Modeling and Safety-Certification of Model-based Development Processes

Oscar Slotosch[1] and Mohammad Abu-Alqumsan [2]

Abstract: In this paper, we describe a two-step approach to show evidence for compliance with safety standards within certification efforts for model-based development projects that share some commonalities (i.e. using the same metamodel). The approach is based on modeling model-based development processes in combination with the requirements imposed on them by safety standards. Besides the typical benefits of model-based approaches (modularity, rigor, formalization and simulation), we use the combined hierarchic processes-requirements model in order to automatically generate formalized descriptions of processes, standard compliance report and verification check-lists. The process description can be used to introduce new team members to the deployed development processes. As a concrete example of the proposed approach, we present representative parts of the Validas model-based tool qualification process that has been fully modeled and certified based on the automatically generated documents by TÜV SÜD.

Keywords: Model-based Development, Process Model, Safety Standards, Tool Qualification

1 Introduction

Thanks to the easiness in which domain-specific models can be developed for dedicated purposes, model-based development is increasingly used in more areas of applications. Yet, there are remaining areas of applications where models do not enjoy the expected/desired level of acceptance, even when promising examples and pilot cases exist. This may suggest that many developers resist switching to model-based development, possibly because they do not want or are not able to do so. It is common in practice to encounter the following arguments against introducing model-based development: “there is no detailed process description of the model-based approach”, “it is not clear how to do this with my tool X”, “the compliance of the model-based development process to safety standards is unclear”, or “the modeling tools cannot be used in safety critical projects, since they are not qualified (or certified)”. Obviously, these statements/arguments are not against the model-based approach per se, but rather are manifestations of why developers may resist introducing/switching to model-based development.

Furthermore, quality aspects of software like consistency, reproducibility and repeatability and roll-out planning cannot be ensured without the availability of a precise process description. Consider for instance a situation where a company’s management

[1] Validas AG, Arnulfstr. 27, 80335 München, slotosch@validas.de
[2] Validas AG, Arnulfstr. 27, 80335 München, abu-alqumsan@validas.de

has decided to use UML (and a specific tool) for the specification of software algorithms in a pilot project. Assume as well that the team is willing to do so, but some team members use state transition diagrams, others sequence diagrams and some others use activity diagrams. Furthermore, some even have developed a very sophisticated combination of component diagrams, class diagrams and C++ code that requires new stereotypes and small changes in the tool. Even if the pilot project may become a success, it is obviously impossible to roll it out to the whole company without making a more detailed description of the modeling process available. A qualification of the used tool might additionally be required according to relevant safety standards.

In the present work, we aim at bridging the gap of lacking such descriptions of modeling processes with a novel approach. The same approach is additionally used for the purposes of facilitating certification efforts, by straightforwardly linking relevant process activities to the corresponding requirements imposed by relevant safety standards.

Hereby, the crucial point to meet these two goals (i.e. providing formal description and examining compliance) is to decide upon and to use the right abstraction level when describing processes and tool usage. While it suffices, from a safety point of view, to satisfy a requirement by tracing it to the respective activity/-ies and its/their produced documentation/s, this is yet insufficient, and often not even necessary, for effective and efficient introduction and usage of models/tools. For the latter case, descriptions have to be provided in more details, but not to the point where concrete objects/examples are being described. Otherwise, repeatability and reproducibility in similar projects would be harmed. We recognize that the metamodel is a well suited abstraction level for both the reasoning on compliance to safety standards and the description of model-based development processes. Since processes consist of actions and input/output artifacts we decided to model the metamodel itself as an artifact that is processed (i.e. created, updated/extended) from within process activities. This renders the proposed approach most suitable when the metamodel is used in several concrete development projects of similar nature.

Without harming the generality of the proposed approach, the present paper focuses on the application of the proposed method for model-based tool qualification processes deployed at Validas AG. In particular, we show how the approach is used to show the compliance of our processes to relevant safety standards.

The remainder of this paper is structured as follows. In Section 2 we introduce our general approach for modeling model-based development. Section 3 roughly introduces model-based tool qualification and the metamodel we use to this end. In Section 4, we introduce how the proposed approach is actually done for modeling the MetaModel in tool qualification projects. Section 5 provides more details on the whole structuring of processes and requirements relevant to tool qualification projects. Section 6 provides some details regarding a practical example from the industry: The certification of Validas tool qualification processes by TÜV SÜD. Section 7 concludes with a summary.

2 Modeling Model-Based Development Processes for Compliance and Formal Description

In order to achieve compliance and to provide formal description of model-based development processes, we propose to jointly formalize safety standard requirements and model-based development processes. To this end, we use the process modeling and requirements management framework AutoFOCUS3[3]. This open-source software tool allows modeling of requirements and processes in a hierarchic manner. It additionally supports backward and forward tractability to requirements. Further, the tool provides a strongly-typed environment with support for enumeration and structure data types and additionally provides the possibility to define functions based on user-defined data types.

The tool also supports simulation (module test) of modeled processes with the help of the so called "DTD Evaluator", which is a functional interpreter able to process user defined functions on user defined data types.

Altogether, these features render the tool ideal for our purposes (some benefits of these features will become clearer in later sections). Standard compliance examination of the model-based development process is developed and achieved with a set of utilities and extensions that allow for automatic generation of documentation and work products. We refer to these extensions as TOPWATER extensions as the name of the project in which they were developed.

In particular, the following documents can be automatically generated through the TOPWATER extensions:

1. *The Formal Process Description*: is integrated eventually as an appendix within a manually written process description
2. *The Compliance Report:* showing the satisfaction of the model to all relevant requirements and their traces.
3. *The verification and validation (V&V) Plan:* V&V have to be performed according to this plan in each project that claims compliance to the standards.

Obviously, evidence for compliance of a concrete project to safety standards is achieved in two steps. Firstly, the general *Compliance Report* shows the general compliance of the methodology to the safety standard. Secondly, the *V&V Report* (produced by following the *V&V Plan*) shows the compliance of the concretization of the general methodology within a concrete project. As such, the *compliance Report* and the *V&V Plan* are the same for all projects that share the same model and modeling process and the *V&V Report* is unique for each project.

The overall safety compliance approach is depicted in Fig. 1.

[3] AutoFOCUS3 is an open-source software and can be freely downloaded from https://af3.fortiss.org/download

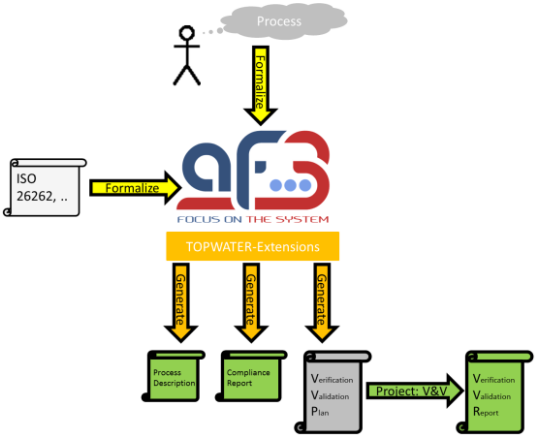

Fig. 1: Validas TOPWATER Compliance Method

3 Model-based Tool Qualification

Tool qualification is imposed by safety standards to ensure that tools can be used with confidence when developing safety-critical items/elements, see [In11], [In10], [RT11]. The main approach adopted by Validas AG for tool qualification is to test the tool in the same exact environment of the tool user (i.e. qualification by validation). This can be achieved using the so-called tool qualification kits (or QKits in short), which additionally can generate the work products (e.g. tool qualification report, tool safety manual) required by safety standards.

The idea of model-based tool qualification is to build a model for the tool qualification (i.e. containing features, known bugs, tests, etc.), that allows performing the available tests and analyzing their results to generate the required documentation and work products.

The tool chain analyzer [Va17] is a tool that supports the modeling of toolchains and qualification kits. It supports a tool qualification model with all required information for classification and qualification of tools, see [Wi12], [Sl12].

The metamodel of the TCA (see user manual) consists of the following elements (for the sake of clarity in the present paper, we use a simplified representation and omit the containment hierarchy):

- TOOLCHAIN: root element containing all other elements

- TOOL: represents a tool in the model
- VERSION: represents a version of the tool
- FEATURE: represents a feature of the tool
- ERROR: represents a potential error of a tool feature
- KNOWNBUG: represents a known mal function of the tool
- CHECK: user action to detect potential errors or real bugs
- RESTRICTION: user action to avoid potential errors or real bugs

Every element in the metamodel has attributes like NAME, DESCRIPTION; some also have other attributes (e.g. COMMENT, IMPACT, etc.) that have to be specified in concrete tool classification and qualification projects. Fig. 2 shows a typical representation of the metamodel.

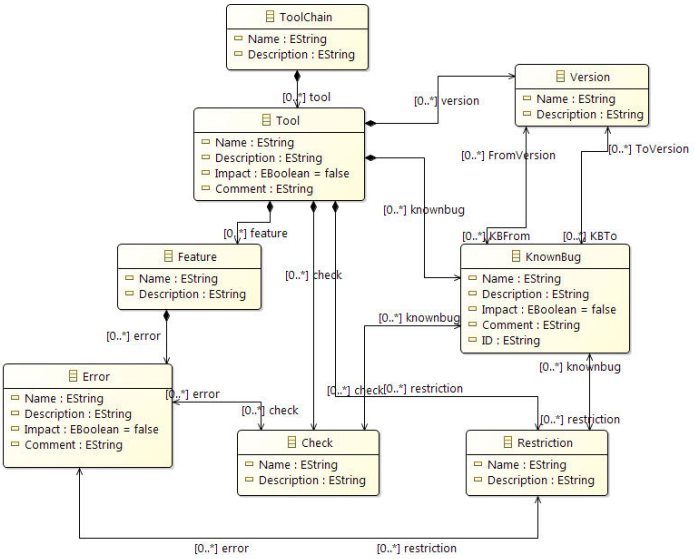

Fig. 2: Example Metamodel of Toolchains/Tools

The tool qualification requirements imposed by the relevant safety standards [In11], [In10] and [RT11] are similar in their nature. They mainly require a three phase approach:

- Classification of the tools
- Qualification of critical tools

- Safe usage of the tools according to safety manuals, which are based on results from the previous two phases (i.e. classification and qualification of tools)

4 Modeling the MetaModel used in Model-based QKit Development Processes

At the core of our approach is the modeling of model-based development processes and at the core of such modeling is the specification of the **MetaModel** being used.

The basis for specifying the **MetaModel** in AutoFOCUS3 is a simple enumeration data type called "**ModelSpecification**" with the enumerator names {**NotSet**, **Specified**, **NotRequired**}, corresponding to the three possible states every model element can have. The states of all model elements are initially set to **NotSet,** and once they are specified, their states change to **Specified**. Sometimes it might be undesirable or even not possible to specify an actual value for a model element. In these cases, the respective model elements are set to be in the **NotRequired** state. To clarify the last point, consider for instance the link TOOL_VERSION_TO of a KNOWN_BUG element, which defines the version in which the known bug was fixed. This attribute can be therefore **Specified** (in case the known bug is already fixed in a specific version) or **NotRequired** (in case the known-bug remains an open bug). The attribute TOOL_VERSION_FROM, which defines the version in which the bug is introduced, has to be specified in either case.

In general, for every class with name <ClName> with attributes <A1>, to <An> and links <L1> to <Lm> in the metamodel, there are three structure types modeled within AutoFOCUS3:

- `<ClName>Class:{Attributes:<ClName>Attributes, Links:<ClName>Links}`
- `<ClName>Attributes:{<A1>:ModelSpecification,…<An>:ModelSpecification}`
- `<ClName>Links:{<L1>:ModelSpecification,.. <Lm>:ModelSpecification}`

The type **MetaModel** is a structure data type consisting of components for each class:

- `MetaModel: {<C1>: <ClName>Class,…}`

The types, which are required for modeling the **MetaModel** from Fig. 2 are modeled in AutoFOCUS3 as shown in Fig. 3.

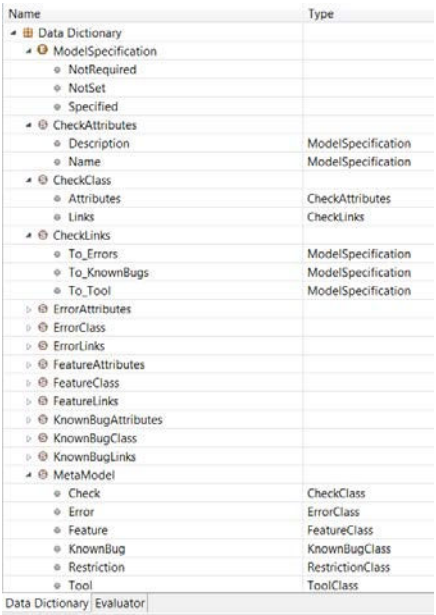

Fig. 3: Modeling the MetaModel Data Type in AutoFOCUS3

Having modelled the **MetaModel** in AutoFOCUS3, actions that define how it is being created and updated during the modeling processes are defined straightforwardly using function definition.

For example, the modeling of known bugs of a tool requires, as its input, a defined TOOL model with contained FEATURE model. The output would be an updated model, where KNOWN-BUG elements are added together with links to corresponding TOOLs and affected FEATUREs. Adding mitigations (CHECKs and RESTRICTIONs) and linking them to the KNOWN-BUGs further extends the model. Such procedures can be described using the attributes of the model elements that need to be described in each step: “TOOL.NAME” and “TOOL.VERSION” or “KNOWN-BUG.ID”. The predicate that checks whether known bugs can be modelled or not can be formulated in AutoFOCUS3 as follows:

```
readyForKBModeling(MetaModel:M) =
    M.Tool.Attributes.Name==Specified() &&
    M.Tool.Attributes.Description==Specified() &&
    M.Tool.Links.To_Features==Specified() &&
    M.Feature.Attributes.Name==Specified() &&
    M.Feature.Attributes.Description==Specified() &&
    M.Feature.Links.To_Tools==Specified() &&
```

```
        M.Version.Attributes.Name==Specified() &&
        M.Version.Attributes.Description==Specified() &&
        M.Version.Links.To_Tools==Specified()
```

And the update of the model by specifying known bugs can be done within AutoFOCUS3 in a functional programming style by defining the following function

```
updateKBModeling(MetaModel:M) =
return combineModels(M, addKnownBugClass( {
        Attributes: combineKnownBugAttributes(
                specifyKnownBugName(),
                specifyKnownBugDescription(),
                specifyKnownBugID()),
        Links: combineKnownBugLinks(
                specifyKnownBugLinkToTool(),
                specifyKnownBugLinkToVersion())}));
```

The helper function can be implemented easily using the definition

```
specifyKnownBugName:KnownBugAttributes = {
        Name:Specified,
        Description:UnSet,
        ID:UnSet}
```

Furthermore, in order to facilitate the generation of the *V&V Plan*, we mark the V&V actions using the keyword *Criterion*.

5 Modeling of Development Processes and Requirements

The process description starts from a high abstraction level describing the process with the customer interaction, see Fig. 4. When constructing a model-based QKit, the core activity is to build the corresponding model. This process is further detailed as depicted in Fig. 5, which also shows the main phases of building the QKit: Structure modeling, analysis modeling and test modeling.

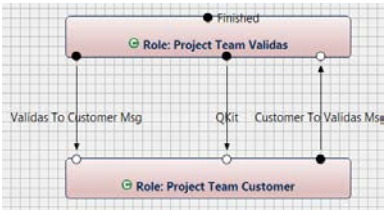

Fig. 4: Validas Interaction Process (High-Level) with Tool Providers

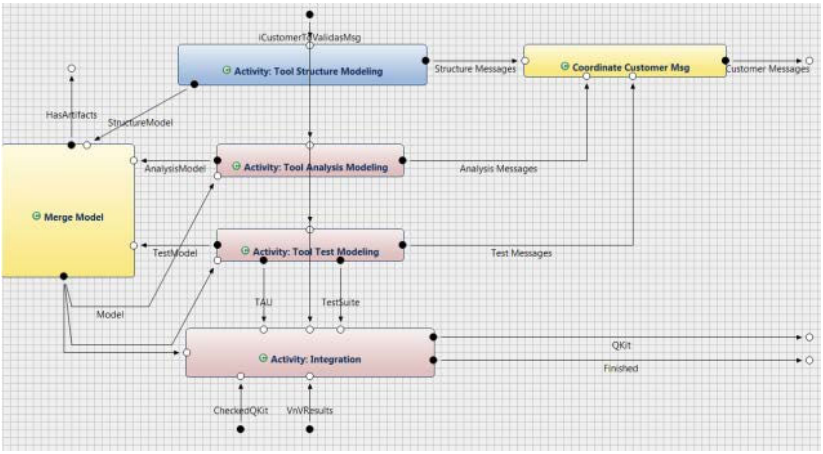

Fig. 5: Model Construction Main Process

The interface of the known bug modeling process (carried out in the analysis modeling phase) is described in Fig. 6. This example emphasizes the adequateness of the strongly-typed environment of AutoFOCUS3 to our modeling purposes. The type of the input model is **MetaModel** as detailed in Fig. 3. The name of the input model is "ToolFeatureModel", the name of the output model is "KBModel".

Fig. 6: Known-Bug modeling Process Interface

As has been already shown with the previous figures, AutoFOCUS3 supports a hierarchical description of processes (and requirements as will be seen later). The process of specifying known bugs can then be modelled with more details through a state transition diagram as depicted in Fig. 7 and Fig. 8 for the first part of the modeling process.

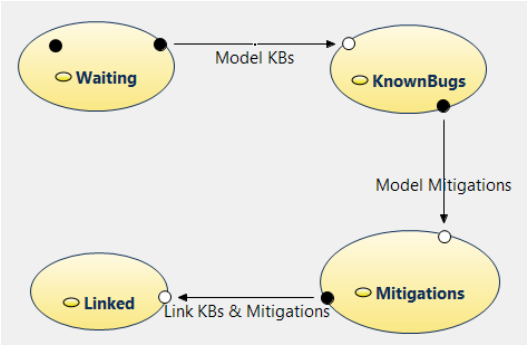

Fig. 7: Specification of Known Bug Modeling Behavior

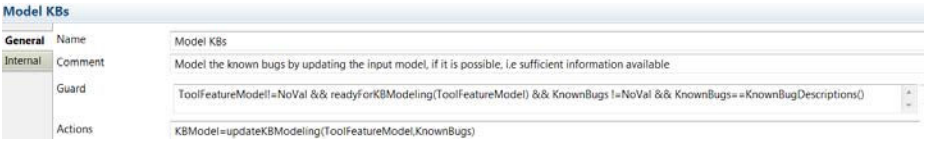

Fig. 8: Specification of the Transition link "Model KBs" from Fig. 7.

As an example of defining verification and validation activities for known bugs, we define the corresponding *Criterion* for known bugs modeling as depicted in Fig. 9. Note that this example shows the relevant V&V activities only partially.

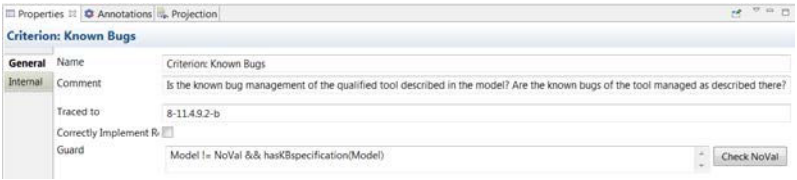

Fig. 9: Criterion Known Bugs

Requirements are typically structured in a hierarchic manner, which can be done in AutoFOCUS3 straightforwardly. It remains however to link/trace these requirements with the process modeling. To do so, we recognize that safety standard requirements are typically imposed on a) Processes and b) Products. For the purposes of certification and showing compliance, evidence should be provided that a) comply with requirements and that these processes have been actually followed/deployed for creating b). In our approach, we model the modeling activities/actions themselves including V&V activities that can be seen as a checklist for the concrete project. As such, every requirement has at least two traces in the processes model:

1. One to a process activity that describes it
2. Another to a V&V activity that checks it on the concrete example.

By splitting the requirements into these two parts, we can demonstrate that our process satisfies the requirements, provided that for each project the V&V activities are performed successfully. The concrete projects do not need to bother on the standard compliance, but have only their concrete checklists to perform.

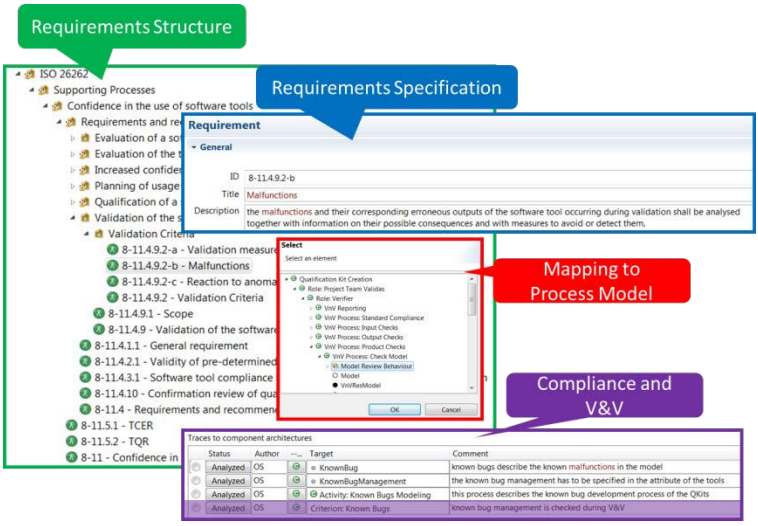

Fig. 10: Requirements Management and V&V Activities

Fig. 10 shows some examples of how the requirements from the safety standards are managed. The highlighted trace to the “Criterion: Known Bugs” is a trace to the V&V activity. The way in which this Criterion is defined is already discussed (and shown in Fig. 9).

6 Example: Certification of Tool Qualification Processes

Validas has applied (and validated) the proposed approach on the example of tool qualification as has been detailed in previous sections. Validas claims to build ISO 26262 and IEC 61508 compliant tool qualification kits using a model-based qualification process. The Validas process for building qualification kits has been modeled within AutoFOCUS3 and it has been shown to satisfy all 120 requirements from ISO 26262 and IEC 61508 for tool qualification. 13 additional requirements from Validas (functional and quality requirements for the model-based QKit) have been satisfied using a process description consisting of over 1150 element describing the process and the interaction with the customer. Every Criterion (V&V Check) consists of several questions, separated by “?”. In total over 250 (simple and concrete) questions in the criteria have to be answered for each QKit to pass the V&V.

According to the Method described in Section 2 the relevant documents have been generated.

Throughout the certification process regarding Validas process, TÜV requested not only the compliance with the tool qualification requirements, but also some general requirements on the management of functional safety. Those requirements could be easily satisfied by filling out the required Excel checklists and providing the evidences from the general Validas processes.

The main requirements have been successfully certified based on the generated documents: Validas Qualification Method (including the generated, formal process description), compliance report (generated as described) and V&V Plan (generated from the model as described).

7 Summary

We have presented a general two-step compliance approach that builds upon formalizing and modeling arbitrary requirements and model processes. In the first step, compliance to safety standards of the general deployed methodology is shown and compliance report is generated, whereas in the latter step, compliance of concrete projects that make use of that general methodology is checked against a V&V plan. Both the compliance report and the V&V plan are automatically generated from the constructed model. The method has been applied successfully within our certification efforts of the Validas model-based qualification kit construction processes with TÜV SÜD, according to ISO 26262 and IEC 61508.

8 Acknowledgments

This work has been supported in part by the German Federal Ministry of Research and Education (BMBF) within the project TOPWATER (ZIM) under research grant ZF41611701BZ5. The authors would like to thank Joachim Schramm for the fruitful and stimulating discussion within the TOPWATER project and Thomas Escherle for proof reading and formatting the paper.

9 References

[In10] International Electrotechnical Commission: IEC 61508, Functional safety of electrical/electronic/programmable electronic safety-related systems, Edition 2.0, 2010.

[RT11] RTCA: DO-330: Software Tool Qualification Considerations 1st Edition, 2011.

[In11] International Organization for Standardization: ISO 26262 Road Vehicles –Functional safety–. 1st Edition, 2011.

[Sl12] Slotosch, Oscar: Model-Based Tool Qualification - The Roadmap of Eclipse towards Tool Qualification – opencert, 2012.

[Wi12] Wildmoser, Martin; Philipps, Jan; Jeschull, Reinhard; Slotosch, Oscar; Zalman, Rafael: ISO 26262 - Tool Chain Analysis Reduces Tool Qualification Costs. In SAFECOMP 2012, 2012.

[Va17] Validas AG: Tool Chain Analyzer Tool, can be downloaded from www.validas.de/TCA.html, 2017.