process start with reviewing the documents in the tool qualification kit regarding assumed use cases and detection measures for potential tool malfunctions. Based on the use cases and the tool qualification summary given in Table 1, the user would identify the information applicable to the project at hand and remove the non-relevant entries.

Example: To illustrate this step, we focus on use case [UC2] and the corresponding malfunction (E5).

In a first scenario we assume that the tool user does not use the automatic fixing capability of Model Advisor. In this case, [UC2] from the generic tool classification would be deleted. The user would also delete malfunction (E5) and all rows in the tool classification summary that are related to this malfunction.

In an alternative second scenario it is assumed that the project uses the automatic fixing capability to reduce the effort associated with correcting the model. We further assume that the project uses 'Re-checking of the model after automatic fixing of reported issues' as the only measure to mitigate potential issues with the fixing

capability. The likelihood that re-checking of the model detects unintended changes introduced by the auto fixing capability is considered to be ‘medium’ (TD2). Therefore the corresponding Tool Confidence Level for E5 is TCL 2. The user would delete the rows describing the two alternative mitigation measures for (E5) since these measures are not used by the project.

Since the tool user only utilizes a subset of the tool use cases and the resulting TCL does not exceed the maximum TCL determined in the generic tool classification, the user won’t need to start further activities to increase the tool’s confidence. The user can refer to the generic pre-qualification and the independent assessment of the pre-qualification.

However, the pre-qualification can only cover the shipping Model Advisor checks. If the project uses additional custom checks, these checks need to be qualified by the user, i.e. the user would need to extend or modify the test cases in the qualification test suite to provide confidence in the proper functioning of the custom checks and their auto fixing capabilities.

Table 1. Tool Classification Summary for Model Advisor

ID	Potential malfunction or erroneous output	Use case	TI	Justification for TI	Measure(s) to prevent or detect error in tool output	TD	Justification for TD	TCL
E1	False negative: The modeling guideline checker incorrectly marks model as compliant	UC1	TI2	Incorrect analysis result could prevent modeling guidelines violations from being detected	Preceding or subsequent dynamic verification (testing) of the model	TD2	Static analysis tools typically detect only a subset of the existing modeling standard violations in the model. Therefore, other process steps can not assume completeness of modeling guideline check results. Modeling standard violations do not necessarily imply incorrect models. Functional or structural testing help detect real errors in the model. The likelihood of detecting these errors by testing is considered to be ‘medium’.	TCL2
E2	False positive: The modeling guideline checker incorrectly marks model as non-compliant	UC1	TI1	Nuisance only, software does not violate modeling guidelines	-	-	-	TCL1
E3	Non interference: The modeling guideline checker contains an error, but model to be analyzed does not invoke the erroneous portion of the tool	UC1	TI1	Error in the tool does not affect analysis results	-	-	-	TCL1
E4	Incorrect hyperlinks: Hyperlinks in the analysis results contain errors.	UC1	TI1	Nuisance only, software does not violate modeling guidelines	-	-	-	TCL1
E5	Incorrect fixing of reported issues: Automatic fixing of reported issues does not work correctly.	UC2	TI2	Incorrect fixing could introduce error in the model	Re-checking of the model after automatic fixing of reported issues	TD2	Re-checking of the model will detect modeling standard violations introduced by the automatic fixing but might miss other errors introduced	TCL2
					Subsequent dynamic verification (testing) of the model	TD2	Functional or structural testing help detect real errors in the model. The likelihood of detecting these errors by testing is considered to be ‘medium’.	TCL2
					Automatic comparison of XML files exported from the original and fixed Simulink models and subsequent manual review of the comparison results	TD1	Manual review of the comparison results can verify that fixing of changes resulted did not introduce un-intended changes	TCL1

ID	Potential malfunction or erroneous output	Use case	TI	Justification for TI	Measure(s) to prevent or detect error in tool output	TD	Justification for TD	TCL
E6	Misinterpretation of results: User interprets correct analysis results incorrectly	UC1	TI2	Misinterpretation of analysis results could prevent errors from being detected	Adequate competency of the project team	TD1	Adequate training of users can prevent these issues	TCL1

4 Summary and conclusion

ISO 26262-8 provides a tool classification framework for software tools but leaves its implementation up to the applicant. By using the example of Model Advisor, a typical verification tool included in a Model-Based Design tool chain, the authors provided an example of how the tool classification process could be implemented in practice. They also showed how the tool classification could incorporate an application-agnostic pre-qualification that could be carried out by the tool vendor. Under this scenario, the amount of work to be done by the tool user is significantly reduced such that the user only needs to review the generic tool classification for applicability to the system / item under consideration and to instantiate or adapt the tool classification information provided by the tool vendor.

The exemplary tool classification of Model Advisor resulted in $TCL_{MAX}2$. This result is consistent with the TCL for design and coding guideline checkers provided in [LPP10].

5 References

[CD06] CONRAD, M., AND DÖRR, H. 2006. Model-based development of in-vehicle software. In *Proc. Conf. on Design, Automation and Test in Europe (DATE '06)*, Munich, Germany, 89-90.

[CFG+05] CONRAD, M., FEY, I., GROCHTMANN, M., AND KLEIN, T. 2005. Modellbasierte Entwicklung eingebetteter Fahrzeugsoftware bei DaimlerChrysler. *Inform. Forsch. Entwickl.* 20(1-2): 3-10.

[Con10] CONRAD, M.: Software Tool Qualification According to ISO 26262 - An Experience Report. Supplementary Proc. of 21. Int. Symposium on Software Reliability Engineering (ISSRE 2010), pp. 460-466

[CSM10] CONRAD, M., SAULER, J., MUNIER, P.: Experience Report: Two-stage Qualification of Software Tools. 2. EUROFORUM Conference ISO 26262, Stuttgart, Germany, September 2010

[ECoder] Embedded Coder™. www.mathworks.com/products/rtwembedded

[HKM+05] HELMERICH, A., KOCH, N., MANDEL, L. ET AL. 2005. 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.

[HKW+11] HAMANN, R., KRISO, S., WILLIAMS, K., KLARMANN, J., AND SAULER, J 2011: ISO 26262 Release Just Ahead: remaining problems and Proposals for Solutions. SAE 2011 World Congress, Detroit, MI, US, April 2011.

[IECCertKit] IEC Certification Kit. www.mathworks.com/products/iec-61508

[KHJ07] KAMGA, J., HERRMANN, J., AND JOSHI, P. 2007: Deployment of model-based technologies to industrial testing. D-MINT automotive case study – Daimler, ITEA 2 Project Deliverable 1.1. .

[LPP10] LÖW, P., PABST, R., AND PETRY, E.: Funktionale Sicherheit in der Praxis. dpunkt Verlag, 2010

[Mai09]: MAIHÖFER, M. 2009: Umgang mit Entwicklungs-werkzeugen in Software-Entwicklungsprozessen der Automobil-industrie – ISO/DIS 26262, Band 8, Kapitel 11: Inhalt, Bewertung, Auswirkung und Umsetzung (in German). EOROFORUM Konferenz 'Funktionale Sicherheit nach ISO/DIS 26262', Stuttgart, Germany, September 2009

[MBD] Model-Based Design. www.mathworks.com/model-based-design

[MdlAdv] Model Advisor. blogs.mathworks.com/seth/2008/11/04/introduction-to-model-advisor/

[Mos06] MOSTERMAN, P. J.: Automatic Code Generation: Facilitating New Teaching Opportunities in Engineering Education. 36. Annual ASEE/IEEE Frontiers in Education Conf, San Diego, USA, 2006, pp. 1 - 6

[Polyspace] Polyspace® for C/C++. www.mathworks.com/products/polyspace

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