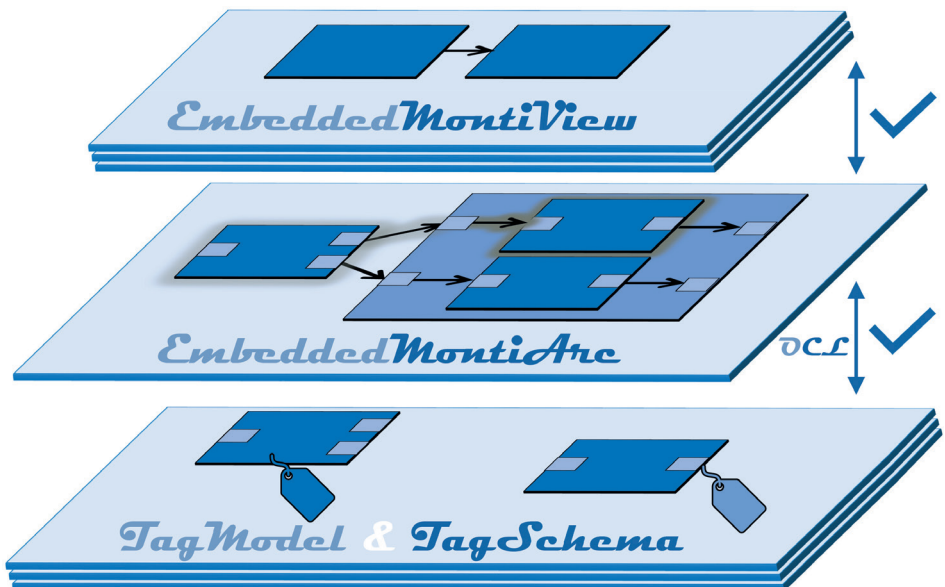


Michael von Wenckstern

Verification of Structural and Extra-Functional Properties in Component and Connector Models for Embedded and Cyber-Physical Systems



Aachener Informatik-Berichte,
Software Engineering

Hrsg: Prof. Dr. rer. nat. Bernhard Rumpe

Band 44

Verification of Structural and Extra-Functional Properties in Component and Connector Models for Embedded and Cyber-Physical Systems

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der
RWTH Aachen University zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Dipl.-Math. Michael von Wenckstern
aus Karl-Marx-Stadt

Berichter: Universitätsprofessor Dr. rer. nat. Bernhard Rumpe
University Professor Shahar Maoz, PhD

Tag der mündlichen Prüfung: 31.10.2019

Aachener Informatik-Berichte, Software Engineering

herausgegeben von
Prof. Dr. rer. nat. Bernhard Rumpe
Software Engineering
RWTH Aachen University

Band 44

Michael von Wenckstern
RWTH Aachen University

**Verification of Structural and Extra-Functional
Properties in Component and Connector Models for
Embedded and Cyber-Physical Systems**

Shaker Verlag
Düren 2020

Bibliographic information published by the Deutsche Nationalbibliothek

The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at <http://dnb.d-nb.de>.

Zugl.: D 82 (Diss. RWTH Aachen University, 2019)

Copyright Shaker Verlag 2020

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior permission of the publishers.

Printed in Germany.

ISBN 978-3-8440-7239-6

ISSN 1869-9170

Shaker Verlag GmbH • Am Langen Graben 15a • 52353 Düren

Phone: 0049/2421/99011-0 • Telefax: 0049/2421/99011-9

Internet: www.shaker.de • e-mail: info@shaker.de

Abstract

The industry area of embedded and cyber-physical systems is one of the largest and it influences our daily life. The global embedded systems market was valued at about 160 billion US dollar in 2015 and it is getting up to 225 billion US dollar by end of 2021 [Zio17]. Example domains of embedded and cyber-physical systems are: automotive [DHJ⁺08], avionics [FLV03], robotics [WICE03], railway [DNCH10], production industry [EWSG94], telecommunication [ZSM11], healthcare [ERA09], defense [BNP⁺04], and consumer electronics [VOVDLKM00].

Model-based engineering, esp. component and connector (C&C) models to describe logical architectures, are one common approach to handle the large complexity of embedded and cyber-physical systems [FR07, MBNJ09, OMG15, EJM⁺03]. Components encapsulate software features; the hierarchical decomposition of components enables formulating logical architectures in a top-down approach. Connectors in C&C models describe the information exchange via typed ports; they model black-box communication between software features.

The current development of complex C&C-based embedded systems in industry mostly involves the following steps [BMR⁺17a, DGH⁺19]: (1) formulating functional and extra-functional requirements as text in *IBM Rational DOORS*; (2) creating a design model of the software architecture including its environment interactions in *SysML*; (3) developing a complete functional/logical model to simulate the embedded system in *Simulink*; and (4) system implementation based on available hardware in C/C++ satisfying all extra-functional properties.

This current development process has the following disadvantages [KBFS12, HKK⁺18, BMR⁺17a]: (a) *SysML* models do not follow a formalized approach; i.e., engineers may interpret these models differently due to missing semantics; (b) the check between the informal *SysML* architecture design against the *Simulink* model is done manually, and thus, error-prone and very time-consuming; (c) refactoring of *Simulink* models (e.g., dividing a subsystem) needs manual effort in updating the design model, and therefore, due to timing constraints this step is often skipped resulting in inconsistencies; and (d) most tools do not support a generic approach for different extra-functional property kinds, and thus, extra-functional properties are mostly modeled as comments or stereotypes and consistencies between these properties are checked manually.

This thesis aims to improve the software development process of large and complex C&C models for embedded and cyber-physical systems by providing model-based methodologies to develop, understand, validate and maintain these C&C models. Concretely, this thesis presents concepts to support the embedded software engineer with: (i) automatic consistency checks of C&C models; (ii) automatic verification of logical C&C models against their design decisions; (iii) automatic addition of traceability links between design and implementation models; (iv) finding structural inconsistencies during model evolution; (v) providing a flexible framework to define different extra-functional property types; (vi) presenting an *OCL* framework to specify (company-specific) constraints about structural or extra-functional properties for C&C models; and (vii) generation of positive or negative witnesses to explain why a C&C model satisfies or violates its extra-functional or structural constraints or its design decisions.

Prototype implementations of above mentioned concepts and an industrial case study in cooperation with Daimler AG show promising results in improving the model-based development process of embedded and cyber-physical systems in industry.

Acknowledgments

Special thanks goes to my doctor father Prof. Dr. Bernhard Rumpe for giving me the opportunity to research in the area of model-driven software engineering and for providing many connections to other research groups and industrial partners; participating in several research and automotive industry projects gave me access to many different views on this research topic. Furthermore, I want to express great laud to Prof. Dr. Bernhard Rumpe for the fruitful discussions, ideas and feedback rounds - especially during the 3-day long research workshops - on this thesis.

I would like to thank Prof. Shahar Maoz for being the second reporter of this thesis. Additionally, I want to express gratitude to Prof. Shahar Maoz, Dr. Jan Oliver Ringert, and Prof. Dr. Bernhard Rumpe for the successful teamwork on the GIF research project I-1235-407.6/2014, for the published papers together, and for the time together in Tel Aviv and in Aachen. I also want to thank the Deutsche Forschungsgemeinschaft (DFG) for funding this research and thus supporting this thesis.

Moreover, I would like to thank Prof. Dr. Stefan Kowalewski for heading my PhD exam committee, and Prof. Dr. Erika Ábrahám for being part of this committee.

In addition, my thank goes to Prof. Dr. Manfred Nagl for many interesting discussions about computer science, GI, and architecture.

“If everyone is moving forward together, then success takes care of itself.” - Henry Ford about teamwork. Therefore, special thanks goes to Evgeny Kusmenko; together we developed the EmbeddedMontiArc modeling family and had good times on several conferences and workshops presenting our papers. Moreover, I want to thank Dr. Pedram Mir Seyed Nazari for the interesting discussions about language engineering and digitalization; for the first level MontiCore support thanks goes to Pedram, Marita Breuer, and Galina Volkova. Additionally, I want to thank Dr. Alexander Roth, Dr. Christoph Schulz, and Vincent Bertram for the many papers we wrote together. I want to thank Deni Raco for organizing all the extra activities such as poker, or soccer events on our chair.

I also would like to thank the students whose bachelor or master theses supported my research significantly; without them the large evaluation part would not be possible. Thanks goes to “Web(Assembly) expert” Jean-Marc Ronck, “OCL specialist” Ferdinand Mehlan, “C++ generator enthusiast” Sascha Schneiders, and “tooling and gaming fan” Malte Heithoff for supporting me more than two years.

I want to extend my thanks to the other students: Luca Tabone, Stefan Brunecker, Alexander Kogaj, Nicolai Strodthoff, Severin Tolksdorf, Vladimir Parashin, David Ernst, Christian Bajana, Igor Shumeiko, Dinh-An Ho, Fabian Kahlert, Manuel Schrick, Ievgen Strepkov, Ahmet Tayfun Özen, and all seminar and lab students.

Special thanks for reviewing this PhD thesis goes to Sebastian von Wenckstern, Pedram Mir Seyed Nazari, Evgeny Kusmenko, Kai Adam, Simon Varga, and Oliver Kautz.

Furthermore, I would like to thank all colleges of the Software Engineering chair for the nice time together: Lennart Bucher, Arvid Butting, Manuela Dalibor, Anabel Derlam, Florian Donath, Imke Drave, Robert Eikermann, Dr. Timo Greifenberg, Sylvia Gunder, Dr. Arne Haber, Dr. Lars Hermerschmidt, Dr. Christoph Herrmann, Gabriele Heuschen, Steffen Hillemacher, Dr. Katrin Hölldobler, Carsten Kolassa, Thomas Kurpick, Achim Lindt, Dr. Markus Look, Matthias

Markthaler, Dr. Judith Michael, Dr. Klaus Müller, Sonja Müßigbrodt, Lukas Netz, Jerome Pfeiffer, Nina Pichler, Dimitri Plotnikov, Manuel Putzer, Dr. Dirk Reiß, David Schmalzing, Stephanie Schrader, Brian Sinkovec, Max Voß, Louis Wachtmeister, and Dr. Andreas Wortmann.

My largest thanks goes to my family - Bettina, Bodo, and Sebastian von Wenckstern - who supported me all these years; and their belief that education is the best investment for my future.

Aschaffenburg, 13.12.2019

Michael von Wenckstern

Contents

1. Introduction	1
1.1. Context and Foundations	2
1.1.1. Component and Connector Models and their Specification Language . .	3
1.1.2. Model Based Systems Engineering	5
1.1.3. <i>MontiCore</i>	9
1.2. Requirements on PhD Thesis	13
1.2.1. Enhancing the C&C Views Language	13
1.2.2. Advancing C&C Views Analyses	13
1.2.3. Integrating C&C Views in the Development Process and Environment .	14
1.2.4. Evaluation	14
1.2.5. Further Remarks	14
1.3. Objective and Main Results	15
1.4. Thesis Organization	17
1.5. Publications	17
2. Underlying Development Methodology	21
2.1. Systems Engineering Process at Daimler AG	21
2.1.1. Current Development Process at Daimler AG	21
2.1.2. Improving the Development Process at Daimler AG	23
2.2. Digitalizing the Systems Engineering Process using SMARDT	27
2.2.1. Current Systems Engineering Process at BMW Group	27
2.2.2. Overview of SMARDT process	28
2.3. Similar Existing Methodologies and Model-based Approaches	33
2.3.1. <i>Simulink</i> Requirements	34
2.3.2. <i>Mentor Capital</i>	35
2.3.3. <i>Polarsys Arcadia</i>	36
2.3.4. Vector <i>PREEvision</i>	38
3. Concrete Syntax of <i>EmbeddedMontiArc</i>	41
3.1. Requirements for a Logical Architecture Modeling Language	42
3.2. Existing C&C Modeling Languages	43
3.3. Comparison to Other <i>MontiArc</i> Derivatives	49
3.4. Overview of <i>EmbeddedMontiArc</i> Modeling Family	54
3.5. Typing in <i>EmbeddedMontiArc</i>	62
3.5.1. Port Type System	62
3.5.2. Matrices as Port Types	64
3.5.3. Configuration Parameters of Port Type System	65

3.5.4.	Type Parameters	66
3.6.	Components and Ports in <i>EmbeddedMontiArc</i>	70
3.6.1.	Component Type Definitions and Component Instantiations	70
3.6.2.	Arrays of Ports and Component Instantiations	72
3.6.3.	Component Interfaces	73
3.6.4.	Reference Architectures with Configuration Parameters	74
3.6.5.	Product-Line Modeling with Configuration Parameters and Default Values	78
3.6.6.	Connections	83
3.6.7.	Main Component Instantiation and Packaging	88
3.6.8.	Arrays of Component Types, Generic and Configuration Parameters	90
3.7.	Concepts of New Language Features	92
3.8.	Example Use Case for <i>EmbeddedMontiArc</i> in Business Domain	95
3.9.	<i>EmbeddedMontiArcStudio</i> : Tooling for Users	96
4.	Internal Representation of <i>EmbeddedMontiArc</i>	103
4.1.	Deriving Class Diagrams from <i>MontiCore</i> Grammars	104
4.2.	Component and Connector Model	107
4.2.1.	Port Type System	108
4.2.2.	Parameter Definitions and Parameter Bindings	111
4.2.3.	Component Instantiation	115
4.2.4.	Ports and Connectors	116
4.2.5.	Effector	119
4.2.6.	Component and Component Interface	120
4.2.7.	Component and Connector Model	121
4.3.	Component and Connector Instance Structure	122
4.4.	Derivation of C&C Instance Structure from C&C Model	124
4.5.	Comparison of <i>EmbeddedMontiArc</i> against <i>MontiArc</i> Derivatives	129
4.6.	Realization of the Abstract Syntax with Symbol Management Infrastructure	137
5.	Enriching <i>EmbeddedMontiArc</i> Models with Extra-Functional Properties	145
5.1.	Overview of Existing Extra-Functional Properties	146
5.2.	Existing Approaches For Annotating Models with Extra-Functional Properties	148
5.3.	Requirement Analysis	153
5.4.	Running Example	154
5.5.	Tagging Mechanism for Component and Connector Models	157
5.5.1.	General Approach	158
5.5.2.	Tag Schema	159
5.5.3.	Tag Model	161
5.5.4.	Derivation of Class Diagrams based on Tag Schemas	165
5.5.5.	Consistency Rules between Tag Model and Tag Schema	167
6.	OCL Framework to Describe Structural and Extra-Functional Properties	169
6.1.	OCL Framework to Define Context Conditions of C&C Models	170
6.1.1.	Workflow to Define and Validate OCL Context Conditions	170

6.1.2.	CO1: Connectors May Not Pierce Through Component Interfaces . . .	171
6.1.3.	R1/R2: Connection Rules for Outgoing and Incoming Ports	174
6.1.4.	R13: No Subcomponent Instantiation Cycles	175
6.1.5.	B1: Names Are Unique Within Component Namespace	176
6.1.6.	CV5/CV6: All Ports Should Be Used Once	177
6.1.7.	R9/R10: All Parameters Must be Bound During Instantiation	179
6.2.	Defining Extra-Functional Properties in <i>OCL</i>	182
6.2.1.	Traceability for Component Instantiation	182
6.2.2.	Maximal Power Consumption for Component Instantiation	183
6.2.3.	Encryption for Port Instantiation	184
6.2.4.	Authentication for Connector Instantiation	185
6.2.5.	Certificates for Component Instances/Port Definitions	186
6.2.6.	Maximal Power Consumption of Subcomponent Instantiations	187
6.2.7.	Encryption for Target Ports	188
6.2.8.	Power Consumption Considering Encryption	189
6.2.9.	Automotive Safety Integrity Level for Component Definitions	192
6.2.10.	Worst Case Execution Time for Single Core Processors	193
6.2.11.	Worst Case Execution Time for Processors with Infinite Cores	193
6.2.12.	Worst Case Execution Time for Multi Thread Processors	195
6.3.	Witnesses of <i>OCL</i> Constraints for Extra-Functional Properties	197
6.3.1.	Positive Consistency Witnesses	197
6.3.2.	Negative Inconsistency Witnesses	199
6.4.	<i>OCL</i> to Specify Transformations between Abstract Syntax of Two Languages .	201
6.5.	Some Remarks about the Implementation	207
6.6.	Related Approaches to Constrain Structure of Architectures	210

7. *EmbeddedMontiView*: A High-Level Design Language **213**

7.1.	Requirements/Features on the C&C View Language	214
7.2.	Related Concepts for Verifying Component and Connector Models	216
7.3.	Concrete and Abstract Syntax of <i>EmbeddedMontiView</i> Language	221
7.3.1.	Abstract Component Type Definition	221
7.3.2.	Abstract Component Instantiations	224
7.3.3.	Array of Abstract Component Instances and Abstract Ports	224
7.3.4.	Completeness of Abstract Component Instances	226
7.3.5.	Abstract Type Parameters	228
7.3.6.	Matrices as Abstract Port Types	229
7.3.7.	Abstract Connections	232
7.3.8.	Abstract Effectors	234
7.3.9.	Imports and Full-Qualified Names	235
7.3.10.	Component and Connector View	236
7.3.11.	Some Remarks	237
7.4.	Satisfaction Relation between <i>EmbeddedMontiView</i> and <i>EmbeddedMontiArc</i> . .	237
7.4.1.	Abstract Ports	237

7.4.2.	Abstract Subcomponent Instantiations	240
7.4.3.	Abstract Type Parameters	240
7.4.4.	Abstract Tensors as Port Types	241
7.4.5.	Abstract Connections	242
7.4.6.	Abstract Effectors	247
7.4.7.	Some Remarks	247
7.5.	Witnesses Based on Satisfaction-Relation	250
7.5.1.	Satisfaction Witnesses	250
7.5.2.	Tracing Witnesses	262
7.5.3.	Non-Satisfaction Witnesses	265
8.	Industrial Case Study on Component and Connector Views	269
8.1.	Overview of Three Stages of Industrial Case Study	269
8.2.	Preliminary Study	271
8.2.1.	Execution of Preliminary Study	271
8.2.2.	Results of Preliminary Study	272
8.3.	Main Study	279
8.3.1.	Addressing Traceability	282
8.3.2.	Example of a Requirement, C&C View, and Graphical Witness	283
8.3.3.	Design Decisions for Creating C&C Views	286
8.3.4.	Addressing Evolution	286
8.3.5.	Translating <i>Simulink</i> Block Diagrams to <i>EmbeddedMontiArc</i>	288
8.4.	Results of Main Study	292
8.4.1.	Feasibility and Effort to Create C&C Views	292
8.4.2.	Technical Applicability	294
8.4.3.	Helpfulness of Witnesses	296
8.4.4.	Results from Addressing the Identified Challenges	298
8.5.	Subsequent Study	299
8.5.1.	Generation of Graphical Representations of C&C Models and Witnesses	300
8.5.2.	Results of Subsequent Study	306
8.6.	Additional Observations and Desired Extensions	308
8.7.	Threats to Validity	310
8.8.	Similar Studies	310
8.9.	Summary of Industrial Case Study	311
9.	Summary and Conclusion	313
9.1.	Main Results	313
9.2.	Conclusion	315
A.	Appendix for Industrial Case Study	317
A.1.	Screenshots of Visualisation	317
A.2.	Screenshots of Graphical Representation of C&C Views and Witnesses	325
A.2.1.	FA-24	325
A.2.2.	FA-4	328

A.2.3. FA-5	330
A.3. Identified Errors during Case Study	332
A.3.1. ADAS	332
A.3.2. ALS	335
A.4. Statistics about Running Time of Verification Tool	335
A.4.1. Verification Time to Generate Textual (Non-)Satisfaction Witnesses . .	336
A.4.2. Verification Time to Generate Graphical (Non-)Satisfaction Witnesses .	344
B. Class Diagram in CD4A Syntax	349
C. Other Material	369
C.1. <i>MontiCore</i> 5 grammar for C&C instance structure	369
C.2. Operator Priority in <i>OCL</i>	370
C.3. Material to Chain Instances	371
Bibliography	373
List of Figures	427
List of Tables	439
Curriculum Vitae	441
Related Interesting Work from the SE Group, RWTH Aachen	443