Tool-Support to Foster Model-based Requirements Engineering for Cyber-Physical Systems

Kevin Keller[1], Adrian Neubauer[2], Jennifer Brings[3] and Marian Daun[4]

Abstract: Requirements modeling is known not only as a technique for documenting requirements but also for eliciting requirements from and discussing requirements with stakeholders. Modeling tools used in requirements engineering range from simple paper sketches to sophisticated model editors that allow for code generation from requirements models. While the former can be used by virtually anybody, tools that provide functionalities beyond creating simple drawings are based on underlying formalisms, which makes such tools not only unable to accommodate even the smallest deviations from the underlying modeling principles but also often unintuitive when it comes to the creation of simple diagrams. To bridge this divide, we suggest developing Visio extensions that extend Visio's user-friendly diagramming functionalities with custom-made shapes and code/model generation and verification functionalities as needed. This paper presents a solution for the SPES XT modeling framework that supports the model-based development of embedded and cyber-physical systems by providing shapes and model-generation and verification capabilities to aid developing systems according to the SPES XT modeling framework. The extension has proven itself valuable in university education and industry training, as it allows learners to easily create and verify diagrams that conform to the SPES approach.

Keywords: Model-based engineering, model editor, model verification, Microsoft Visio, tool support

1 Introduction

Modeling requirements is an important activity during the requirements engineering process, as it supports elicitation by facilitating communication between stakeholders and engineers and documentation of requirements [Po12]. There are different possibilities to create diagrams, e.g., with computer-aided tools or by hand. Hand-drawn models (including those created with drawing tools like PowerPoint, Visio, etc.) are fairly easy to create and offer a lot of freedom to express the desired concepts [Sh05] while dedicated modeling tools (e.g., Enterprise Architect) allow for automated verification and code generation and thus support further development activities [Sp16].

[1] University of Duisburg-Essen, paluno-The Ruhr Institute for Software Technology, Gerlingstraße 16, DE-45127 Essen, kevin.keller@paluno.uni-due.de
[2] University of Duisburg-Essen, paluno-The Ruhr Institute for Software Technology, Gerlingstraße 16, DE-45127 Essen, adrian.neubauer@paluno.uni-due.de
[3]University of Duisburg-Essen, paluno-The Ruhr Institute for Software Technology, Gerlingstraße 16, DE-45127 Essen, jennifer.brings@paluno.uni-due.de
[4]University of Duisburg-Essen, paluno-The Ruhr Institute for Software Technology, Gerlingstraße 16, DE-45127 Essen, marian.daun@paluno.uni-due.de

Consequently, both ways have their advantages but also disadvantages. Drawing tools allow for customizing figures and creating domain specific modeling languages easily, but offer no code generation and verification of the created diagrams. In addition, often only a few basic shapes, such as rectangles and ellipses are available out of the box. Dedicated model editors exist for a large number of standardized modeling languages and often offer the possibilities to verify diagrams and generate code artifacts from the created diagrams (e.g., [Be12]). However, apart from often being expensive (at least for small and medium sized companies), these kinds of tools usually require intensive training due to their large set of features and often unintuitive use [Am06]. Another disadvantage of model editors is their limitation to pre-defined modeling languages. While some allow the definition of other modeling languages, this usually requires the manual creation of profiles and extensions, which can be quite laborious (e.g., [Sp16]).

In this paper, we propose to bridge the gap between both approaches by customizing common off the shelf software. Namely, we propose to create a customized model editor with integrated formal methods using Microsoft Visio. This allows not only to tailor tool support for requirements elicitation but to integrate model-based requirements engineering seamlessly into a continuous model-based engineering framework. Specifically, we show how to use extension, stencils and hyperlinked shapes to adapt Microsoft Visio in such a way that it can serve as model editor for requirements engineering models which are in accordance with the SPES XT modeling framework [Bö16]. Furthermore, we show that automated methods such as model verification techniques, which were defined to ensure consistency among the different models of the SPES XT modeling framework can be implemented and applied using the approach. Hence, the resulting tool support cannot only serve as model editor that allows documenting requirements in model-based fashion as foreseen by the SPES XT modeling framework, but to also allow implementation of advanced automated techniques, which have been defined for diagrams and models of the SPES XT modeling framework (cf. [Po16]). Thereby, we show that this approach combines the easy-to-use nature of diagramming tools and the powerful capabilities of dedicated model editors.

The paper is structured as follows: Section 2 introduces the SPES XT modeling framework for which we developed a Visio-extension. In Section 3, the shapes and the functions of the extension are presented. Subsequently, a brief overview of related approaches for customizing model editors for specific model generations and model verifications is given in Section 4. Finally, Section 5 concludes the paper and discusses future work.

2 The SPES XT Modeling Framework

The SPES XT modeling framework [Bö16] is an artifact-centric approach to continuous model-based engineering of embedded systems. The SPES XT modeling framework defines four viewpoints and several granularity levels as can be seen from Fig. 1. The

granularity levels allow reducing complexity by supporting the decomposition of the system into subsystems, while the viewpoints allow reducing complexity by allowing for separation of concerns. The four viewpoints are:

- The *requirements viewpoint* addresses the model-based documentation of requirements and the separation of the system from system context. To this end the requirements viewpoint supports the creation of context, goal, scenario, and requirements models.
- The *functional viewpoint* addresses the functional system specifications (derived from the requirements viewpoint) and modeling of system functions.
- The *logical viewpoint* addresses the decomposition of the functional building blocks into logical sub-systems with the main goal of delivering a logical architecture i.e. functions are clustered into logical components and
- The *technical viewpoint* addresses the mapping of logical components to the technical realization and their hardware and software components.

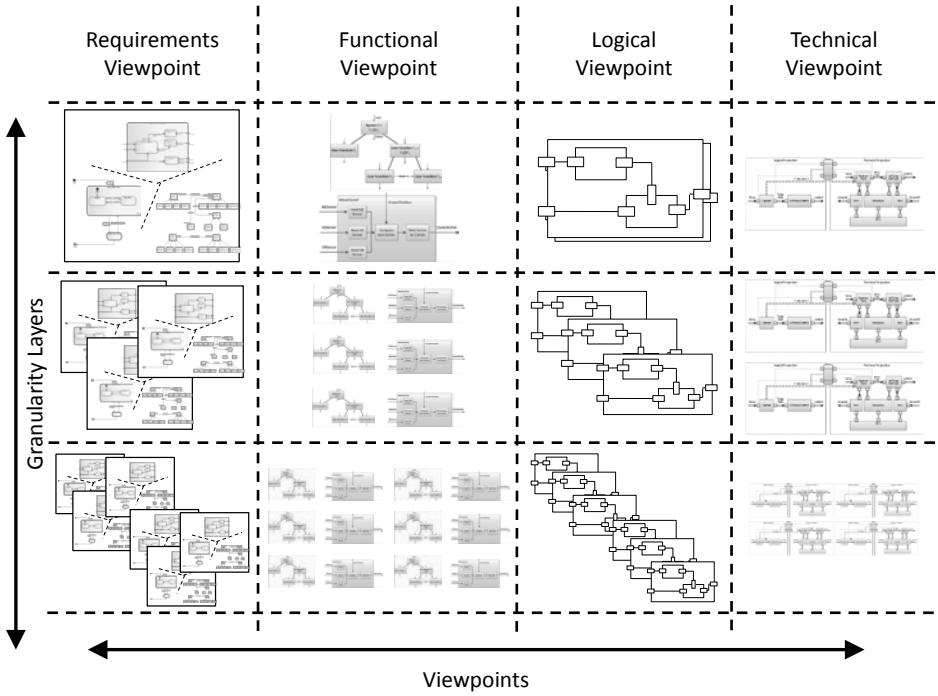

Fig. 1: SPES XT modeling framework [Bö16]

3 The SPES Visio Extension

Microsoft Visio offers a model editor for standardized modeling languages. In addition, by using stencils and associated shapes, new modeling languages can be created. Furthermore, Visio provides an API to write specific functions for model generation, verification, and import and export. In this section, we will show how to make use of these functionalities to create a Visio extension which provides support for the SPES XT modeling framework by ensuring conformance to the defined modeling languages and which implements defined methodological building blocks of the SPES XT modeling framework.

Visio provides a selection of pre-installed stencils for various modeling languages such as UML or BPMN and layout functions for creating diagrams. The Visio model editor consists (as shown in Fig. 2) of the shape view, the opened drawing sheet, the toolbar and existent drawing sheets for the document. In the shape view (left side), the installed stencils can be viewed and selected to be used in the drawing sheet. It is also possible to model on different sheets, which are located at the bottom of the editor window. The toolbar (top) provides various functionalities such as changing color, size, or alignment of model elements. To provide support for modeling languages not included in Visio, new stencils can be defined.

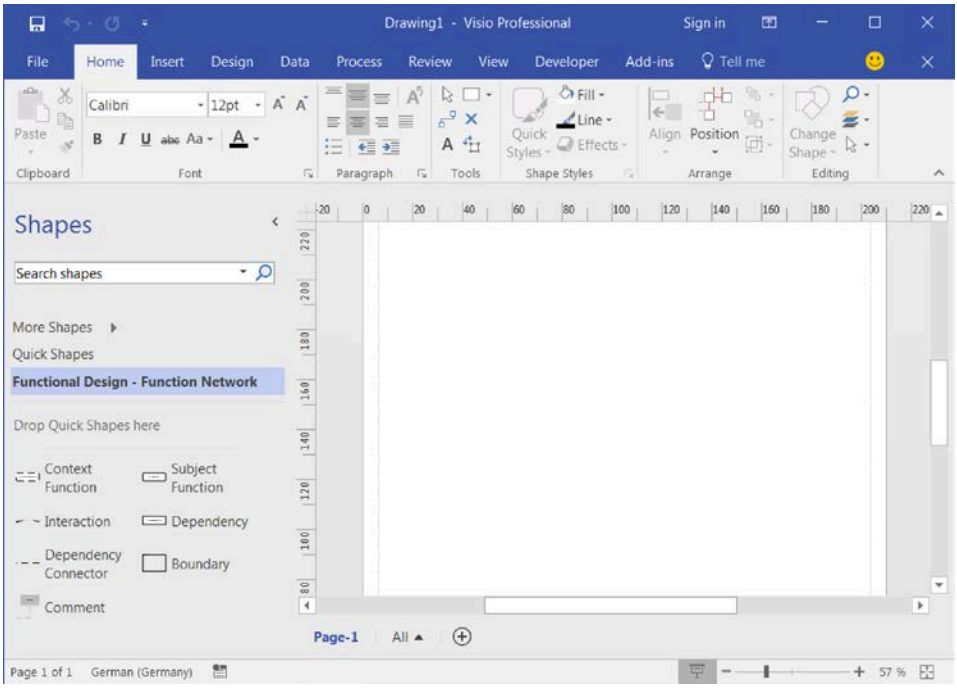

Fig. 2: Model Editor

Beyond the definition of stencils, Visio extensions can add new functionalities to the application and provide them to the user via the toolbar. For model generation and formalization, the necessary functionality has been added via the Visio API and SDK. It offers the possibility to control the Visio application in a programmatic way and all interaction possibilities that are available to the user as well.

3.1 Model Generation

The SPES Visio extension[5] enables users to create diagrams according to the SPES XT modeling framework and helps them navigate between the various viewpoints and granularity layers. As the system or a subsystem is further decomposed, a tree hierarchy of the system is automatically defined and each element in the tree is linked to the sheet of the related viewpoints. An example of an automatically generated decomposition hierarchy is shown in Fig. 3. In a first step, the user specifies the name *Adaptive Cruise Control* as the system under development. Next, the user can navigate to the logical viewpoint and select specific model elements to decompose the system into sub systems. In this example, the system *Adaptive Cruise Control* is divided into the sub-systems *System Management*, *Speed Adjustment* and *Data Processing*.

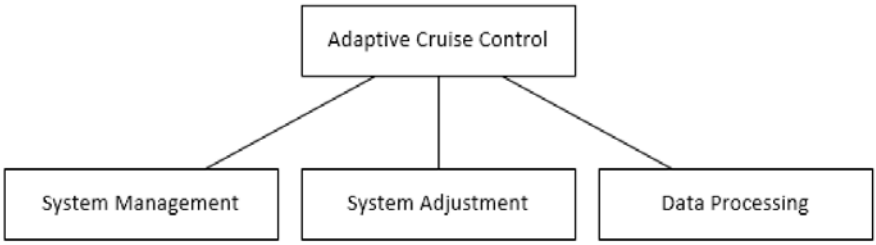

Fig. 3: Generated Components in a Tree Hierarchy

Each node is linked to a corresponding template (shown in Fig. 4 for the adaptive cruise control node). The template contains links to models from the various viewpoints (however, tracing for inter-model changes is not yet supported). The root node and the corresponding template is defined initially for each SPES modeling project. Subsystems and their corresponding templates are automatically added when the user designates a certain model element as a subsystem.

Upon opening an empty file for a certain kind of model (e.g., a scenario model) the corresponding stencils are displayed automatically.

Additionally, several functions that automatically generate skeletons from information already modelled are implemented. For example, for each MSC reference in the

[5] The SPES Visio extension as well as its source code are available https://spes-tool.paluno.uni-due.de/

modelled high-level Message Sequence Chart an empty sheet with the name of the referenced object will be generated and the shapes of the basic Message Sequence Chart will be displayed.[6]

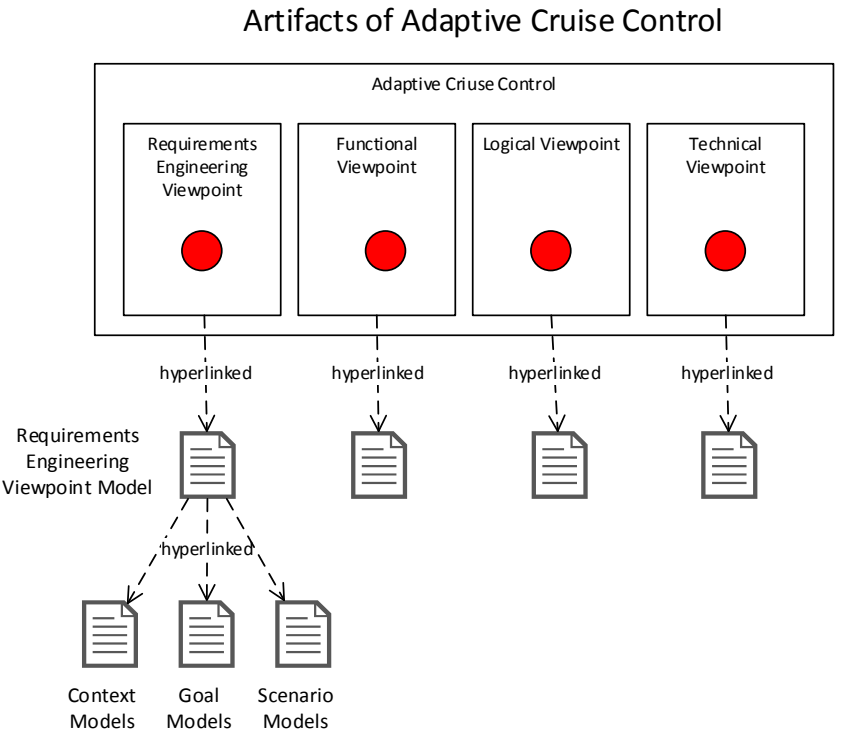

Fig. 4: Generated Viewpoints

3.2 Custom-Made SPES Stencils

For the SPES XT modeling framework several modeling languages are required to define specific requirements models of the system under development (cf. [Da12]). For the artifacts of the different viewpoints standardized model languages such as UML class diagrams [OM15] or the ITU User Requirements Notation (including Goal-oriented Requirements Language and Use-Case Maps) [IT12] are used. Most of the modeling languages exist as stencils for Visio. For the modeling languages that do not exist as stencils, Visio offers the possibility to create customized shapes and bundle them in a stencil. This feature is required for several in the projects SPES2020 and SPES XT

[6] For more information about the ITU Message Sequence Chart Language please refer to [IT11]

defined modeling languages. For example, for creating context of knowledge models [Da14] a customized notation must be created in form of a template (called stencil). The stencil for the context of knowledge and an exemplary context of knowledge model for an adaptive cruise control system is shown in Fig. 5. On the left side, the application offers the elements for modeling the context of knowledge. On the right side of the window, it shows the model and its elements, which describe which knowledge sources provide information for or constrain the development of the adaptive cruise control system. For instance, the document *Regional Traffic Codes* constrains the possible behavior of the adaptive cruise control to be developed.

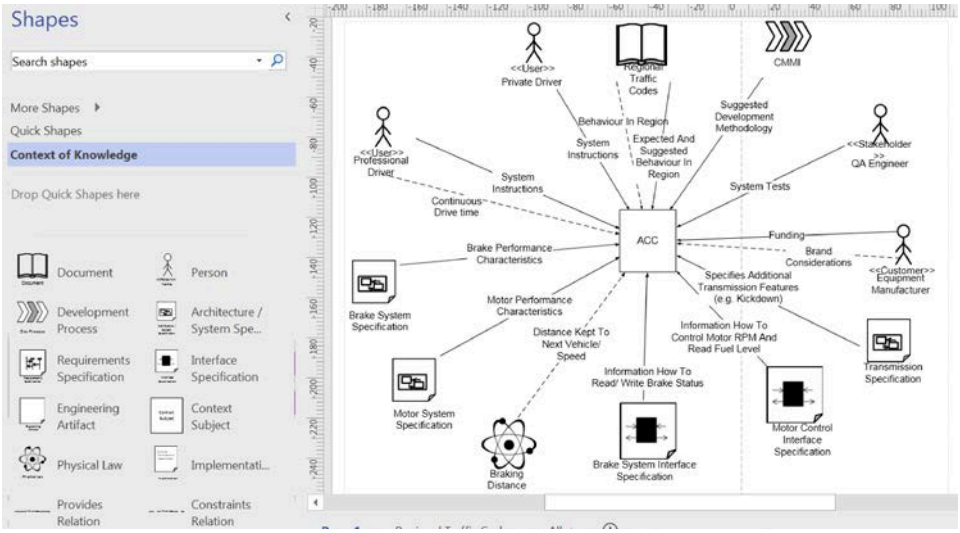

Fig. 5: Example of the created stencil for the Context of Knowledge

3.3 Model Verification

The extension also offers support for model verification. In particular, it implements a variety of model verification algorithms to check the models for syntactic correctness. The checks are realized with validation rules similar to UML constraints. For example, in the functional viewpoint a function network [Al16] can be checked for syntactic correctness, and the associated interface automata [AH01] can be checked with regard to completeness. Therefore, the incoming and outgoing messages are compared. Missed model elements can be detected and automatically added to the model. Fig. 6 illustrates an algorithm using path checking to ensure all elements are reachable in directional models.

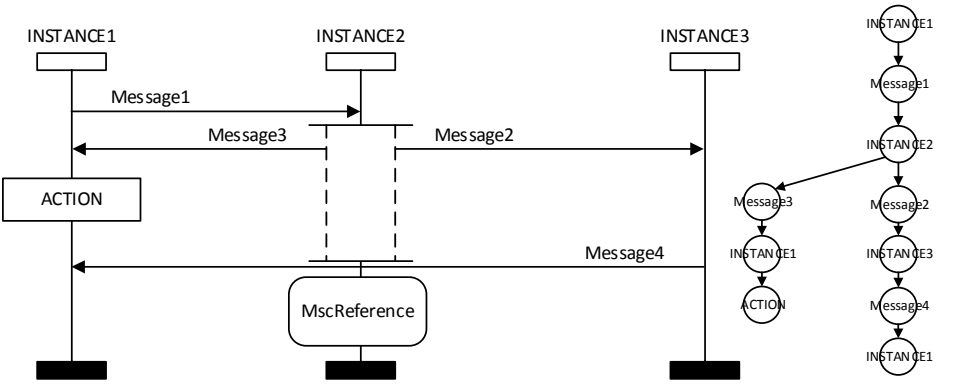

Fig. 6: Example BMSC and associated path-tree

The example in Fig. 6 displays a Basic Message Sequence Chart (left) and the tree the extension creates by traversing the model from element to element. The extension checks reachability by traversing the tree and identifying the elements not contained in it. In this example, the model on the left contains an unreachable element ("MscReference", bottom left) and therefore it does not exist in the traversing tree (right). This missing element then notifies a verification violation.

4 Related Work

There are several model editors for formalizations and model generation of models and textual descriptions. For instance, FormulaBuilder [Jö06] is a tool for graph-based modeling and generation of formulae. Therefore, the formal specification of properties describes the requirements of a system. This is different to the SPES Visio extension, because of the focus. FormulaBuilder deals with formalization of models, in contrast to the Visio extension focusses on the generation of models for requirements engineering. A similar approach is followed by the tool from [EW02]. This tool supports UML activity diagrams as workflow models and exports the model into a format that a model checker can verify against requirements. Here, the focus is more on model verification than on model generation. [KK15] developed DOREF an approach for the generation of models and to check them against requirements in real-time. The focus here, is to generate models from formal text and to perform verification. [Fo15] presented an approach for the Papyrus UML editor. Therefore, a systematic generation of a diagram definition is used to generate a modeling language tool that guarantees compliance to standard notation. [Ca11] presented a plugin called OthelloPlay for the support of formalization of textual requirements. In addition, it simplifies the use of formal techniques for the verification of requirements. OthelloPlay can be used as a Microsoft Word Add-In. The difference to the SPES Visio extension is the generation of a formal model of textual requirements. SCStudio [Be12] is an extension for Visio which allows modeling and verification for Message Sequence Charts. However, as it only supports

ITU MSCs it does not integrate with other modeling languages and methodological approaches as needed by the SPES XT modeling framework.

5 Conclusion and Future Work

It is hard to provide appropriate tool-support for model-based requirements engineering, which seamlessly integrates into further model-based development activities. As requirements engineering is a conceptual activity employing creativity and lots of stakeholder communication, a modeling tool needs to be lightweight to adequately support the requirements engineers in their day to day work. This contradicts the approach of most professional tool-suites for model-based engineering, which in part rely on a formal specification of requirements and models, which the requirements engineer typically is not able to provide at early stages. In contrast, easy to use standard modeling editors commonly lack the potential for formalization and application of automated methods needed in the engineering of complex cyber-physical systems. In this paper, we proposed an approach making use of both concepts leading to a lightweight model editor, which can easily be tailored to fit for automated approaches as well. We presented an MS Visio extension for the modeling of cyber-physical system using the SPES XT modeling framework. The extension shows, that Visio, in combination with its API, can be used for modeling requirements through several viewpoints. Additionally, the extension for Visio allows formal checks of the created models, such as function networks or Message Sequence Charts. Furthermore, Visio offers the possibility to create drawing elements for domain specific modeling languages. The defined shapes can be used for drawing conceptual models and check them. Hence, models can be generated on the needed level of formalization and automated techniques can be applied accordingly. Future work will have to deal with a thorough evaluation, particularly, the approach shall be evaluated with respect to other modeling applications and needs to consider further extensions for verifications of other modeling languages.

Acknowledgment. This research has partly been funded by the German federal ministry for education and research under grant no. 01IS15058C.

References

[AH01] de Alfaro, L. and Henzinger, T.A. 2001. Interface Automata. *Proceedings of the 8th European Software Engineering Conference Held Jointly with 9th ACM SIGSOFT International Symposium on Foundations of Software Engineering* (New York, NY, USA, 2001), 109–120.

[Al16] Albers, K., Beck, S., Büker, M., Daun, M., MacGregor, J., Salmon, A., Weber, R. and Weyer, T. 2016. System Function Networks. *Advanced Model-Based Engineering of Embedded Systems*. 119–144.

[Am06] Amyot, D., Farah, H. and Roy, J.-F. 2006. Evaluation of development tools for

domain-specific modeling languages. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 4320 LNCS, (2006), 183–197. DOI:https://doi.org/10.1007/11951148_12.

[Be12] Bezdeka, M., Bouda, O., Korenciak, L., Madzin, M. and Reh'k, V. 2012. Sequence Chart Studio. *2012 12th International Conference on Application of Concurrency to System Design* (Jun. 2012), 148–153.

[Bö16] Böhm, W., Daun, M., Koutsoumpas, V., Vogelsang, A. and Weyer, T. 2016. SPES XT modeling framework. *Advanced Model-Based Engineering of Embedded Systems: Extensions of the SPES 2020 Methodology*. 29–42.

[Ca11] Cavada, R., Cimatti, A., Micheli, A., Roveri, M., Susi, A. and Tonetta, S. 2011. OthelloPlay - A plug-in based tool for requirement formalization and validation. (2011), 59.

[Da12] Daun, M., Tenbergen, B. and Weyer, T. 2012. Requirements Viewpoint. *Model-Based Engineering of Embedded Systems*.51–68.

[Da14] Daun, M., Brings, J., Tenbergen, B. and Weyer, T. 2014. On the model-based documentation of knowledge sources in the engineering of embedded systems. (2014), 67–76.

[EW02] Eshuis, R. and Wieringa, R. 2002. Verification support for workflow design with UML activity graphs. *Proceedings-International Conference on Software Engineering*. (2002), 166–176.

[Fo15] Fouche, A., Noyrit, F., Gerard, S. and Elaasar, M. 2015. Systematic generation of standard compliant tool support of diagrammatic modeling languages. (2015), 348–357.

[IT11] ITU 2011. Recommendation Z.120- Message Sequence Chart (MSC). *ITU-T*. (2011).

[IT12] ITU 2012. Recommendation Z.151 - User Requirements Notation (URN) Language Definition. *ITU-T*. (2012).

[Jö06] Jörges, S., Margaria, T. and Steffen, B. 2006. FormulaBuilder: A tool for graph-based modelling and generation of formulae. *Proceedings - International Conference on Software Engineering* (2006), 815–818.

[KK15] Kneer, F. and Kamsties, E. 2015. Model-based generation of a requirements monitor. (2015), 156–170.

[OM15] OMG 2015. Unified Modeling Language (OMG UML) Specification, version 2.5. *Technical Report*. (2015).

[Po12] Pohl, K., Hönninger, H., Achatz, R. and Broy, M. 2012. *Model-Based engineering of embedded systems: The spes 2020 methodology*.

[Po16] Pohl, K., Broy, M., Daembkes, H. and Hönninger, H. 2016. *Advanced model-based engineering of embedded systems: Extensions of the SPES 2020 methodology*.

[Sh05] Shumba, R. 2005. Usability of Rational Rose and Visio in a software engineering course. *SIGCSE Bulletin*. 37, 2 (2005), 107–110. DOI:https://doi.org/10.1145/1083431.1083475.

[Sp16] Sparx Systems 2016. Introduction to Enterprise Architect. (2016).