

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Alfred Kobsa

University of California, Irvine, CA, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz

University of Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

TU Dortmund University, Germany

Madhu Sudan

Microsoft Research, Cambridge, MA, USA

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max Planck Institute for Informatics, Saarbruecken, Germany

Holger Giese Gabor Karsai Edward Lee
Bernhard Rumpe Bernhard Schätz (Eds.)

Model-Based Engineering of Embedded Real-Time Systems

International Dagstuhl Workshop
Dagstuhl Castle, Germany, November 4-9, 2007
Revised Selected Papers

Volume Editors

Holger Giese
Hasso-Plattner-Institute
for Software Systems Engineering
Potsdam, Germany
E-mail: holger.giese@hpi.uni-potsdam.de

Gabor Karsai
Vanderbilt University
Nashville, TN, USA
E-mail: gabor.karsai@vanderbilt.edu

Edward Lee
University of California at Berkeley
Berkeley, USA
E-mail: eal@eecs.berkeley.edu

Bernhard Rumpe
RWTH Aachen University
Aachen, Germany
E-mail: rumpe@se-rwth.de

Bernhard Schätz
fortiss GmbH
Garching, Germany
E-mail: schaeetz@fortiss.org

Library of Congress Control Number: 2010935675

CR Subject Classification (1998): D.2.9, D.2, D.3.3, C.3-4, F.3

LNCS Sublibrary: SL 2 – Programming and Software Engineering

ISSN	0302-9743
ISBN-10	3-642-16276-2 Springer Berlin Heidelberg New York
ISBN-13	978-3-642-16276-3 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

springer.com

© Springer-Verlag Berlin Heidelberg 2010
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper 06/3180

Preface

The topic of “Model-Based Engineering of Real-Time Embedded Systems” brings together a challenging problem domain (real-time embedded systems) and a solution domain (model-based engineering). It is also at the forefront of integrated software and systems engineering, as software in this problem domain is an essential tool for system implementation and integration. Today, real-time embedded software plays a crucial role in most advanced technical systems such as airplanes, mobile phones, and cars, and has become the main driver and facilitator for innovation. Development, evolution, verification, configuration, and maintenance of embedded and distributed software nowadays are often serious challenges as drastic increases in complexity can be observed in practice.

Model-based engineering in general, and model-based software development in particular, advocates the notion of using models throughout the development and life-cycle of an engineered system. Model-based software engineering reinforces this notion by promoting models not only as the tool of abstraction, but also as the tool for verification, implementation, testing, and maintenance. The application of such model-based engineering techniques to embedded real-time systems appears to be a good candidate to tackle some of the problems arising in the problem domain.

Model-based development strategies and model-driven automatic code generation are becoming established technologies on the functional level. However, they are mainly applied within a limited scope only. The use of analogous modeling strategies on the system, technical, and configuration levels remains challenging, especially with the increasing shift to networks of systems, tight coupling between the control-engineering oriented and reactive parts of a system, and the growing number of variants introduced by product lines. Specific domain constraints such as real-time requirements, resource limitations, and hardware-specific dependencies often impede the acceptance of standard high-level modeling techniques and their application. Much effort in industry and academia therefore goes into the adaptation and improvement of object-oriented and component-based methods and model-based engineering that promise to facilitate the development, deployment, and reuse of software components embedded in real-time environments. The model-based development approach for embedded systems and their software proposes application-specific modeling techniques using domain specific concepts (e.g., time-triggered execution or synchronous data flow) to abstract “away” the details of the implementation, such as interrupts or method calls. Furthermore, analytical techniques (e.g., the verification of the completeness of function deployment and consistency of dynamic interface descriptions) and generative techniques (e.g., automatic schedule generation, default behavior generation) can then be applied to the resulting more abstract models to enable the efficient development of high-quality software.

Our Dagstuhl seminar brought together researchers and practitioners from the field of model-based engineering of embedded real-time systems. The topics covered included: frameworks and methods, validation, model-based integration technology, formal modeling of semantics, fault management, concurrency models and models of computation, requirements modeling, formal derivation of designs from requirements, test modeling and model-based test generation, quality assurance, design management, abstractions and extensions, and development techniques and problems of application domains. The broad spectrum of presentations clearly illustrate the prevalence of model-based techniques in the embedded systems area, as well as progress in the field.

This volume is a collection of long and short papers that survey the state of the art in model-based development of real-time embedded systems. It is composed of longer chapters that cover broad areas and short papers that discuss specific tools. The chapters are organized into sections as follows:

- **Foundations:** The chapters in this section survey general models of reactive systems, techniques, and approaches for model-based integration, and modeling and simulation of real-time applications.
- **Language Engineering:** The chapters here review metamodeling as a fundamental tool, the methods for specifying the semantics of models for dynamic behavior, and the requirements for modeling languages for real-time embedded systems.
- **Domain-Specific Issues:** Relevant issues of real-time embedded systems are discussed in this section, including the use of model-based techniques for safety-critical software, and analysis and development approaches to dependable systems.
- **Life-Cycle Issues:** These chapters discuss requirements modeling techniques for embedded systems, and the technology for model evolution and management

The short papers provide a state-of-the-art survey of existing tools that are being used in the model-based engineering of embedded real-time systems.

Finally, we would like to thank all authors and contributors to the project without whom such a large and complex project could not have been completed. It has been made possible by several researchers who supported the organizers and kept things going. In particular, we have to thank Claas Pinkernell, Markus Look, and Sven Bürger for their assistance in compiling this book. Thanks also goes to the Dagstuhl organization staff members who always make our meetings there a unique event.

Holger Giese
Gabor Karsai
Edward Lee
Bernhard Rumpe
Bernhard Schätz

Table of Contents

Part I: Foundation

1	Models of Reactive Systems: Communication, Concurrency, and Causality	3
	<i>Bernhard Schätz, Holger Giese</i>	
1.1	Models and Abstraction	3
1.1.1	Approach	4
1.1.2	Overview	5
1.1.3	Terminology	5
1.2	Communication	6
1.3	Concurrency	8
1.4	Causality	9
1.5	Models and Aspects	11
1.6	Methodical Combination	12
1.7	Conclusion and Summary	17
2	Model-Based Integration	17
	<i>Holger Giese, Stefan Neumann, Oliver Niggemann, Bernhard Schätz</i>	
2.1	Introduction	17
2.2	Integration	19
2.2.1	Terminology	19
2.2.2	Classification of Integration Problems	21
2.2.3	Fundamental Integration Techniques	22
2.3	State-of-the-Art Approach	29
2.3.1	Function Development	31
2.3.2	Function Integration	33
2.3.3	Discussion	35
2.4	Advanced Model-Based Solutions	36
2.4.1	AUTOSAR	36
2.4.2	ECHATRONIC UML	43
2.4.3	Other Approaches	46
2.5	Summary	48

Part II: Language Engineering

3	Metamodelling: State of the Art and Research Challenges	57
	<i>Jonathan Sprinkle, Bernhard Rumpe, Hans Vangheluwe, Gabor Karsai</i>	
3.1	Metamodelling: State of the Art	57

3.1.1	Concepts in Metamodelling	57
3.1.2	Meta Object Facility (MOF)	61
3.1.3	Essential MOF (EMOF)	61
3.1.4	Eclipse Modelling Framework (EMF)	63
3.1.5	Metamodelling of Languages	64
3.1.6	Textual Metamodelling	65
3.1.7	Concrete and Abstract Syntax	66
3.1.8	Type System	67
3.1.9	Merging of Metamodels	70
3.2	Metamodelling: Research Challenges	71
3.2.1	Semantic Attachment	72
3.2.2	Inference between Metamodels	72
3.2.3	Evolution of Models Driven by Metamodel Evolution	73
3.3	Conclusions	73
4	Semantics of UML Models for Dynamic Behavior:	
	A Survey of Different Approaches	77
	<i>Mass Soldal Lund, Atle Refsdal, Ketil Stølen</i>	
4.1	Introduction	77
4.2	Characterization of Scope, Main Notions, and Criteria for Evaluation	79
4.3	Main Categories of Semantics	81
4.4	Sequence Diagrams and Similar Notations	83
4.4.1	Denotational Semantics	85
4.4.2	Denotational Semantics with Time	86
4.4.3	Denotational Semantics with Probabilities	87
4.4.4	Operational Semantics	87
4.4.5	Operational Semantics with Time	91
4.4.6	Operational Semantics with Probabilities	91
4.5	State Machines and Similar Notations	91
4.5.1	Denotational Semantics	92
4.5.2	Denotational Semantics with Time	92
4.5.3	Denotational Semantics with Probabilities	93
4.5.4	Operational Semantics	93
4.5.5	Operational Semantics with Time	94
4.5.6	Operational Semantics with Probabilities	95
4.6	Evaluation and Comparison	95
4.7	Summary and Conclusions	98

Part III: Modeling

5	Modeling and Simulation of TDL Applications	107
	<i>Stefan Resmerita, Patricia Derler, Wolfgang Pree, Andreas Naderlinger</i>	
5.1	Introduction	107
5.2	The Timing Definition Language	109
5.2.1	TDL Description	109
5.2.2	TDL Extensions for Control Applications	114
5.3	Simulation of TDL Models	117
5.3.1	TDL Simulation in Simulink	117
5.3.2	Using Ptolemy II	120
5.4	Related Work	125
5.5	Conclusions	126
6	Modeling Languages for Real-Time and Embedded Systems: Requirements and Standards-Based Solutions	129
	<i>Sébastien Gérard, Huascar Espinoza, François Terrier, Bran Selic</i>	
6.1	Introduction	129
6.2	Two Main Architectural Styles for Dealing with Abstraction	132
6.3	Modeling Needs for Real-Time and Embedded Systems Design	133
6.3.1	Layering and Needs for RTEs	133
6.3.2	Slicing and Needs for RTEs	134
6.4	MARTE, a Standard Real-Time and Embedded Modeling Language	136
6.4.1	UML Profiling Capabilities	137
6.4.2	MARTE Basics	139
6.4.3	Architecture and Some Details of MARTE	140
6.4.4	An Extract of the MARTE Specification	143
6.4.5	Typical MARTE Usage Scenarios	145
6.5	Related Work	149
6.6	Conclusions and Perspectives	151
7	Requirements Modeling for Embedded Realtime Systems	155
	<i>Ingolf Krüger, Claudiu Farcas, Emilia Farcas, Massimiliano Menarini</i>	
7.1	Introduction and Overview	155
7.1.1	What's in a Requirement?	156
7.1.2	Why Requirements Engineering for ERS Is Hard	158
7.1.3	Summary and Outline	166
7.2	Requirements Specifications and Modeling for ERS	167
7.2.1	Requirements Models	167
7.2.2	Programming Models	172

7.3	Requirements Engineering Approaches: Processes and Practices	174
7.3.1	Requirements Development and Management ...	174
7.4	Example: Failure Management in Automotive Software	179
7.4.1	Central Locking System (CLS)	180
7.4.2	Modeling the CLS Requirements	181
7.4.3	Discussion	190
7.5	Summary and Outlook	191
8	UML for Software Safety and Certification: Model-Based Development of Safety-Critical Software-Intensive Systems	201
	<i>Michaela Huhn, Hardi Hungar</i>	
8.1	Introduction	201
8.2	Development of Certifiable Software	203
8.3	Safety-Related Extensions of UML	207
8.3.1	The UML Profile for Developing Airworthiness-Compliant (RTCA DO-178B) Safety-Critical Software	208
8.3.2	<i>rtUML</i> and the OMEGA-RT Profile	210
8.3.3	Restricting UML for Specification and Programming in a Certification Context	211
8.3.4	The UML Profile for Modeling and Analysis of Real-Time Embedded Systems (MARTE) ...	213
8.3.5	The Railway Control System Domain Profile (RCSD)	215
8.4	Using UML in Certification-Oriented Processes	216
8.4.1	Questions to Be Addressed by a Certification-Oriented Process	216
8.4.2	Purpose and Scope of the Proposed Process ...	216
8.4.3	Terms and Definitions	218
8.4.4	Phases and Sub-processes	219
8.4.5	The Use of UML in the Process	220
8.4.6	Realization	221
8.5	Verification and Validation Techniques	222
8.5.1	General Remarks on Verification and Validation Techniques in Model-Based Development of Certifiable Software	222
8.5.2	Testing	225
8.5.3	(Formal) Verification	228
8.5.4	Tool Support	229
8.6	Conclusion	233

Part IV: Model Analysis

9	Model Evolution and Management	241
	<i>Tihamer Levendovszky, Bernhard Rumpe, Bernhard Schätz, Jonathan Sprinkle</i>	
9.1	Why Models Evolve and Need to Be Managed?	241
9.1.1	Introduction	241
9.1.2	Model Management	242
9.1.3	Model Evolution	243
9.1.4	Chapter Outline	243
9.2	Model Management	243
9.2.1	Model Quality and Modeling Standards	244
9.2.2	Model Transformation	249
9.2.3	Model Versioning and Model Merging	252
9.3	Evolution	253
9.3.1	Evolutionary Model Development	253
9.3.2	Automating Evolutionary Transformations	255
9.3.3	Semantics of Evolution	257
9.4	Modelling Language Evolution	259
9.4.1	Syntactic Model Evolution	259
9.4.2	Semantic Model Evolution	260
9.4.3	Techniques for Automated Model Evolution	261
9.4.4	Step-By-Step Model Evolution	262
10	Model-Based Analysis and Development of Dependable Systems	271
	<i>Christian Buckl, Alois Knoll, Ina Schieferdecker, Justyna Zander</i>	
10.1	Introduction	271
10.2	An Overview on Dependability	272
10.3	A Generic Model of Fault-Tolerant Systems	275
10.3.1	System Operation without Faults	275
10.3.2	Faults	277
10.3.3	Fault-Tolerance Mechanism	277
10.3.4	Summary: Modeling of Dependable Systems	279
10.4	Reliability and Safety Analysis	279
10.4.1	The FMECA Method	280
10.4.2	The Fault Tree Analysis Method	281
10.4.3	Markov Analysis	282
10.4.4	Testing and Model-Based Testing	283
10.4.5	Summary: Reliability and Safety Analysis	284
10.5	Languages and Tool Support	284
10.5.1	Models	285

10.5.2	Implementations	288
10.5.3	Summary: Language and Tool Support	289
10.6	Conclusion and Research Challenges	289

Part V: Approaches

11	The EAST-ADL Architecture Description Language for Automotive Embedded Software	297
	<i>Philippe Cuenot, Patrick Frey, Rolf Johansson, Henrik Lönn, Yiannis Papadopoulos, Mark-Oliver Reiser, Anders Sandberg, David Servat, Ramin Tavakoli Kolagari, Martin Törngren, Matthias Weber</i>	
11.1	Introduction	297
11.2	Modeling and Analysis Capabilities of the EAST-ADL2	299
11.3	A Small Case Study	301
11.3.1	Vehicle Features: Vehicle Level	301
11.3.2	Abstract Functional Description: Analysis Level	301
11.3.3	Concrete Functional Description: Design Level	302
11.3.4	Software Architecture: Implementation Level	304
11.4	Related Work, Conclusions and Further Work	304
12	Fujaba4Eclipse Real-Time Tool Suite	309
	<i>Claudia Priesterjahn, Matthias Tichy, Stefan Henkler, Martin Hirsch, Wilhelm Schäfer</i>	
12.1	Introduction	309
12.2	Features	310
12.3	Case Study: RailCab	313
12.4	Conclusions and Future Work	314
13	AutoFocus 3 - A Scientific Tool Prototype for Model-Based Development of Component-Based, Reactive, Distributed Systems	317
	<i>Florian Hölzl, Martin Feilkas</i>	
13.1	Introduction	317
13.2	Capabilities of AUTOFOCUS 3	318
13.2.1	Logical Architecture	318
13.2.2	Technical Architecture	320
13.3	Conclusion	321

14	MATE - A Model Analysis and Transformation Environment for MATLAB Simulink	323
	<i>Elodie Legros, Wilhelm Schäfer, Andy Schürr, Ingo Stürmer</i>	
14.1	Introduction	323
14.2	Approach	324
14.3	Application	326
14.4	Conclusion	328
15	Benefits of System Simulation for Automotive Applications	329
	<i>Oliver Niggemann, Anne Geburzi, Joachim Stroop</i>	
15.1	System Models	329
15.1.1	State of the Art and AUTOSAR	330
15.2	System Simulation	332
15.3	Applications of System Simulation	334
15.3.1	Specification Verification	334
15.3.2	Software Component Tests	334
15.3.3	ECU Tests	335
15.3.4	Virtual Integration	335
15.4	Summary	335
16	Development of Tool Extensions with MOFLON	337
	<i>Ingo Weisemöller, Felix Klar, Andy Schürr</i>	
16.1	Introduction	337
16.2	History and Overview of Features	338
16.2.1	MOF Editor and Code Generation for MOF Models	338
16.2.2	Additional Frontends	339
16.2.3	Model Transformations	339
16.2.4	Triple Graph Grammar Editor	339
16.3	Usage Scenarios	340
16.3.1	Tool Adapters	340
16.3.2	Model Analysis and Repair	341
16.3.3	Integration Framework	341
16.4	Conclusions and Future Work	342
17	Towards Model-Based Engineering of Self-configuring Embedded Systems	345
	<i>DeJiu Chen, Martin Törngren, Magnus Persson, Lei Feng, Tahir Naseer Qureshi</i>	
18.1	Introduction	345
18.2	Capabilities	346
18.3	Case Study	350
18.3.1	Architecture Modelling with UML	350
18.3.2	Verification and Validation through Analysis	351
18.3.3	Run-Time Models	352
18.4	Conclusions and Future Work	352

18 Representation of Automotive Software Description Means in ASCET	355
<i>Ulrich Freund</i>	
18.1 Introduction	355
18.2 Overview of Design Means for Automotive Software Design	356
18.2.1 Description Means for Control Engineering	356
18.2.2 Description Means for Software Engineering	356
18.3 Integration of the Design Approaches in ASCET	357
18.3.1 Classes	358
18.3.2 Modules	358
18.3.3 Model-Types	359
18.3.4 Tasks	359
18.3.5 Implementations: Integer Arithmetic and Memory Section	359
18.3.6 Codegeneration Approach	360
18.4 Conclusion	360
19 Papyrus: A UML2 Tool for Domain-Specific Language Modeling	361
<i>Sébastien Gérard , Cédric Dumoulin, Patrick Tessier, Bran Selic</i>	
19.1 Introduction	361
19.2 Capabilities	362
19.2.1 Overview	363
19.2.2 Global Architecture and Design Tenets	363
19.2.3 UML2 Graphical Modeling Capabilities	364
19.2.4 Building DSL Tools Profiling the UML2	366
19.3 Case Study	366
19.4 Conclusions and Future Work	367
20 The Model-Integrated Computing Tool Suite	369
<i>Janos Sztipanovits, Gabor Karsai, Sandeep Neema, Ted Bapty</i>	
20.1 Introduction	369
20.2 Components of the MIC Tool Suite	370
20.2.1 The Generic Modeling Environment (GME)	370
20.2.2 Transforming the Models: UDM and GReAT ...	371
20.2.3 Integrating Design Tools: The Open Tool Integration Framework	372
20.2.4 Design Space Exploration	372
20.3 Application Example: Vehicle Control Platform	373
20.4 Conclusion	374

**21 Application of Quality Standards to Multiple Artifacts
with a Universal Compliance Solution 377**
Tibor Farkas, Torsten Klein, Harald Röbig

21.1 Introduction 377

21.2 Idea: Meta-modeling for Constraint Definition 378

21.3 Approach: Universal Compliance Achievement 379

21.4 Case Studies: Compliance with Modeling Standards 381

21.5 Conclusion 382

Author Index 385