A Model-Based Workflow from Specification Until Validation of Timing Requirements in Embedded Software Systems

Arne Noyer, Padma Iyenghar,
Elke Pulvermueller
Institute for Software Engineering
University of Osnabrueck, Germany
anoyer@uni-osnabrueck.de, padmaiyenghar@ieee.org,
elke.pulvermüller@informatik.uni-osnabrueck.de

Joachim Engelhardt, Florian Pramme,
Gert Bikker
Institute for Distributed Systems
Ostfalia University of Applied Sciences
Wolfenbuettel, Germany
{jo.engelhardt, florian.pramme, g.bikker}@ostfalia.de

***Abstract*—In embedded software engineering, timing requirements are among the foremost non-functional requirements that have to be fulfilled. Therefore, there are specialized tools for analyzing and validating the timing behavior in embedded software. On the other hand, Model Driven Development (MDD) is considered as the next paradigm shift to address the increasing complexity in embedded software development. Despite this paradigm shift, it is advantageous to use specialized Requirements Management (RM) tools for managing requirements. Thus, it is intuitive to perceive that a workflow for collaborating with RM, MDD and timing validation tools is very useful. Nevertheless, such a workflow is still missing. This paper addresses those gaps and proposes an approach towards an integrated workflow for managing timing requirements in RM tools, specifying them in MDD tools and their validation in tools for timing analyses.**

Keywords*—*Timing Requirements; Model-Driven Development; Unified Modeling Language; Requirements Engineering; Requirements Interchange Format; Requirements Traceability

I. INTRODUCTION

Embedded systems have to provide more and more functions and therefore, their complexity is increasing. In order to master the increasing complexity in the development of embedded software, MDD is considered as a possible solution [1]. Thereby, the Unified Modeling Language (UML) [2] has already been established as a widely used industry standard. During the development of complex embedded software, it is further critical to ensure that the software fulfills its requirements. The requirements are usually managed in tools for Requirements Management (RM), such as Polarion [3], DOORS [4] and ProR [5]. In these tools, a subset of the requirements can be non-functional and/or affect the timing behavior of the system. To guarantee that every requirement is realized by a model element, links between them and a traceability of requirements are needed. Further, this allows for making impact analyses down to model elements, when requirements are changed.

In order to facilitate relationships between requirements and model elements, a collaboration between RM tools and MDD tools is desirable. There are already some solutions for interfacing between RM and MDD tools, but they are often proprietary for certain tools. Furthermore, they often only allow to create links between requirements and model elements in their own user interfaces. In this paper, an approach for a workflow is presented which addresses these gaps and uses the standardized Requirements Interchange Format (ReqIF) for interfacing between RM and MDD.

The workflow in this paper further proposes to interface between MDD tools and tools which are specialized for timing validation, such as SymTA/S [6]. In MDD, timing behavior can be further specified for model elements to fulfill related requirements. While timing behavior and elements which affect the timing behavior, can be defined, most MDD tools are unable to validate the timing behavior. Therefore, this paper also addresses this gap and therewith proposes a workflow from specification until validation of timing requirements.

The remaining of this paper is organized as follows. Section II presents state of the art techniques and related work. The approach of this paper for the workflow from specification until validation of timing requirements is presented in section III. Section IV concludes the paper.

II. STATE OF THE ART AND RELATED WORK

Since this paper presents a workflow for specification until validation of timing requirements, this section discusses related work pertaining to standards for exchanging requirements, approaches for traceability between RM and MDD, possibilities for describing timing behavior and methodologies for working with timing requirements.

A. Requirements Interchange Format (ReqIF)

In a typical client/supplier situation, it is often needed to exchange requirements between different partners. For instance, one partner is specifying the end user requirements and the other one is creating system requirements, which are satisfying the end user requirements. Therefore, many RM tools, such as DOORS [4], Polarion [3] and ProR [5] support the standardized Requirements Interchange Format (ReqIF) [7]. Inside ReqIF files, requirements are saved as *SpecObject*s with their attribute values and requirements types are saved as *SpecObjectType*s with attribute definitions. Links between requirements are saved as *SpecRelation*s. Most RM

tools provide powerful mechanisms for traceability between requirements.

B. Requirements Traceability between RM and MDD

In order to ensure that all requirements are fulfilled by model elements, there have to be relationships/links between model elements and requirements. In the UML domain, requirements can be managed by applying the SysML [8] profile and *Dependencys* can be created between model elements and requirements. The focus in this paper is on using UML for MDD, since it has a general approach for modeling software. Unfortunately, there is no general approach for exchanging (timing) requirements with UML models. Most existing approaches for interfacing and traceability between RM and UML are either proprietary for certain tools (e.g. Rational Rhapsody Gateway [9]) or have their own editors for traceability, which makes them less intuitive. This is also confirmed by a survey about traceability solutions in [10].

A general technology for tool integration is Open Services for Lifecycle Collaboration (OSLC), which allows that different tools access their data mutually. Unfortunately, OSLC is currently only supported by a few tools, such as DOORS Next Generation [11].

C. Describing Timing Behavior

The UML profile for *Modeling and Analysis of Real-Time and Embedded Systems* (MARTE) [12] enhances the UML with possibilities for describing timing behavior in the UML domain and is standardized by the Object Management Group (OMG) [13]. A detailed description of MARTE is available in [14]. Further, in the automotive domain, AUTOSAR 4 [15] has its own model for describing timing, which was influenced by the TIMMO (TIMing MOdel) [16] project.

There are also specialized tools for validating and analyzing timing behavior, such as MAST [17] and SymTA/S [6] [18]. Despite the fact that they have capabilities for analyzing the timing behavior early during development, they are often used very late during the development of a software and when actual timing errors occur.

D. Methodologies for Working with Timing Requirements

Some approaches for model-based methodologies for working with timing requirements are discussed in [19] and [20]. Noticeable for both is that they are using UML [2] for MDD and MARTE [12] for describing real-time behavior. A traceability between model elements and requirements is realized by using a SysML [8] based UML profile for managing requirements and by creating relationships between them. However, it is not discussed how RM tools can be used in these methodologies and how requirements from RM tools can be exchanged with UML models in order to enable a traceability between UML elements and requirements from RM tools. Further, some related work towards validating timing behavior at design level is available in [21] and [22].

E. Summary

While there are already approaches for different aspects of the proposed workflow, there is a gap between RM and MDD. Existing solutions for requirements traceability in MDD are either proprietary or additional user interfaces have to be used. The workflow in this paper addresses this gap by using the standardized ReqIF [7] for interfacing between RM and MDD (in both directions), which also allows developers to create links between model elements and requirements directly in MDD tools. Further, traceability information down to model elements is made available in RM tools, which allows requirements managers to make analyses like coverage analyses in their RM tools. In addition, the timing behavior in MDD is exchanged with tools for timing validation and annotated timing behavior in MDD is made visible in RM.

III. MODEL-BASED WORKFLOW FOR TIMING REQUIREMENTS

Figure 1 illustrates the proposed workflow for developing embedded software systems with timing requirements. In this workflow, RM tools are used for specifying requirements. Non-functional requirements such as timing requirements can be specified in RM tools to a certain extent as well. Then, the RM tool is exchanging information about the requirements with a MDD (UML) tool by using ReqIF [7]. Inside the MDD tool, representations for requirements are created, which allow developers to create relationships between model elements and requirements. Afterwards, information about the relationships and representations for linked model elements can be transferred back to RM tools by using ReqIF, again. In addition, when updated requirements are re-transfered from RM tools to MDD tools, changed requirements are highlighted. Thus, impact analyses can be made in MDD tools.

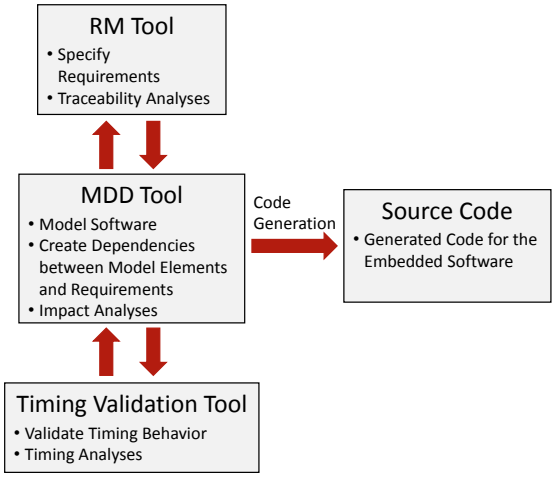

Fig. 1. Proposed Workflow for Working with Timing Requirements in MDD

In MDD, timing behavior can further be specified for model elements. For instance, UML profiles such as MARTE [12] enhance the UML with the capabilities for this. Thereby, *Stereotypes* are assigned to model elements to alter their semantics and *Tagged Values* are used for annotating timing behavior. Then, information about the timing behavior is exchanged with specialized tools for validating timing behavior. After validating the behavior, validation results are transfered back to the MDD tool. In addition, the source code for the developed embedded software system can be generated directly from the MDD tool. In [23], there is more information about (tool independent) code generation.

A. Interfacing between RM and MDD Tools using ReqIF

The process for exchanging requirements between RM tools and MDD tools is visualized in figure 2. In this figure, the RM domain is illustrated on the left side and the MDD domain is illustrated on the right side. As discussed above, ReqIF [7] is used for interfacing between the domains.

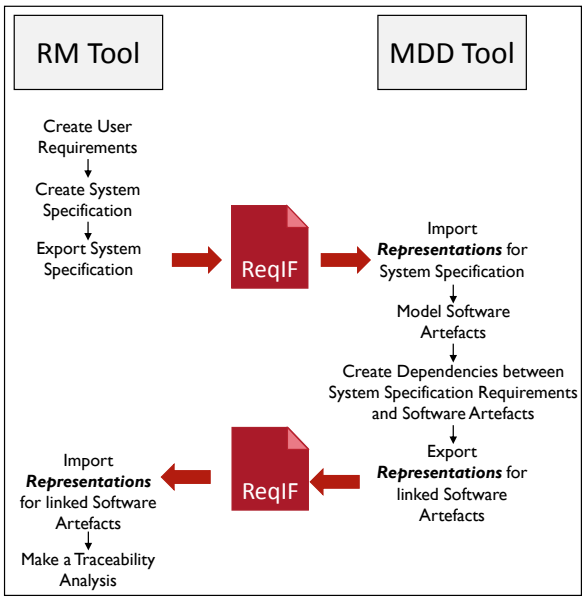

Fig. 2. Interfacing between RM Tools and MDD Tools using ReqIF

While there can be different refinement layers of requirements in RM, only those should be exported to a ReqIF file, which are relevant for software developers. Then, *representations* of these requirements can be imported into a MDD tool. In the UML domain, this can be realized by creating SysML [8] requirements, as the UML does not natively support requirements. Requirements types can be represented by using *Stereotypes* and assigning them to requirements. For instance, there could be Stereotypes like *«System Specification»*, *«Non-functional requirement»* or even *«Timing requirement»*. For requirements attributes such as *priority*, which may have been defined in a RM tool for a requirements type, *Tagged Values* can be used in UML. Thereby, all the exported information about requirements is made visible to developers in MDD.

Inside the MDD tool, relationships can be created between model elements and *representations* of requirements. The requirements are called *representations* in MDD, because they should only be used for creating relationships to them. Thereby, synchronization problems can be avoided, when it is assumed that requirements are not changed in the MDD tool. In order to create relationships, *Dependencies* can be used in UML. Further, stereotypes can be assigned to them such as *«trace»* or *«satisfies»* for different kinds of relationships.

Afterwards, the model can be analyzed by an algorithm, which finds model elements that are linked to requirements with *Dependencies*. For these model elements, *representations* (*SpecObjects*) are created inside the ReqIF file and they are linked to requirements in it. In order to represent model elements in ReqIF, a new requirements type (*SpecObjectType* in ReqIF) like *Software Artifact* can be created and assigned to them. Further, attributes are created for storing information about what kind of software artifact is represented (i.e. UML class) and the identifier in the MDD tool. In addition, attributes are created for each *Tagged Value*, which may contain information such as annotated timing behavior (e.g. when using MARTE [12]).

The information about model elements and their links to requirements can be re-imported into the RM tool. Thus, requirements managers are able to make traceability analyses such as coverage analyses down to model elements in their RM tool. In addition, annotated timing behavior is made visible in the RM tool.

Further, when re-synchronizing requirements with a MDD tool, existing representations of requirements are compared to the requirements in the ReqIF file. Thereby, it can be identified if requirements have been changed or deleted. Those requirements are highlighted in the MDD tool. In the UML domain, Stereotypes *«Changed»* and *«Deleted»* are created and assigned to requirements for this purpose. Thereby, impact analyses can be made in MDD tools and developers can check, which model elements have to be adjusted according to changed requirements. Information about a prototype, which realizes this approach for interfacing between RM and MDD using ReqIF, is available in [24].

B. Modeling of Timing Requirements and their Validation

The workflow proposed in section III-A can be used to link model elements with timing and other requirements. However, in MDD, timing requirements can often be further specified for model elements. For an operation inside the model, an execution time limit may be defined, as it is possible with MARTE [12] in UML.

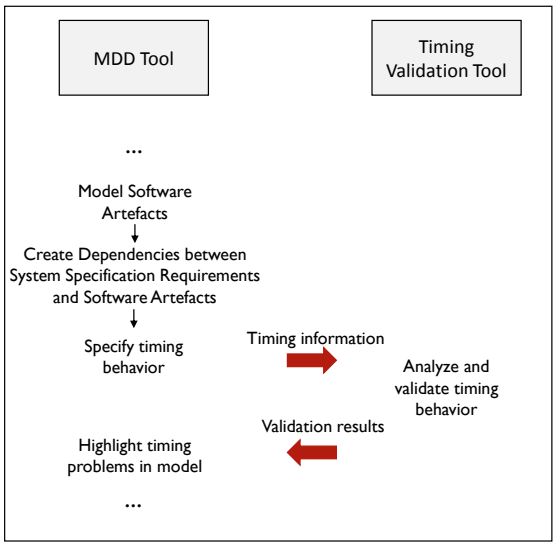

Fig. 3. Interfacing between MDD Tools and Tools for Timing Validation

Unfortunately, most MDD tools do not allow to analyze or validate timing behavior in detail. This is possible by using specialized tools for timing validation (i.e. SymTA/S [6]), which have their own data models. In order to facilitate validating timing behavior directly while modeling an embedded

software system, information must be exchanged with MDD (UML) tools and such timing validation tools.

Figure 3 extends the workflow in figure 2. It shows that after creating *Dependencies* between (timing) requirements and software artifacts, the timing behavior is further specified in the MDD tool. Afterwards, information about the timing behavior is exported to a tool, which is specialized on describing and validating timing behavior. Then, the information about the validation is transfered back to the MDD tool. Thereby, in order to ensure a complete traceability, the exchanged elements must have traceability information about their representations in the MDD tool, like unique identifiers. Afterwards, the model elements in the MDD tool can be enriched with information about the validation results. In the UML domain, this can be done by using *Tagged Values* and *Stereotypes*, again. In contrast to ReqIF for exchanging requirements, there is no standardized format for exchanging timing information. Therefore, a custom format can be used or the data can be exchanged by using a UML model with a profile (MARTE), which is persisted in XMI.

IV. CONCLUSION AND FUTURE WORK

In this paper, a workflow from the specification until the validation of timing requirements is discussed. The workflow enables to use RM tools for requirements management, MDD tools (UML) for software development and timing validation tools for validating timing behavior. Requirements are exchanged between RM and MDD by using the standardized format ReqIF [7]. By this, model elements and annotated timing behavior can be linked to requirements in MDD tools and the traceability information can be transfered back to RM tools. Thereby, traceability analyses like a coverage analyses can be made inside RM tools and it is possible to keep track of which timing behavior resulted from which requirements.

In addition, it is desirable to be able to validate timing behavior directly while developing a software, and not only when timing errors occur. Therefore, the workflow discusses how to exchange information about the timing behavior between MDD tools and specialized tools for timing validation.

Some future work directions are: to include measured time values in the workflow, to continue with developing a prototype and experimental evaluation, and to consider embedded systems modeled with heterogeneous modeling domains [25].

ACKNOWLEDGMENT

This project is supported by a grant from the Federal ministry of Economics and Technology (BMWi), Germany. This project work is carried out in cooperation with Willert Software Tools GmbH, Symtavision GmbH, Ostfalia University of Applied Sciences and the University of Osnabrueck.

REFERENCES

[1] R. B. France, S. Ghosh, T. Dinh-Trong, and A. Solberg, "Model-driven development using UML 2.0: promises and pitfalls," *Computer*, vol. 39, pp. 59 – 66, 02 2006.

[2] Object Management Group, "Unified Modeling Language Specification," August 2013. [Online]. Available: http://www.uml.org/

[3] Polarion Software, "Polarion alm," 2014. [Online]. Available: https://www.polarion.com/products/alm/index.php

[4] IBM, "Rational DOORS Website," 2014. [Online]. Available: http://www-03.ibm.com/software/products/en/ratidoor

[5] Eclipse Foundation, "ProR Requirements Engineering Platform," 2015. [Online]. Available: http://eclipse.org/rmf/pror/

[6] Symtavision GmbH, "Symta/s & traceanalyzer," 2015. [Online]. Available: https://www.symtavision.com/products/symtas-traceanalyzer/

[7] Object Management Group, "Requirements Interchange Format Specification 1.1," 2013. [Online]. Available: http://www.omg.org/spec/ReqIF

[8] Object Managemeng Group (OMG), "SysML Specification 1.3," 2012. [Online]. Available: http://www.omgsysml.org/

[9] IBM, "Rational Rhapsody Gateway Add On User Manual," 2010. [Online]. Available: http://pic.dhe.ibm.com/infocenter/rhaphlp/v7r5/topic/com.ibm.rhapsody.oem.pdf.doc/pdf/manual.pdf

[10] S. Winkler and J. von Pilgrim, "A survey of traceability in requirements engineering and model-driven development," *Software & Systems Modeling*, vol. 9, no. 4, pp. 529–565, 2010.

[11] IBM, "Rational DOORS Next Generation Website," 2015. [Online]. Available: http://www-03.ibm.com/software/products/en/ratidoorng/

[12] Object Management Group, "MARTE Specification 1.1," 2011. [Online]. Available: http://www.omg.org/omgmarte/Specification.htm

[13] Object Management Group (OMG), "OMG Website," 2014. [Online]. Available: http://www.omg.org

[14] B. Selic and S. Gerard, *Modeling and Analysis of Real-Time and Embedded Systems with UML and MARTE - Developing Cyber-Physical Systems*. Morgan Kaufmann Publishers, 2014.

[15] AUTOSAR, "AUTomotive Open System ARchitecture website," 2014. [Online]. Available: http://www.autosar.org/

[16] D. Karlsson, "TIMMO-2-USE Innovation Report," 2012. [Online]. Available: https://itea3.org/project/result/download/6631/09033-TIMMO-2-USE-TIMMO-2-USE%20Innovation%20Report.pdf

[17] University of Cantabria, "Modeling and Analysis Suite for Real-Time Applications," 2015. [Online]. Available: http://mast.unican.es/

[18] R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter, and R. Ernst, "System level performance analysis-the symta/s approach." IEEE Proceedings-Computers and Digital Techniques 152.2, 2005, pp. 148–166.

[19] A. Albinet, S. Begoc, J. Boulanger, O. Casse, I. Dal, H. Dubois, F. Lakhal, D. Louar, M. Peraldi-Frati, Y. Sorel *et al.*, "The memvatex methodology: from requirements to models in automotive application design," in *4th European Congress ERTS (Embedded Real Time Software), Toulouse, France*, 2008.

[20] H. Le Dang, H. Dubois, and S. Gérard, "Towards a traceability model in a marte-based methodology for real-time embedded systems," *Innovations in Systems and Software Engineering*, vol. 4, no. 3, pp. 189–193, 2008.

[21] J. L. Medina and Á. G. Cuesta, "From composable design models to schedulability analysis with uml and the uml profile for marte," *ACM SIGBED Review*, vol. 8, no. 1, pp. 64–68, 2011.

[22] C. Mraidha, S. Tucci-Piergiovanni, and S. Gerard, "Optimum: a marte-based methodology for schedulability analysis at early design stages," *ACM SIGSOFT Software Engineering Notes*, vol. 36, no. 1, pp. 1–8, 2011.

[23] A. Noyer, P. Iyenghar, E. Pulvermueller, F. Pramme, J. Engelhardt, B. Samson, and G. Bikker, "Tool Independent Code Generation for the UML - Closing the Gap between Proprietary Models and the Standardized UML Model," in *Evalutation to Novel Approaches of Software Engineering (ENASE 2014)*, Lisbon, Portugal, 2014.

[24] Willert Software Tools, "ReqXChanger DataSheet," 2014. [Online]. Available: http://www.willert.de/assets/Datenblaetter/DatS-ReqXChanger-V-1.1.4en-2014.pdf

[25] P. Iyenghar, B. Samson, M. Spieker, A. Noyer, J. Wuebbelmann, C. Westerkamp, and E. Pulvermueller, "A Mechanism for Data Interchange Between Embedded Software Sub-Systems Developed Using Heterogenous Modeling Domains," in *International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2015)*, Angers, France, 2015.