MODEL BASED DEVELOPMENT OF EMBEDDED SYSTEMS USING LOGICAL CLOCK CONSTRAINTS AND TIMED AUTOMATA

Jagadish Suryadevara

Akademisk avhandling

som för avläggande av teknologie doktorsexamen i vid Akademin för innovation, design och teknik kommer att offentligen försvaras måndagen den 9 december 2013, 13.15 i Kappa, Mälardalen University, Västerås.

Fakultetsopponent: Professor Robert De Simone,
INRIA Sophia Antipolis - Méditerranée research centre
Abstract

In modern times, human life is intrinsically depending on real-time embedded systems (RTES) with increasingly safety-critical and mission-critical features, for instance, in domains such as automotive and avionics. These systems are characterized by stringent functional requirements and require predictable timing behavior. However, the complexity of RTES has been ever increasing requiring systematic development methods. To address these concerns, model-based frameworks and component-based design methodologies have emerged as a feasible solution. Further, system artifacts such as requirements/specifications, architectural designs as well as behavioral models like statemachine views are integrated within the development process. However, several challenges remain to be addressed, out of which two are especially important: expressiveness, to represent the real-time and causality behavior, and analyzability, to support verification of functional and timing behavior.

As the main research contribution, this thesis presents design and verification techniques for model-based development of RTES, addressing expressiveness and analyzability for architectural and behavioral models. To begin with, we have proposed a systematic design process to support component-based development. Next, we have provided a real-time semantic basis, in order to support expressiveness and verification for structural and behavioral models. This is achieved by defining an intuitive formal semantics for real-time component models, using ProCom, a component model developed at our research centre, and also using the CCSL (Clock Constraint Specification Language), an expressive language for specification of timed causality behavior. This paves the way for formal verification of both architectural and behavioral models, using model checking, as we show in this work, by transforming the models into timed automata and performing verification using UPPAAL, a model checking tool based on timed automata. Finally, the research contributions are validated using representative examples of RTES as well as an industrial case-study.
Abstract

In modern times, human life is intrinsically depending on real-time embedded systems (RTES) with increasingly safety-critical and mission-critical features, for instance, in domains such as automotive and avionics. These systems are characterized by stringent functional requirements and require predictable timing behavior. However, the complexity of RTES has been ever increasing requiring systematic development methods. To address these concerns, model-based frameworks and component-based design methodologies have emerged as a feasible solution. Further, system artifacts, such as requirements, architectural designs as well as behavioral models like statemachine views, are integrated within the development process. However, several challenges remain to be addressed, out of which two are especially important: expressiveness, to represent the real-time and causality behavior, and analyzability, to support verification of functional and timing behavior.

As the main research contribution, this thesis presents design and verification techniques for model-based development of RTES, addressing expressiveness and analyzability for architectural and behavioral models. To begin with, we have proposed a systematic design process to support component-based development. Next, we have provided a real-time semantic basis, in order to support expressiveness and verification for structural and behavioral models. This is achieved by defining an intuitive formal semantics for real-time component models, using ProCom, a component model developed at our research centre, and also using CCSL (Clock Constraint Specification Language), an expressive language for specification of timed causality behavior. This paves the way for formal verification of both architectural and behavioral models, using model checking, as we show in this work, by transforming the models into timed automata and performing verification using UPPAAL, a model checking tool for timed automata. Finally, the research contributions are validated using representative examples of RTES as well as an industrial case-study.
Människans vardagliga liv är allt mer beroende av inbyggda realtidssystem. Det vill säga tidskritiska datorsystem som till exempel finns i bilar, flygplan och andra elektroniska apparater. Till skillnad från traditionella stationära datorer måste realtidssystemens funktionalitet vara förutsägbart med avseende på förväntat tidsbeteende. Till exempel måste säkerhetskritiska funktioner såsom krockkuddarna i en bil aktiveras inom angivna tidsgränser i händelse av en olycka. Detta innebär att för tidig eller för sen aktivering av krockkuddarna inte ger önskat skydd och till och med kan orsaka ytterligare skador.

Realtidssystemen blir ständig mer komplexa vilket resulterar i ett ökande behov av ingenjörsmässiga metoder för att garantera förutsägbarheten hos dessa system. Därutöver måste affärsässiga krav tillgodoses såsom kort utvecklingstid och låga utvecklingskostnader. Modellbaserade metoder har visat sig mycket tillämpliga för att tillgodose både krav på förutsägbar funktionalitet samt låga ekonomiska och tidsmässiga kostnader vid utveckling av realtidssystem. I modellbaserad utveckling används systemmodeller såsom struktur- och designbeskrivningar som utvecklas och kan analyseras med avseende på specifikerade funktionalitets- och tidskrav.

I denna avhandling presenteras modellbaserade formella tekniker för att stödja konstruktion och analys av realtidssystem under de tidiga faserna av utvecklingsprocessen. Vi har tillämpat tekniker såsom modellbaserad verifiering (eng. model-checking) för att stödja design och verifiering av inbyggda realtidssystem. Dessa tekniker appliceras på struktur- och beteendemodeller av realtidssystem för att garantera systemets funktionalitet och tidsbeteende. Vi har validerat de föreslagna teknikerna på relevanta exempel av inbyggda realtidssystem samt i industriella fallstudier.
To my family
Acknowledgements

In the beginning, it was not an easy decision to make, to move all the way from India to Sweden, with my family in tow, to pursue a PhD study!! It has not been a smooth sail too having to adjust work, study and family in a new country. However, it has been a great journey in terms of learning in broadest possible sense! Having reached the final step of making this thesis, I look back with a sense of achievement and fulfillment in many ways. Thank you, Sweden!

There are many people I would like to thank who directly or indirectly helped me achieve the milestone of completing this PhD thesis. Foremost, I thank my PhD advisors Paul Pettersson and Cristina Seceleanu, for giving all possible support so that I make a successful research work. Thanks Paul, for all useful discussions and wonderful words of wisdom. My best pick has always been “.. think simple, think stupid...”, and I love to remember this! Thanks Cristina, for always giving an extra effort in making sure I continue my work without losing focus and passion during critical moments. It has been a wonderful time working with both of you and I couldn’t have asked for more from you as my PhD advisors, thanks Paul and Cristina!

I would like to thank all former and present members of my research group Aida Causevic, Leo Hatvani, Aneta Vulgarakis, Stefan Björnander, Raluca Marinescu, Eduard Paul Enoiu, and Eun-Young Kang for inspiring research discussions and presentations. I also thank our industrial collaborators, at ABB Corporate Research, Tiberiu Seceleanu, Gaetana Sapienza, and Stein-Erik Elleveseth for all the discussions and fruitful collaboration. During my PhD study, I have got opportunity to visit INRIA, Sophia Antipolis, France. I would like to thank Frédéric Mallet, Julien DeAntoni, Arda Goknil, Ling Yin, Jean-Vivien Millo, and Marie-Agnès Peraldi-Frati for the wonderful time and successful research collaboration during my visit to INRIA. Special thanks to Aida and Leo, my officemates, for being very helpful colleagues, in all research and practical matters.
I would like to thank all teachers, researchers, professors at the department: Hans A. Hansson, Ivica Crnkovic, Björn Lisper, Kristina Lundqvist, Sasikumar Punnekkat, Mikael Sjödin, Damir Isovic, Thomas Nolte, Mats Björkman, Lars Asplund, Dag Nystöm, Radu Dobrin, Daniel Sundmark, Jan Carlson, Frank Lüders, Jukka Mäki-Turja, Antonio Cicchetti, and Moris Behnam for valuable discussions during workshops and also many other occasions. Interacting with you all has always been a great learning experience. Special thanks to Kristina Lundqvist, and Thomas Nolte for reviewing my research proposals and valuable suggestions.

I thank all members of administrative staff and research coordinators for helping in many practical things. Special thanks to Carola Ryttersson, Malin Swanström, Gunnar Widforss, Malin Rosqvist, and Susanne Fronnå.

It was not all work but lot of fun too especially during lunches, coffee breaks, conference trips and special occasions. I would like to thank Abhilash, Adnan, Andreas G., Andreas J., Barbara, Batu, Bob, Federico, Fredrik, Giacomo, Guillermo, Hamid, Hang, Hüseyin, Irfan, Josp, Jonsson, Juraj, Luka, Mehrdad, Meng, Mikael, Mohammad, Nikola, Nima, Omar, Rafia, Raluca, Saad, Sara Abbaspour, Sara Afshar, Svetlana, Saad, Séverine, Yue, Zhou and all for great fun and wonderful time at the department. Special thanks to ‘lunch-time-running’ team members: Malin, Ingrid, Andreas G., Antonio, Nikola, and Moris, for initiating me to the fun of running! It was an achievement of different kind when I once ran a stretch of 20km!

My deepest gratitude to Swedish Foundation for Strategic Research, Swedish Research Council, Ericsson Foundation, and Mälardalen University for funding my research work.

Finally, I would like to thank my family for always being there as an anchor of support and strength through the trials and tribulations: my wife Anuradha, and my daughters Nandana and Mahima. I also feel thankful to Tobi, our little cute rabbit, for keeping my little daughters engaged, while i was busy with the thesis! My deepest love to you all!

Jagdish Suryadevara
Västerås, November, 2013.
Publications

Included in the thesis


Other Publications

Not included in the thesis

A) Journal


B) Conference/ Workshop


C) Technical Reports


## Contents

I  Thesis 7

1  Introduction 9
   1.1  Research Motivation  ......................... 10
       1.1.1  Model-based Development of RTES  .......... 10
       1.1.2  Heterogeneous models  .................... 11
   1.2  Thesis Outline  ............................... 13

2  Background 15
   2.1  UML and MARTE  ............................... 15
   2.2  EAST-ADL  .................................... 16
   2.3  ProCom  ........................................ 17
   2.4  CCSL  ........................................... 18
   2.5  Timed Automata & Model Checking  .............. 19

3  Problem Statement 23
   3.1  Research Goals  ............................... 23
   3.2  Research Method  ............................... 27
   3.3  Included Papers - An Overview  ................. 28

4  Research Contributions 33
   4.1  Pattern-based Design Support for RTES  ........ 33
       4.1.1  Modeling real-time components  ............. 34
       4.1.2  Pattern-based component designs  .......... 35
   4.2  CCSL and TA Semantics for Verification and Validation of RTES Models  ............... 37
       4.2.1  Formalizing real-time component models  .... 38
       4.2.2  Architectural Descriptions using CCSL and TA  .... 40
## Contents

4.2.3 Semantic Basis for MARTE Mode Behaviors using CCSL and Timed Automata ................................................. 42  
4.3 Industrial Case-Study : A Wind Turbine System - Formal Modeling and Verification .......................... 44  
4.3.1 Wind Turbine System (WTS) - An Overview ........ 44  
4.3.2 Formal Modeling using CCSL and Timed automata .... 45  
4.3.3 Formal Verification of WTS and Lessons Learned ... 47  
4.4 Discussion ................................................................. 49  

5 Related Work 51  
5.1 Design Support for Embedded Systems ......................... 51  
5.2 Formal Verification Support for Component-based RTES ... 53  

6 Conclusions and Future Work 55  
6.1 Summary and Conclusions ........................................... 55  
6.2 Future Work .............................................................. 56  

Bibliography 59  

II Included Papers 65  

7 Paper A: Analyzing a Pattern-Based Model of a Real-Time Turntable System 67  
7.1 Introduction ............................................................. 69  
7.2 SaveCCM ............................................................... 70  
7.3 Component Modeling Patterns ................................... 72  
7.3.1 Run-to-Completion Pattern ................................... 72  
7.3.2 History Pattern ................................................... 74  
7.3.3 Execution-Time Pattern ....................................... 75  
7.4 Turntable Production Cell ......................................... 76  
7.4.1 System Design ..................................................... 77  
7.4.2 Modeling a Closed System .................................... 82  
7.4.3 Requirements and Verification ............................... 85  
7.5 Related Work .......................................................... 87  
7.6 Conclusion ............................................................... 88  
Bibliography ................................................................. 91
8 Paper B:
Formal Semantics of the ProCom Real-Time Component Model 95
8.1 Introduction ............................................. 97
8.2 The Component Model ................................. 98
  8.2.1 ProCom ............................................. 98
  8.2.2 Particularities of ProCom ......................... 100
8.3 Formal Semantics of Selected ProCom Architectural Elements 102
  8.3.1 Formalism and Graphical Notation ............... 103
  8.3.2 Formal Semantics of the FSM Language .......... 104
  8.3.3 Overview of ProCom Formalization ............... 105
  8.3.4 Services ........................................... 106
  8.3.5 Data and Trigger Connections .................... 107
  8.3.6 Component Hierarchy ............................. 108
  8.3.7 Linking Passive and Active Components .......... 108
8.4 Discussion and Related Work ............................. 110
8.5 Conclusions ............................................. 112
Bibliography .............................................. 115

9 Paper C:
Pattern-driven Support for Designing Component-based Architectural Models 119
9.1 Introduction ............................................. 121
9.2 ProCom Component Model: An overview ................ 122
9.3 Example: Temperature Control System (TCS) ........... 124
9.4 Our Specification Language: Modemachine + Marte CCSL 125
  9.4.1 Modemachine Definition and Graphical Notation ... 125
  9.4.2 Modes, and Behaviors .............................. 125
  9.4.3 Events, Triggers, and Timeouts .................... 126
  9.4.4 Mode constraints using UML/Marte CCSL .......... 126
  9.4.5 Example Specification: TCS Modemachine .......... 128
9.5 Component Patterns ..................................... 129
  9.5.1 Timer Pattern ...................................... 129
  9.5.2 Discrete Clock Pattern ............................. 131
  9.5.3 Periodic Behavior Pattern ......................... 132
  9.5.4 Controller Pattern .................................. 134
9.6 Pattern Verification ................................. 135
  9.6.1 Verification of periodic behavior pattern ........ 135
  9.6.2 Verification of other Patterns .................... 137
9.7 Temperature Control System: A Complete ProCom Design . 137
9.8 Related work ............................................. 139
9.9 Conclusions ............................................. 140
Bibliography ............................................. 141

10 Paper D:
Analysis Support for TADL2 Timing Constraints on EAST-ADL Models 145

10.1 Introduction ............................................ 147
10.2 Running Example: Brake-By-Wire Application .................. 148
10.3 TADL2: Timing Augmented Description Language ............. 150
  10.3.1 TADL2 Timing Constraints ............................ 150
  10.3.2 TimeBase, Dimension and Unit in TADL2 .................. 151
  10.3.3 BBW Example in TADL2 ............................... 151
10.4 TADL2 to CCSL: Simulation Approach ....................... 153
  10.4.1 The Clock Constraint Specification Language (CCSL) .... 153
  10.4.2 Modelling TADL2 Constraints in CCSL ................. 155
  10.4.3 Executing TADL2 specification with TimeSquare .......... 157
10.5 TADL2 to Timed Automata/UPPAAL: Verification Approach ... 158
  10.5.1 UPPAAL model-checker: An overview .................... 159
  10.5.2 Modeling TADL2 in UPPAAL ......................... 159
  10.5.3 Verification Results .................................... 162
10.6 Discussion of the Approach ................................ 163
10.7 Related Work .......................................... 164
10.8 Conclusions .......................................... 165
Bibliography ............................................. 167

11 Paper E:
Validating EAST-ADL Timing Constraints using UPPAAL 169

11.1 Introduction ............................................ 171
11.2 Running Example: Anti-lock Braking System .................. 173
11.3 Timing Constraints ...................................... 173
  11.3.1 The Clock Constraint Specification Language (CCSL): 
  An Overview .............................................. 174
  11.3.2 EAST-ADL Timing Constraints ......................... 175
  11.3.3 Timing Constraints for the ABS ....................... 175
11.4 Modeling EastADL/CCSL Timing Constraints in UPPAAL ..... 177
  11.4.1 Timed Automata (TA) / UPPAAL: An overview .......... 177
  11.4.2 TA Modeling ......................................... 178
11.5 Verification ............................................. 184
11.6 Related Work and Discussion ........................................ 186
11.7 Conclusions ......................................................... 187
Bibliography .......................................................... 189

12 Paper F:
Verifying MARTE/CCSL Mode Behaviors using UPPAAL ............ 193
12.1 Introduction ......................................................... 195
12.2 Example Systems and Mode-behavior Specifications .............. 196
  12.2.1 Example1: A Temperature Control System (TCS) ............ 196
  12.2.2 Example2: An Anti-lock Braking System (ABS) ............. 198
12.3 CCSL .............................................................. 198
  12.3.1 CCSL Constraints ............................................. 199
  12.3.2 CCSL Constraints for TCS and ABS ....................... 200
  12.3.3 Synchronized Product of CCSL Constraints: An example201
12.4 MARTE/CCSL Mode Behaviors to Timed Automata ............... 202
  12.4.1 Timed automata and UPPAAL: An overview .................. 202
  12.4.2 Transforming Mode Behaviors into Timed automata ........ 203
  12.4.3 The transformed automaton for the TCS .................... 205
  12.4.4 The transformed automaton for the ABS .................... 207
12.5 Verification ....................................................... 209
12.6 Related Work ..................................................... 211
12.7 Conclusion and Future work ...................................... 212
Bibliography .......................................................... 213

13 Paper G:
A Windturbine System: An Industrial Case-Study in Formal Mod-  
eling and Verification .................................................. 215
13.1 Introduction ......................................................... 217
13.2 Windturbine System (WTS): An overview ........................ 218
  13.2.1 Development Process and Environment ..................... 218
  13.2.2 The Wind Turbine System Model ........................... 219
13.3 Preliminaries ....................................................... 220
  13.3.1 EAST-ADL .................................................. 220
  13.3.2 CCSL ...................................................... 221
  13.3.3 Timed Automata ............................................. 221
13.4 WTS: Formal Specification and Modeling ........................ 224
  13.4.1 Data and Events ........................................... 224
  13.4.2 Specification of timed causality behavior .................. 225
  13.4.3 Modeling functional behavior of real-time components .... 226
Contents

13.4.4 Formal modeling of Plant components . . . . . . . 226
13.4.5 Formal modeling of Controller components . . . . . 228
13.4.6 Modeling the WTS system . . . . . . . . . . . . . . 229
13.5 WTS Analysis . . . . . . . . . . . . . . . . . . . . . . . 230
  13.5.1 Simulation . . . . . . . . . . . . . . . . . . . . . 230
  13.5.2 Verification . . . . . . . . . . . . . . . . . . . . . 231
13.6 Discussion and Lessons-learned . . . . . . . . . . . . . . 233
13.7 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . 234
Bibliography . . . . . . . . . . . . . . . . . . . . . . . . . . . . 235
# List of Figures

1.1 a) Model-based development b) V-model of system development ........................................ 11
1.2 Heterogeneous modeling for RTES. ........................................................... 12

2.1 A timed interpretation of UML model using CCSL. ............... 16
2.2 An EAST-ADL model of Brake-By-Wire example. .............. 17
2.3 Data and trigger transfer in a ProCom-based component-design. 17
2.4 Simulation of CCSL specification using TimeSquire tool. .... 19
2.5 Timed automata modeling: lamp and user example. .......... 20
2.6 A schematic view of model checking based verification ....... 21

4.1 Modeling a real-time component behavior using the run-to-completion pattern. .............. 34
4.2 An equivalent timed automata model for the component behavior in Fig. 4.1 with run-to-completion pattern. .............. 35
4.3 MARTE mode behavior model: Temperature Control System. 36
4.4 The discrete clock pattern in ProSave. ................................. 37
4.5 ProCom semantics notation in FSM and timed automata. .... 39
4.6 (a) A ProSave clock with period $P$ and (b) its formal semantics. 40
4.7 EAST-ADL and TADL2: Schematic view of Brake-By-Wire architectural design and timing constraints. ................ 41
4.8 TADL timing constraints as TA: (a) An Event Chain (b) DelayConstraint ($TC’1$) (c) RepeatConstraint ($TC’3$) (d) SynchronizationConstraint ($TC’10$). ................................. 42
4.9 Temperature Control System: MARTE/CCSL Mode behavior specification to timed automaton. ....................... 43
4.10 Wind Turbine System Model ...............................
4.11 Structural modeling for WTS: The plant model. ......... 46
4.12 Timed automata modeling for ROTOR: (a) $\text{RT}_\omega$ (b) $\text{RT}_\text{in} \xrightarrow{\text{RT}_\text{out}}$ (c) Modeling the partial function - An abstraction of ROTOR computation. ............. 47
4.13 Functional behavior of the MainControl component. .......... 47
4.14 An observer automaton to verify the safety-property: $A \land \text{obs}.B \implies x \leq 30$ ........................................... 48

7.1 An example of (a) a composition where components A, B and C are composed by connecting port $p_1$ to $p_3$, and $p_2$ to $p_4$, and timed behaviors: (b) a clock with period $T$ and jitter $J$, (c) a computation updating data variable $a$ after between $\text{Min}$ and $\text{Max}$ time units. ........................................ 71
7.2 PFSM specification of a component behavior ................... 73
7.3 An equivalent timed automata model with run-to-completion pattern ............................................................. 73
7.4 PFSM specification of a component behavior with history .... 74
7.5 A timed automata behavior with history pattern ............... 75
7.6 Annotation of time attributes on PFSM models for execution-time pattern ..................................................... 75
7.7 A timed automata behavior with execution-time pattern ...... 76
7.8 Schematic diagram of a Turntable system ..................... 77
7.9 Software architecture design layout of Turntable system .... 77
7.10 Behavioural model Turntable component. ..................... 80
7.11 Functions and predicates used by Turntable. ................. 80
7.12 Behavioral model of Loader component. ....................... 81
7.13 State machine model of the Driller component. ............. 81
7.14 State machine model of the Tester component. ............ 81
7.15 State machine model of the Unloader component. .......... 82
7.16 Control structure and system architecture of the turntable system as modeled in Save-IDE. ................................. 83
7.17 Behavior of the Clamp environment model. .................. 84
7.18 Behavior of Drill of the environment model. .................. 85

8.1 A ProSys subsystem and a simple ProSave component. ....... 100
8.2 Example of a critical modeling of data and trigger transfer in ProCom. ......................................................... 101
8.3 The graphical notation of the FSM elements and their translation into TA. ..................................................... 104
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.4</td>
<td>The automaton used for synchronization.</td>
<td>105</td>
</tr>
<tr>
<td>8.5</td>
<td>(a) A ProSave service $S_1$ and (b) its formal semantics.</td>
<td>106</td>
</tr>
<tr>
<td>8.6</td>
<td>(a) A ProSave data connection and (b) its formal semantics.</td>
<td>107</td>
</tr>
<tr>
<td>8.7</td>
<td>(a) A ProSave trigger connection and (b) its formal semantics.</td>
<td>107</td>
</tr>
<tr>
<td>8.8</td>
<td>A ProSys subsystem internally modelled by ProSave.</td>
<td>109</td>
</tr>
<tr>
<td>8.9</td>
<td>(a) A ProSave clock with period $P$ and (b) its formal semantics.</td>
<td>109</td>
</tr>
<tr>
<td>8.10</td>
<td>(a) A ProSave input message port and (b) its formal semantics.</td>
<td>110</td>
</tr>
<tr>
<td>8.11</td>
<td>(a) A ProSave output message port and (b) its formal semantics.</td>
<td>110</td>
</tr>
<tr>
<td>9.1</td>
<td>a) A ProSys subsystem and b) A ProSave component.</td>
<td>123</td>
</tr>
<tr>
<td>9.2</td>
<td>Modemachine specification of a temperature control system.</td>
<td>126</td>
</tr>
<tr>
<td>9.3</td>
<td>The timer pattern in ProSave.</td>
<td>130</td>
</tr>
<tr>
<td>9.4</td>
<td>Transformation of the composite mode Cooling of TCS into a ProSave Design, by applying the timer pattern.</td>
<td>131</td>
</tr>
<tr>
<td>9.5</td>
<td>The discrete clock pattern in ProSave.</td>
<td>132</td>
</tr>
<tr>
<td>9.6</td>
<td>The periodic behavior pattern in ProSave.</td>
<td>133</td>
</tr>
<tr>
<td>9.7</td>
<td>Transformation of composite mode TempControl of TCS into a ProSave design, by applying the periodic behavior pattern.</td>
<td>133</td>
</tr>
<tr>
<td>9.8</td>
<td>The controller pattern in ProSave.</td>
<td>134</td>
</tr>
<tr>
<td>9.9</td>
<td>Transformation of the top level mode transitions of TCS into a ProSave design, by applying the controller pattern.</td>
<td>135</td>
</tr>
<tr>
<td>9.10</td>
<td>Translation of the periodic behavior pattern in ProSave into the corresponding network of timed automata.</td>
<td>136</td>
</tr>
<tr>
<td>9.11</td>
<td>The Temperature Control System in ProCom: a ProSys component made of ProSave components.</td>
<td>138</td>
</tr>
<tr>
<td>10.1</td>
<td>Brake-By-Wire functional view augmented with TADL2 timing constraints.</td>
<td>149</td>
</tr>
<tr>
<td>10.2</td>
<td>CCSL Simulation focusing on the constraint TC10 of the BBW Example.</td>
<td>158</td>
</tr>
<tr>
<td>10.3</td>
<td>universal_time: a timebase automaton.</td>
<td>160</td>
</tr>
<tr>
<td>10.4</td>
<td>TADL timing constraints as TA: (a) EventChain ec1 (b) DelayConstraint tc1 (c) RepeatConstraint tc3a (d) SynchronizationConstraint tc10.</td>
<td>161</td>
</tr>
<tr>
<td>10.5</td>
<td>(a) join_stimulus (b) join_response (c) Observer TA to verify ‘tc1a’</td>
<td>162</td>
</tr>
<tr>
<td>11.1</td>
<td>A timing model corresponding to ABS system.</td>
<td>173</td>
</tr>
<tr>
<td>11.2</td>
<td>$TA_{is}PeriodicOn(x, P)$: a perfect periodic clock template.</td>
<td>180</td>
</tr>
</tbody>
</table>
List of Figures

11.3 $TA_{isPeriodicOn.jitter}(x, P, J)$: a clock with period $P$ and jitter $J$. ................................................. 180
11.4 $TA_{delayedFor}(x, y, u, l)$: modeling delayedFor. .... 181
11.5 $TA_{inf}(i1, i2, i3, i4, z)$: modeling inf. ................. 182
11.6 $TA_{sup}(i1, i2, i3, i4, z)$: modeling sup. ............... 183
11.7 $TA_{precedes}(x, y)$: modeling precedes. ............... 184
11.8 Simulation of timing constraint TC3 ...................... 187

12.1 UML/MARTE mode behavior specifications. ............ 197
12.2 MARTE/CCSL mode behavior specifications. ............ 201
12.3 Example LTS: a) $d \sim c$ b) $s \subseteq c$ c) Synchronized product of $d \sim c, s \subseteq c$ .......................... 202
12.4 MARTE/CCSL mode behaviors to timed automata: A mapping strategy. ........................................... 204
12.5 TCS mode behavior to timed automaton. .................. 206
12.6 Timed automaton for ABS Calibrate mode. ............... 207
12.7 LTS of CCSL constraints: a) $a \lesssim x$ b) $x = b$ delayedFor 1 on $c$ ............................................. 208
12.8 a) Extending mode TA transitions b) Observer TAn for chronometric durations. ........................................ 210

13.1 Wind Turbine System Model ................................ 220
13.2 Structural modeling: a plant model for the WTS. ........ 223
13.3 Structural modeling: a controller model of the WTS. ... 224
13.4 Functional behavior of the MainControl component. ..... 226
13.5 Timed automata modeling: (a) $RT_{omega} = RT_{in}$ (b) $RT_{in} = RT_{out}$ (c) Computation RT. ................. 227
13.6 Timed automata model for (event-triggered) ROTOR. ... 227
13.7 Semantic modeling: (a) Periodic triggering (b) Min. exec. time (c) Max. exec. time .................. 228
13.8 Timed automata modeling of MainControl: (a) time-triggering (b) functional behavior. ....................... 229
13.9 Simulation Results ........................................... 231
13.10 An observer automata to verify the safety-property: $A() \text{obs.B implies } x \leqslant 30$ ................................. 232
# List of Tables

3.1 Illustration of how the included papers A-G contribute to the research goals ........................................ 28

7.1 Common interface for components Loader, Driller, Tester, and Unloader .................................................. 78

7.2 Interface of the environment components .......................................................... 84

11.1 EAST-ADL timing constraints for ABS control unit. ........................................ 176

11.2 TA templates for EAST-ADL timing constraints. ........................................ 184

11.3 Performance statistics for verification of EAST-ADL timing constraints. ................. 186

12.1 CCSL constraints for logical and chronometric properties of TCS and ABS. ............... 200

13.1 Timing attributes of Controller components. .................................................. 225
I
Thesis
Chapter 1

Introduction

An embedded system is a computer system with dedicated functionality within a larger electrical or mechanical system. An embedded system is in constant interaction with its physical world (consisting of electrical, mechanical parts) via sensors (devices that measure environment aspects) and actuators (devices that cause necessary changes to environment dynamics). These systems range from simple devices such as home appliances like microwaves, dish-washers, etc., to complex safety-critical and/or mission-critical systems as, for instance, automotive and avionics systems.

Compared to general purpose computers, embedded systems are often associated with real-time constraints and operate in unpredictable physical environments. For instance, in case of an automobile, the air-bag safety feature (meant to protect the driver and other occupants from severe injuries caused by possible violent collisions in case of accidents) needs to be operational within specific time limits, neither earlier nor later, in case an accident occurs. Such time-constrained behavior of an embedded system leads to the need of addressing system predictability and dependability issues in addition to functional correctness. Thus, unlike general purpose computer systems, functional correctness alone is not sufficient for an embedded system. Ensuring predictable behavior of system functionality with respect to associated timing constraints, during system development, is highly desirable [1].

To support the development of real-time embedded systems (RTES), disciplines such as real-time computing [2], real-time design [3] etc., have evolved. For instance, timing issues are addressed by partitioning the system functionality into executable tasks and predictable execution behavior is defined through
task scheduling techniques. Predictability of the system is analyzed through analytical methods such as schedulability analysis, Worst-Case-Execution-Time (WCET) analysis [4], etc. Also, system verification and validation (V&V) is performed through rigorous testing. However, the above described techniques rarely cover all possible system behaviors to be able to establish the required system properties. This leads to the need for applying holistic approaches in system development, for instance through abstraction and modularity, taming complexity and setting the grounds for ensuring predictability by exhaustive analysis.

The need for abstraction and modularity has long been recognized in the context of software engineering [5, 6]. Architecture description languages (ADL) and paradigms such as programming-in-the-Large [7] are deemed effective techniques to address system complexity. The need for structured development methodologies was also identified, leading to methods such as the water-fall model [6], which divides the overall development activity into well-defined phases corresponding to system abstraction levels. Each of these phases such as requirements, design and implementation has been associated with corresponding modeling languages, design techniques and tools. This leads to correctness (meeting requirements) issues among various system artifacts of different development phases with respect to both functional as well as timing behaviors. To address the issues, in addition to traditional testing techniques, advanced analysis techniques such as simulation and exhaustive verification are envisaged.

As the main scope of this thesis, we consider model-based development of embedded systems and address some of the challenges pertaining to complexity and analyzability with respect to overall timing behavior. The solutions we propose fall within the area of real-time embedded systems (RTES).

1.1 Research Motivation

In this section, we present an overview of model-based development for RTES. We will also discuss challenges in addressing complexity, analyzability and reusability, as well as predictability with respect to both functionality and timing, which motivate our research.

1.1.1 Model-based Development of RTES

The model-based development paradigm, as shown schematically in Fig. 1.1(a), is a holistic top-down approach, correlating specification and design models
for developing systems. To address analyzability and reusability, in developing complex RTES, model-based frameworks divide the overall system development into layers of system abstractions. The abstractions not only support the design process but facilitate continuous verification and validation (V&V) of the system, as represented by applying the V-model [6] of development (Fig. 1.1(b)). These abstractions also facilitate advanced analysis techniques, such as simulation and exhaustive verification, to ensure predictability and dependability of RTES before implementation. Further, model-based approaches enhance reusability of system artifacts, which leads to cost-effective development.

![Deployment Model](image1)  
![V-model](image2)

**Figure 1.1**: a) Model-based development b) V-model of system development

### 1.1.2 Heterogeneous models

Various modeling notations such as statemachines and data-flow (DF) models can be combined to achieve both modeling flexibility as well as expressiveness of the RTES models. For instance, as shown in Fig. 1.2, statemachine-based behavior models can be used at two levels of the modeling hierarchy. The highest level model, a statemachine view, can capture the overall reactive behavior (for instance, event driven) of the system. A ‘state’ at this modeling level may represent high-level operational modes (e.g. M1 in Fig. 1.2) of the system, for instance, ‘TakeOff’, ‘Flying’, and ‘Landing’ modes of an aircraft. Within
a mode, a DF model can represent the structural entities (e.g. components C1, C2 and C3 within mode M1 in Fig. 1.2) that constitute the corresponding configuration of the mode. The DF models across different modes are usually not disjoint but may have common elements (e.g. C2 in Fig. 1.2), for instance representing system entities that are ‘active’ during certain operational modes and not active in other modes.

In the context of model-based development for RTES, as described above, we present the following challenges as addressed by this thesis.

- The design methodologies need to address the correlation of structural and behavioral models with respect to overall real-time execution and timing behavior of the system.

- The modeling artifacts, such as architectural and abstract behavioral models of a system need to be intuitive and expressive as well as include a semantic basis to support formal verification of system properties for systems described by heterogeneous models that use different triggering and communication mechanisms.

In this thesis, we have presented a pattern-based design support to develop architectural designs (component-based) for RTES. Next, using real-time formalisms such as CCSL (Clock Constraint Specification Language) and timed automata, we have proposed a semantic basis for system models to support formal verification. We have also proposed verification techniques for model checking structural and behavioral models of RTES. Finally, we have validated the research contributions using relevant examples of RTES as well as an industrial case-study, a simplified wind turbine system from ABB Corporate Research.
1.2 Thesis Outline

This thesis is presented as a collection of research papers and is organized into two parts. The remainder of Part 1 is divided as follows. In Chapter 2, we present the preliminaries of the research work. In Chapter 3, we describe the research problem and present the related research goals. The research contributions of the thesis work are presented in Chapter 3. In Chapter 4, we discuss related work, as well as some limitations of the contributions presented in this thesis. In Chapter 5 we conclude the thesis and present a discussion of future work.

Part 2 of the thesis contains the included research papers, as listed in the beginning of the thesis.
Chapter 2

Background

In this chapter, we present preliminaries needed to comprehend this thesis work.

2.1 UML and MARTE

The Unified Modeling Language (UML) [8] consists of standard notations for multi-view modeling of computer-based systems. It provides a set of graphical notations to model both structure and dynamics of a system. For instance, functional structure of a system can be modeled hierarchically using a class-diagram or a component-diagram. The system dynamics such as interaction between functional elements and reactive behavior can be modeled using message sequence charts (MSC) and statemachines, respectively. The detailed timing behavior of a system can be modeled using timing-diagrams.

The semantics of UML structural diagrams can be defined using OCL (Object Constraint Language) [9], yet for the behavioral formalisms, UML does not provide, by choice, a unified semantics. However, UML provides specialized modeling profiles for specific domains, such as MARTE [10] (Modeling and Analysis of Real-Time Embedded systems) for real-time embedded systems, using modeling language extension mechanisms such as stereotypes and tags. MARTE provides modeling support for schedulability and performance analysis of RTES models. MARTE includes CCSL, a clock constraint specification language [11] (described below), for timed interpretation of UML/MARTE models for RTES, as shown in Fig. 2.1.
2.2 EAST-ADL

EAST-ADL is an emerging architecture standard for the development of RTES in the automotive domain. It is the result of several research projects such as ATESST (language definition)[12], TIMMO (timing description language) [13], and ATESST2 (model-based development together with support for verification and validation (V&V). The model-based development framework developed in the above described projects, contributes to improving communication among system stakeholders, as well as documentation, and V&V capabilities. This represents a shift from document-driven testing and simulation-based development methodologies, to a model-based paradigm. Besides, the framework is aimed to provide means for stakeholders to deal with the complexity and risk management of cooperative active safety systems. In EAST-ADL based development, the overall development process is organized into well-defined phases such as Vehicle, Analysis, Design, Implementation, and Operational levels. Further, the implementation phase is aligned with the corresponding standard, known as AUTOSAR (Automotive Open System ARchitecture) [14], facilitating a unified view of the system development across heterogeneous development environments.

In Fig. 2.2, we present an EAST-ADL model of a Brake-By-Wire system at Analysis Level. As a simplification, we have modeled only two out of the four wheels of the system. The model is composed of nine FunctionPrototype components as follows. The BrakePedalSensor provides information about the position of the pedal; the BrakeTorqueCalculator computes the global braking force based on the pedal position received from the BrakePedalSensor; the GlobalBrakeController calculates the local braking force by updating the global braking force based on the speed of the wheel. At each of the two wheels, the WheelSensor reads the rotation speed of the wheel and inputs to GlobalBrakeController and the corresponding ABS, which calculates the slip
rate to decide if the braking force can be applied without locking the wheel. Finally, the braking force is applied by the corresponding WheelActuator.

## 2.3 ProCom

The ProCom component model [15] is specifically developed to address the particularities of the embedded systems domain, including resource limitations and requirements on safety and timeliness. ProCom is organized in two distinct, but related, layers: ProSys and ProSave. In addition to the difference in granularity, the layers differ in terms of architectural style and communication paradigm.

In ProSys, the top layer, a system is modeled as a collection of communicat-
ing subsystems that execute concurrently, and communicate by asynchronous messages sent and received at typed output and input message ports.

Contrasting this, the lower level, ProSave, consists of passive units, and is based on a pipe-and-filter architectural style with an explicit separation between data and control flow. The data-flow is captured by data ports where data of a given type can be written or read, and the control-flow by trigger ports that control the activation of components. Data ports always appear in a group together with a single trigger port, and the ports in the same group are read and written together in a single atomic action.

2.4 CCSL

CCSL (The Clock Constraint Specification Language [11]), initially specified in an annex of MARTE, provides an expressive set of constructs to specify causality (both synchronous and asynchronous) as well as chronological and timing properties of the system models. The CCSL is formally defined making the specifications executable at the model level [16]. CCSL is a declarative language that specifies constraints imposed on the logical clocks (representing activation conditions) of a model. CCSL clocks refer to any repetitive events of the system and should not be confused with system clocks. A CCSL clock is defined as a sequence of clock instants (event occurrences). If \( c \) is a CCSL clock, \( c[k] \) denotes its \( k^{th} \) instant, for any \( k \in \mathbb{N} \). Below, we briefly describe the constraints used in this paper. A complete list of CCSL constructs can be found in André’s work [11]. CCSL constraints are of three kinds, as described below:

**Synchronous constraints.** Such constraints rely on the notion of coincidence of clock instants. For example, the clock constraint “\( a \ isSubclockOf b \)”, denoted by \( a \subset b \), specifies that each instant of the ‘subclock’ \( a \) must coincide with exactly one instant of the ‘superclock’ \( b \). Other examples of synchronous constraints are discretizedBy or excludes (denoted \( \# \)). The latter prevents two clocks from ticking simultaneously. The former discretizes a dense clock to derive discrete chronometric clocks. IdealClk, a perfect dense chronometric clock, is predefined in MARTE Time Library, and assumed to follow the ‘physical time’ faithfully (with no jitter).

**Asynchronous constraints.** They are based on instant precedence, a strict (\( < \)) or a non-strict (\( \preceq \)) form. The clock constraint “\( a \ isFasterThan b \)” (denoted by \( a \prec b \)) specifies that \( \forall k \in \mathbb{N}; a[k] \preceq b[k] \), that is, the \( k^{th} \) instant
of a precedes or is coincident with the \( k^{th} \) instant of \( b \). Alternation is another example of an asynchronous constraint. It is a form of bounded precedence. The constraint "\( a \) alternatesWith \( b \)" (denoted by \( a \sim b \) or \( a \triangleq_1 b \)) states that \( \forall k \in \mathbb{N}; a[k] \prec b[k] \land b[k] \prec a[k+1] \), that is, an instant of \( a \) precedes the corresponding instant of \( b \) which in turn precedes the next instant of \( a \).

**Mixed constraints.** Those that combine coincidence and precedence. The constraint "\( c = a \) delayedFor \( n \) on \( b \)" constrains \( c \) to tick synchronously with the \( n^{th} \) tick of \( b \) following a tick of \( a \). It is a mixed constraint since \( a \) and \( b \) are not assumed to be synchronous.

In Fig. 2.4, we present a simulation trace of the CCSL specification for the UML model illustrated in Fig. 2.1. Simulation traces capture the timing behavior of a given model, as specified by the corresponding CCSL specification.

### 2.5 Timed Automata & Model Checking

Timed automata (TA) is a theory for modeling and verification of real time systems [17]. A timed automaton is a finite-state automaton extended with a set of real-valued variables modeling clocks and (dense) chronometric time. UPPAAAL, extends timed automata (TA), originally introduced by Alur and Dill [17], with a number of features, such as, global and local (bounded) integer variables, arithmetic operations, arrays, and a C-like programming language. The tool consists of three parts: a graphical editor for modeling timed automata, a simulator for trace generation, and a verifier for symbolic (exhaustive) verification of a system modeled as a network of timed automata. A subset of CTL (computation tree logic) is used as the input language for the verifier.

A timed automaton (TA) is a tuple \( \langle L, l_0, C, A, E, I \rangle \), where \( L \) is a set of locations, \( l_0 \in L \) is the initial location, \( C \) is the set of clocks, \( A \) is the set of actions, co-actions and the internal \( \tau \)-action, \( E \subseteq L \times A \times \mathcal{B}(C) \times 2^G \times L \) is a
set of edges between locations with an action, a guard, a set of clocks to be reset, and $I : L \rightarrow B(C)$ assigns clock invariants to locations. A location can be marked urgent (u) or committed (c) to indicate that the time is not allowed to progress in the specified location(s), the latter being a stricter form indicating further that the next edge can only be traversed from the corresponding location. Synchronization between two automata is modeled by channels (e.g., $x!$ and $x?$) with rendezvous or broadcast semantics.

Semantically, the state of a TAn represents the current location (several in case of a network of TA) and the current valuation of all the variables. An enabled edge (when the guard becomes true) indicates a transition that may be taken in the current state. The semantics of a TAn defines transitions between locations as well as the time progress; an enabled edge at a current location may be taken (non-deterministically in case of many) when the invariant at the corresponding target location is preserved, otherwise no state-changing transition is fired and the time is allowed to progress as long as the invariant at the current location holds.

An automaton, described as above, is considered an abstract model of a timed system. Timed automata supports modeling of real-time concepts, such as periodicity, jitter, execution time, priority, urgency etc. In Fig. 2.5, we present an example automaton for modeling the timing behavior of a lamp manipulated by a user. When the ‘user’ non-deterministically ‘presses’ the button, the action is communicated by the corresponding timed automaton, through synchronization channels press! (send) and press? (receive), with
the automaton representing the lamp. The lamp automaton consists of three modes: OFF, DIM, and BRIGT. Mode-change behavior is triggered by ‘press’ actions as well as timing behavior. For instance, in DIM mode, the automaton moves to OFF mode, if no ‘press’ is received within 5 seconds of entering DIM.

*Model checking* is a formal technique for the automatic verification of properties of finite-state system models [18]. Given a model of a system, e.g. a finite state machine, denoted by $M$, model checking verifies (Fig. 2.6) whether the model satisfies a given requirement specification e.g. a temporal logic formula, denoted by $\rho$. This can be formally expressed as follows:

$$M, s_0 \models \rho$$

that is, given model $M$, and initial state $s_0$, $\rho$ holds.

In model checking, the above problem reduces to a reachability problem or to temporal logic verification that is, verifying if the expression $\rho$ is satisfied by a state (or set of states) in $M$, by algorithmically traversing the state transition graph of $M$. Further, model checking can produce a counter-example, that is, a partial execution trace leading to a system state where the property is not satisfied by the model. Several *model checkers* use timed automata as the input language (references), out of which UPPAAL and its variants (TIMES[19], UPPAAL PORT [20] etc.) are among the most popular.
Chapter 3

Problem Statement

In this chapter, we describe the research problem and the corresponding research goals, within the context of model-based development for RTES, addressed in this thesis. We will also present an overview of the included research papers.

3.1 Research Goals

Developing complex industrial real-time systems is a challenging task as the shorter development cycles demand reliable engineering methods, in addition to cost-efficient development by reusing existing system artifacts. In this context, model-based development is a promising solution that supports analyzability with respect to the specified functional and timing behavior of the system.

The design space of the model-based development for RTES consists of several artifacts such as requirement documents, specification models, analysis and design models, deployment models etc. These system artifacts or models, often organized into well-defined phases of system development, may be based on different formalisms as described below:

- System requirements are described by informal text-based documents.
- Specification models are based on expressive formalisms such as UML statemachines.
- Analysis-level models are based on time-triggered data-flow notations e.g. architecture description languages (ADLs).
• Design-level models are closer to implementation, with system components often implemented in code (e.g., in C, C++).

To achieve a mature model-based development towards predictable behavior of RTES, a continuous verification and validation (V&V) process, using advanced analysis techniques, is necessary. This is a non-trivial problem, as advanced analysis techniques such as simulation or model-checking, though tool-supported, require a semantic-aware modeling approach for developing expressive models that capture real-time aspects, as well as the timed causality behavior of the system. On the other hand, the current state-of-the-practice is dominated by an ad-hoc mixture of methods and tools, and system validation is mostly done using analytical methods such as schedulability analysis and testing, during the implementation phase. Most often, the above validation phase occurs too late in the design process, adding an overhead to the overall development. Hence, systematic design steps for managing system complexity, combined with advanced analysis techniques for verification of system properties during early phases of development, are necessary. In this context, we state the overall research goal of this thesis as follows:

Provide a semantic-aware model-based framework for verification and validation of real-time embedded systems, and enhance expressiveness and analyzability for structural and behavioral models.

The research problem and the corresponding research goal stated above represent a very wide scope in the development of real-time embedded systems. Hence, within this scope, we will present some specific research (sub)goals that are addressed in this thesis, and have been identified to serve our main goal.

A Design Support for Model-based Development. In the model-based development, the system artifacts at higher abstraction levels guide the design process of artifacts at lower level. However, behavior consistency of the system across these “layers-of-abstraction”, e.g. specification versus design, is a serious concern. For instance, semantically expressive models such as state machines can describe timed causality aspects of computation and communication, besides specifying the corresponding functionality. They use an aperiodic, event-triggered representation of the system behavior. Such a modeling paradigm facilitates the precise specification of system behavior in terms of abstract states or operational modes, events, mode-change transitions, etc. Also, the associated time model of the specifications is close to the real-world, that is, the dense
3.1 Research Goals

chronometric time. On the other hand, design models may use a different modeling paradigm, e.g. a time-triggered, data-flow based architectural model. For these models, the data is read from a buffer, according to a triggering condition generated by, e.g., a periodic clock or the occurrence of an event. The associated time model represents a discrete time implementation of the physical time. Hence, to obtain a mature model-based development framework for RTES, it is necessary to ensure the correctness of the design process that enables consistency of the system behavior across development phases that might use different modeling paradigms, and the associated time model to address the specific modeling needs.

Another common problem in the model-based development relates to the design models based on (real-time) components, referred to as component-based designs. These models are generally developed using a real-time component-model with primitive components representing the system functionality implemented in code. However, real-time components exhibit recurring behavior features such as run-to-completion, history, delay, etc. This execution behavior of the components is often hidden within the implemented code, hindering reusability of the component, as well as analysis of corresponding design models. However, a model-based approach for components, based on the separation of functionality and timing behavior, enhances both reusability and analyzability of component-based designs.

With respect to the specific research problem, described above, we state the following research goals that we address in this thesis:

**Research Goal G1a.** Provide a systematic design support for correlating structural and behavior models of real-time embedded systems.

**Research Goal G1b.** Apply separation-of-concerns for component-based designs to achieve analyzability.

**A Semantic Basis to support Verification and Validation.** The increasing complexity and safety-criticality of real-time embedded systems, in domains such as automotive and avionics, stresses the need for applying rigorous analysis techniques, e.g. model-checking, to verify system behavior with respect to both functionality and timing. In the model-based development, an early-phase verification and validation process to establish system timing behavior
is much desired. While analytical methods such as schedulability analysis are often applied late into the design process, an early validation of the system’s timed causality behavior is very useful for depicting unfeasible execution paths, or unreasonable assumptions. However, to provide rigorous verification and validation, the system models need to capture real-time characteristics such as urgency, priority (with respect to timing), as well as represent the timed causality semantics of the underlying model-of-computation. The model-based development frameworks often employ architectural languages, such as AADL and EAST-ADL, to model system structure and execution behavior. Also, behavioral models such as UML statemachines (or specific extensions for real-time domain, such as MARTE mode behaviors) are used to specify the abstract behavior of the system. While architectural/component models support explicit modeling of real-time concepts such as periodicity, jitter etc, the underlying model-of-computation is however implicit or under-specified. Hence, the formalization of real-time component models as well as the explicit specification of the associated model-of-computation, is a one pre-requisite to enable formal verification. On the other hand, behavior models at higher abstraction levels of system development, representing early-phases of the latter, can be very expressive. While architectural models are commonly based on the discrete notion of time, the behavior models may refer to different time model, e.g., the dense time model. However, expressive models are often not amenable for the rigorous verification due to problems such as undecidability and state-space explosion.

With respect to the research problem described above, we state the following research goal:

**Research Goal G2.** Provide a formal semantic basis to specify real-time and timed causality aspects of structure and behavior models, to enable exhaustive verification by model checking.

**Research Validation.** The research contributions with respect to the research goals as described above should be validated in terms of applicability and usability for industrial real-time systems. Hence, we state the following research goal.

**Research Goal G3.** Validate the proposed semantic-aware model-based framework by applying it on a relevant industrial case-study.
3.2 Research Method

We outline the research methodology adopted in carrying out the research that addresses the research goals stated in the previous section. The research methodology consists of the following research activities:

- Problem formulation and literature-survey
- Research collaboration
- Industrial case-studies and validation
- Research publications

During the initial phase of the research, literature survey has been carried out to identify the research issues in component-based development approaches for embedded systems. As a hands-on approach, existing techniques have been applied for an industrial case study, that is, a turntable system example. This has provided an opportunity to identify the limitations of existing approaches for component-based development. As a result, we have published the work as Paper A of the thesis. As an extension of this work, we have proposed a pattern-based design support published as Paper C. These papers address the first research goal of the thesis, presented in the previous section.

With the initial phase described above completed, we have been able to identify the modeling issues and the limitation in providing verification support, due to lack of expressiveness of the models. We have investigated existing real-time formalisms that can be integrated in the model-based development of RTES. As a result, we have chosen CCSL and timed automata, which are real-time specification and modeling formalisms, as the suitable languages to enhance the expressiveness of RTES models. We have initiated extensive research collaborations with academic researchers at INRIA, France, and industrial researchers at ABB, Sweden & Norway. As a result, we have published Paper B, Paper D, Paper E and Paper F. These papers address mainly the second research goal of the thesis, thus providing a semantic basis to develop verification support for RTES.

As a final phase of the research work, we have chosen to validate the research contributions using an industrial-strength system, a simplified wind turbine system. The result of the validation is presented in Paper G.
3.3 Included Papers - An Overview

In this thesis, we have included seven research papers (listed chronologically) that address the research goals presented in the previous section. The mapping between the research goals and the papers is summarized in Table 3.1. Next, we briefly overview the seven papers included in this thesis.

Table 3.1: Illustration of how the included papers A-G contribute to the research goals

<table>
<thead>
<tr>
<th>Papers</th>
<th>G1a</th>
<th>G1b</th>
<th>G2</th>
<th>G3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A, C</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>D, E, F</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>


Abstract: Designers of industrial real-time systems are commonly faced with the problem of complex system modeling and analysis, even if a component-based design paradigm is employed. In this paper, we present a case-study in formal modeling and analysis of a turntable system, for which the components are described in the SaveCCM language. The search for general principles underlying the internal structure of our real-time system has motivated us to propose three modeling patterns of common behaviors of real-time components, which can be instantiated in appropriate design contexts. The benefits of such reusable patterns are shown in the case-study, by allowing us to produce easy-to-read and manageable models for the real-time components of the turntable system. Moreover, we believe that the patterns may pave the way toward a generic pattern based modeling framework targeting real-time systems in particular.
Comment: In this paper, I have specifically contributed by proposing the behavior modeling patterns for components and in applying the proposed patterns to the case study. All authors participated in writing and paper discussions. The paper addresses the research goals G1a and G1b.

Aneta Vulgarakis, Jagadish Suryadevara, Jan Carlson, Cristina Seceleanu, and Paul Pettersson. In proceedings of 35th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Patras, Greece, August, 2009

Abstract: ProCom is a new component model for real-time and embedded systems, targeting the domains of vehicular and telecommunication systems. In this paper, we describe how the architectural elements of the ProCom component model have been given a formal semantics. The semantics is given in a small but powerful finite state machine formalism, with notions of urgency, timing, and priorities. By defining the semantics in this way, we (i) provide a rigorous and compact description of the modeling elements of ProCom, (ii) set the ground for formal analysis using other formalisms, and (iii) provide an intuitive and useful description for both practitioners and researchers. To illustrate the approach, we exemplify with a number of particularly interesting cases, ranging from ports and services to components and component hierarchies.

Comment: In this paper, I have contributed to the formalization of architectural elements of ProCom. All authors participated in writing and paper discussions. The paper partially addresses the research goal G2.

An extended version of the paper is available as a technical report [21].


Abstract: The development of embedded systems often requires the use of various models such as requirements specification, architectural (component-based), and deployment models, across different phases. However, there exists little design support for obtaining suitable component-based designs that satisfy specified requirements and timing constraints. In order to provide guided support for the design process of embedded systems, we introduce several component templates, referred as patterns, which we also formally verify against
relevant properties. To illustrate the usefulness of the approach, we have applied the proposed patterns to obtain a component-based design of a temperature control system.

Comment: I was the main driver and principal author of this paper. All authors participated in writing and paper discussions. The paper addresses the research goals G1a and G1b.


Abstract: It is critical to analyze characteristics of real-time embedded systems such as timing behavior early in the development. In the automotive domain, EAST-ADL is a concrete example of model-based approach for architectural modeling of real-time systems. The Timing Augmented Description Language V2 (TADL2) allows the specification of timing constraints on top of EAST-ADL models. In this paper we focus on TADL2 timing constraints and propose solutions to execute and verify such timing constraints. The formal semantics of the considered timing constraints is given as a mapping to the Clock Constraint Specification Language, a formal language that implements the MARTE Time Model. Then verification is performed through a transformation into Timed Automata such as implemented by UPPAAL. The whole process is illustrated on a Brake-By-Wire application.

Comment: In this paper, I was mainly responsible for proposing a formal verification technique using timed automata and UPPAAL model-checking tool. All authors participated in writing and paper discussions. The paper addresses the research goals G2 and G3.


Abstract: Systematic and formal development approaches for safety- and mission-critical systems are of increasing importance. These systems are often implemented as periodically triggered control systems, to ensure deterministic
and analyzable timing behavior. However, integrating timing ‘constraints’ in the
development process remains a challenging task. For instance, these constraints
should be formally verified as consistent as well as feasible with respect to
the system design. In this paper, we present a timed automata based valida-
tion approach for EAST-ADL timing constraints for periodic control systems.
The constraints are specified using CCSL – the Clock Constraint Specification
Language, and transformed into timed automata, to enable formal verification
with UPPAAL model-checker. The resulting timed automata specification can
be simulated and verified for the formal validation of the timing constraints.
Further, the transformed specification model can be easily integrated with the
Corresponding model of the actual system design, also specified in CCSL, thus
Extending verification aspects. The proposed approach is demonstrated using
the timing constraints for an Anti-lock Braking System (ABS) example.

Comment: The paper addresses the research goals G2 and G3.

Paper F. Verifying MARTE/CCSL Mode Behaviors using UPPAAL. Ja-
gadish Suryadevara, Cristina Seceleanu, Frédéric Mallet and Paul Pettersson,
In proceedings of 11th International Conference on Software Engineering and
Formal Methods (SEFM), Madrid, Spain, September 2013.

Abstract: In the development of safety-critical embedded systems, the ability
to formally analyze system behavior models, based on timing and causality,
helps the designer to get insight into the systems overall timing behavior. To
support the design and analysis of real-time embedded systems, the UML mod-
eling profile MARTE provides CCSL – a time model and a clock constraint
specification language. CCSL is an expressive language that supports specifi-
cation of both logical and chronometric constraints for MARTE models. On the
other hand, semantic frameworks such as timed automata provide verification
support for real-time systems. To address the challenge of verifying CCSL-
based behavior models, in this paper, we propose a technique for transforming
MARTE/CCSL mode behaviors into Timed Automata for model-checking using
the UPPAAL tool. This enables verification of both logical and chronometric
properties of the system, which has not been possible before. We demonstrate
the proposed transformation and verification approach using two relevant exam-
pies of real-time embedded systems.

Comment: I was the principal author and main driver of this paper. All authors
participated in writing and paper discussions. The paper addresses the research
goals G2 and G3.


**Abstract:** In the development of real-time embedded systems (RTES), the ability to formally analyze system artifacts, such as structure and behavior models, helps the design engineers to get insight into the overall functional and timing behavior the system. In this case study paper, we present our experience in applying formal verification and validation techniques, we developed earlier, for an industrial case study, namely, a windturbine system (WTS). We demonstrate the formal verification benefits against traditional simulation and testing practice prevailing in the industry. However, we also present some design trade-offs and challenges we have identified through the case-study, which needs to be addressed by formal method researchers, for instance, modeling the expressiveness of the system artifacts and system properties w.r.t existing limitations in providing rigorous verification, such as model-checking, for industrial systems.

**Comment:** I was the principal author and main driver of this paper. All authors participated in writing and paper discussions. The industrial partners provided the system description and simulation data, whereas my supervisors participated in paper writing, discussions of the solution and feedback. The paper is a validation paper addressing research goal G3.
Chapter 4

Research Contributions

In this chapter, we present an overview of the main contributions of this thesis work. The contributions are divided into three parts as follows:

i. A pattern-based design support for component-based development of RTES.

ii. A semantic basis to support verification of architectural and behavior models of RTES.

iii. Validation of the research contributions using example systems and an industrial case-study from ABB CRC, Sweden & Norway.

4.1 Pattern-based Design Support for RTES

In this thesis, we adopt a heterogeneous modeling approach (see Fig. 1.2 in Chapter 1) for the design and verification of RTES as follows: for structural modeling, we use EAST-ADL architecture modeling language as well as ProCom, a complementary real-time component model. Although both EAST-ADL and ProCom describe the architecture of a system, they can be used together in a complementary way: an EAST-ADL description can be refined by using a ProCom-based design, to achieve model reusability and analyzability. For behavior modeling, we use UML statemachines to specify component behaviors with respect to functionality and timing. We also use the UML/MARTE *mode behaviors* to specify abstract system behavior in terms of operational *modes*, mode-change *transitions*, and timing behavior.
The search for general principles underlying the structural modeling of RTES has motivated us to propose a ‘pattern-based design support’ to develop component-based designs and corresponding functional models. In this thesis, we propose two classes of modeling patterns:

i. For modeling recurrent behaviors of real-time components as abstract behavior specifications that also enable transformations to analyzable models.

ii. For transforming higher-level behavior models, e.g., mode behavior specifications, to component-based designs.

4.1.1 Modeling real-time components

To support verification of component-based designs, a pre-requisite is to produce manageable and easy-to-grasp design models for components and their compositions. For instance, by using behavioral models, it should be possible to automatically generate code (e.g. in C or C++) as well as facilitate easier transformation into real-time semantic domains, such as timed automata, for verification. This has motivated us to extract real-time aspects, such as, run-to-completion, history, and execution-time patterns, and represent them using a finite-state-machine like notation, which lets us apply the patterns at high-levels of software development, while simplifying the produced models.

• **Run-to-Completion Pattern.** As shown in Fig. 4.1, the pattern abstracts the read-execute-write semantics of a real-time component. In Fig. 4.2, we show the corresponding timed automata model for the behavior. The pattern hides the more detailed semantic representations, which facilitates easy-to-understand behavior models and provides a basis for automatic generation of code.
4.1 Pattern-based Design Support for RTES

• History Pattern. The pattern represents an abstract modeling of event-based execution behavior for real-time components with respect to events, control states etc. Further, the pattern eliminates non-deterministic time delays in the execution of component behaviors.

• Execution-time Pattern. The pattern represents the timing attributes such as deadline or execution time associated with component behaviors.

For further details about component patterns, we refer to the included PAPER A of the thesis. We also demonstrate the benefits of the pattern-based approach in modeling and verification of component-based designs, using an illustrative turntable system case-study.

4.1.2 Pattern-based component designs

The model-based development of RTES often involves different kinds of models, such as, behavior specification, architectural models, component-based designs etc. For instance, in Fig. 4.3, we present a MARTE/CCSL mode behavior model of an example temperature control system (TCS). It describes the operational modes of TCS, where a mode represents causally related atomic behaviors, e.g. MeasureTemp, DeleteRod of TCS. The timed causality semantics of a mode can be formally specified in CCSL, as shown in Fig. 4.3.

During the system development, the atomic behaviors are implemented as elementary components, which can be composed to form the overall system design. To assist the transformation of a mode behavior specification into predictable and valid composition of elementary components, we propose the following design patterns, instantiated by using ProCom.

• Discrete Clock Pattern. The pattern corresponds to a CCSL clock and represents the periodic triggering of a component due to causality. In
Fig. 4.4, the pattern is exemplified for periodic (chronometric) triggering of a component called 'Clock'. It represents a chronometric clock and the period is specified with respect to the finer-grained system clock. As shown in the figure, both 'clocks' are modeled as elementary components and connected using a trigger connection.

- **Timer Pattern.** It implements a timeout mechanism for component-based designs. When triggered by the system clock, the time is internally measured (using a state variable) until the specified duration is expired. The output, sigTimeOut, is indicated as both data and control. This facilitates using the timeout as either sampled data or reactive trigger (useful design choice with respect to the specified timing constraints).

- **Periodic Behavior Pattern.** It represents the causality and precedence constraint between two periodically triggered behaviors, i.e., components in a mode, as specified using the CCSL constraint alternatesWith, as shown in Fig. 4.3. The components are composed according to the pattern implementation to guarantee the associated timing behavior.

- **Controller Pattern.** It corresponds to the event-triggered control behavior of a system, specified by a mode behavior model (see Fig. 4.3). The pattern implements the event-triggered behavior using a dedicated component that periodically triggers the subsystem corresponding to the current 'active' operational mode of the system.
The timing behavior of the above proposed patterns has been verified by transforming the corresponding ProCom designs into timed automata, based on the ProCom formal semantics (as presented in PAPER B of the thesis).

The patterns described above support the development of component-based design models with predictable timing behavior, as we demonstrate by transforming the mode behavior specification of the TCS example, given in Fig. 4.3, into the corresponding design using ProCom. For further details of the patterns we refer to the included PAPER C of the thesis.

In this section, we have presented two kinds of component-based patterns that are aimed at enhancing the model-based development for RTES. The first set of patterns facilitate abstract, easy-to-understand, as well as analyzable models for real-time component behaviors. On the other hand, the second set of patterns correlate abstract behavior specifications, based on a synchronous and dense-time model, with the time-triggered, data-flow based designs. Hence, the pattern-based design support presented in this thesis provides a promising solution towards a systematic development of structural and behavioral models of RTES.

4.2 CCSL and TA Semantics for Verification and Validation of RTES Models

System artifacts such as architectural models and component-based designs refer to the time-triggered, data-flow based model-of-computation. In this paradigm a system is composed out of elementary functional components whose activations and communications are based on causality and timing. As intended, a
formal semantics of structural models facilitates unambiguous system designs. However, formal semantics alone is not sufficient to represent the underlying model-of-computation, it requires additional expressive formalisms to be aligned with the system models. Similarly, behavioral models, such as UML statemachines, though expressive, require an explicit time model to capture the associated timed causality aspects. On the other hand, expressive models often lead to complex formalizations and verification intractability, when applying formal techniques such as model checking.

To address the formal semantics and expressiveness for structural and behavioral models of RTES and also achieve tractability in formal verification using model checking, in this thesis, we propose the following light-weight formal techniques:

i. A formalization approach for real-time component models to support the design process as well as provide a basis for model transformations.

ii. Application of CCSL constraints for architectural descriptions and transformation to timed automata for verification.

iii. A transformation approach for MARTE/CCSL mode behaviors into timed automata to enable model checking of CCSL-based specifications.

4.2.1 Formalizing real-time component models

In the model-based development of RTES, component models play a significant role in the design process by providing building blocks, such as components, connectors, and ports, as well as general principles for composing components towards developing a complete system design.

In this thesis, we present a formalization approach for real-time component models, demonstrated using ProCom. As shown in Fig. 4.5, we have proposed a finite state machine (FSM) based formalism, to describe the formal semantics for architectural elements of ProCom. The proposed formalization, within the context of ProCom, clarifies modeling issues such as component activations, data/control flow semantics, as well as timing semantics with respect to real-time features, such as urgency, priority and timing. We give an overview of the formalization below.

Notation: The proposed FSM notation (Fig. 4.5) has graphical appeal and is simpler than the corresponding timed automata model, as it abstracts real-valued variables and synchronization channels. For instance, a delay interval
4.2 CCSL and TA Semantics for Verification and Validation of RTES Models

<table>
<thead>
<tr>
<th>Informal</th>
<th>FSM</th>
<th>TA</th>
</tr>
</thead>
<tbody>
<tr>
<td>urgent transition</td>
<td>↑</td>
<td>c?</td>
</tr>
<tr>
<td>urgent transition with priority</td>
<td>↑</td>
<td>b?</td>
</tr>
<tr>
<td>non-urgent transition</td>
<td>*</td>
<td>↑</td>
</tr>
<tr>
<td>non-urgent transition with priority</td>
<td>*</td>
<td>d?</td>
</tr>
<tr>
<td>urgent transition with guard x==5 and update x=x+1</td>
<td>↑</td>
<td>↑</td>
</tr>
</tbody>
</table>

Initial state

State

State with delay interval \([n_1, n_2]\)

Figure 4.5: ProCom semantics notation in FSM and timed automata.

\([n_1, n_2]\) is specified using the state with delay node (Fig. 4.5) that corresponds to timed automata location with invariant \(clk_i \leq n_2\) and guard \(clk_i \geq n_2\) to force the corresponding timeout transition, with \(clk_i\) being a clock variable. Similarly, to specify urgency and priority we use transitions with special annotations, which correspond to timed automata edges with synchronization channels.

Some of the modeling issues that the ProCom formalization gives precise meaning to are:

- The data and triggering of an output group of a service must always be produced atomically, and each of the service output port groups must have been activated exactly once before the service returns to idle state.
- All the data must arrive to its end destinations before the trigger signal. This rule should also hold in cases when data is transferred through a connector.
- Bridging the two communication paradigms in the sub-languages of ProCom: message passing in ProSys and pipes-and-filters in ProSave.

In Fig. 4.6 (a), we present the clock element of ProCom. A clock serves for generating periodic triggers. For instance, a ProSave component can be activated by receiving a periodic trigger from a clock with appropriate period. The formal semantics of a ProSave clock with period \(P\) is shown in Figure 4.6.
Figure 4.6: (a) A ProSave clock with period $P$ and (b) its formal semantics.

(b). Given the above description, it follows that the formal semantics complies to the informal semantics of a clock in ProCom.

As described in this section, we intend to make the formalization of real-time component models intuitive and easy-to-understand, so that it serves as a basis both for engineers using ProCom, as well as researchers developing analysis techniques, model-transformation tools etc, using the ProCom framework. For further details of the ProCom formalization, we refer to the included PAPER B of the thesis.

4.2.2 Architectural Descriptions using CCSL and TA

Analyzing non-functional aspects, such as timing behavior, are central to model-based development of RTES. Structural models such as architectural designs provide a suitable abstraction to apply verification techniques like model checking. In this thesis, we have used EAST-ADL to describe architectural designs. EAST-ADL is associated with sub-languages such as TADL2 (Timing Augmented Description Language) for specification of timing constraints.

In this thesis, we propose verification techniques for EAST-ADL and TADL2 specifications, by transforming the latter into timed automata for verification using UPPAAL. As shown in Fig. 4.7, the timing constraints for the Brake-By-Wire (BBW) architectural design are specified using TADL2 timing constraints, as below:

- **DelayConstraint** imposes duration bounds (minimum and maximum) between two events, that is, the source (stimulus) and the target (response). For the BBW example (Fig. 4.7), TC1 represents delay constraint for event chain consisting of BrakePedalSensor (Input) to Rear-RightBrake (Output). Further, the event chain contains sub-event chains corresponding to timing constraints TC5, TC8, etc.
- **SynchronizationConstraint** is a constraint on a set of events.
4.2 CCSL and TA Semantics for Verification and Validation of RTES Models

Brake By Wire Functionality

TC1(Reaction) $X \in [0\text{ms} - 330\text{ms}]

TC5(Reaction) = 0.275 \times X

TC10(Synchronization) = 5 \text{ ms}

TC3(Periodic) = 20 \text{ ms}

TC8(Reaction) = 10\text{ms}

Figure 4.7: EAST-ADL and TADL2: Schematic view of Brake-By-Wire architectural design and timing constraints.

All events must occur within a timing window, specified by the tolerance attribute, i.e., maximum allowed skew between the events. For the BBW example (Fig. 4.7), TC9 and TC10 represent an input synchronization constraint (for GlobalBrakeController component), and an output synchronization constraint (for BBW actuator outputs) respectively.

- RepetitionConstraint specifies periodic triggering of a component. It is specified using timing attributes upper and lower. For BBW example (Fig. 4.7), TC3 represents a repetition constraint.

We transform TADL2 specifications into timed automata, as shown in Fig. 4.8: an event is mapped to a synchronization channel; event chains and timing constraints are modeled as timed automata. For further details of transformation and verification, we refer to PAPER D and E included in the thesis.
4.2.3 Semantic Basis for MARTE Mode Behaviors using CCSL and Timed Automata

MARTE mode behaviors can be used to specify the abstract behavior of a system in terms of operational modes and corresponding mode-change behavior. In this view, a mode represents an operational segment that is characterized by a configuration of system entities. For instance, during ‘TakeOff’, ‘Flying’ and ‘Landing’ modes of an aircraft, different parts of the control system may be active in different modes.

In this thesis, we associate CCSL specifications to MARTE mode behaviors, to take advantage of the underlying MARTE time model for specification and analysis. For instance, logical (of synchronous and asynchronous nature) as well as timing (chronometric) properties of a system in a mode can be precisely specified using CCSL constraints. In Fig. 4.9, we present a mode behavior specification, of an example temperature control system (TCS), and its corresponding timed automata representation. TCS operates in two operational modes as follows: After 100 s in Diagnostic mode, the system changes to Control mode. However, the mode-change from Control to Diagnostic is triggered by an event occurrence, indicating the sensed temperature in the reactor is within the specified limits. The functional and timing properties of TCS are as follows:

TCS1: Diagnostics is always followed by Reconfig.

TCS2: The behavior of Reconfig is ’extended’ by StatusUpdate, only
4.2 CCSL and TA Semantics for Verification and Validation of RTES Models

When there is a change in the control rod configuration.

TCS3: PeriodicSense executes periodically with a period of 10 s.

TCS4: PeriodicSense is followed by InsertRod or RemoveRod but not both.

TCS5: At most two rods can be used in sequence, for cooling the reactor core.

As shown in (Fig. 4.9), we can use CCSL to specify the functional and timing properties of TCS within mode behavior specification. The actions in TCS modes are represented by CCSL logical clocks as follows: Diagnostics: d, Reconfig: c, StatusUpdate: s, PeriodicSense: p, InsertRod: i, and RemoveRod: r. Using these logical clocks we can specify the causality and timing behavior associated with the actions. For instance, the property TCS1 is specified using CCSL operator alternatesWith, as \[ s \subset c \]

In this thesis, we propose a transformation approach from MARTE/CCSL mode behavior models into timed automata, as shown in Fig. 4.9 using the TCS mode behavior specification. The transformation consists mainly of three steps as described below:

- Using the labelled transition system (LTS) based semantics of CCSL constraints, we obtain the synchronization product of CCSL

Figure 4.9: Temperature Control System: MARTE/CCSL Mode behavior specification to timed automaton.
constraints in a mode. The synchronized product is transformed into a timed automaton.

- The transformed mode automata are connected using *entry*, *exit* transitions corresponding to the mode change behavior.
- Additional issues such as history, timing are mapped into timed automata using invariants and history variables.

By using the transformed automata models, we verify both logical and chronometric properties of the system, such as *deadlock paths* for CCSL clocks, minimum and maximum interval times, etc. For further details of the transformation and verification results, we refer to PAPER F included in the thesis.

### 4.3 Industrial Case-Study: A Wind Turbine System - Formal Modeling and Verification

To demonstrate the benefits of the proposed techniques, we apply them on an industrial case-study system, a wind turbine system developed in ABB CRC.

#### 4.3.1 Wind Turbine System (WTS) - An Overview

The WTS is developed within the context of iFEST (industrial Framework for Embedded Systems Tools) project, an ARTEMISIA project by ABB CRC, Sweden. In the iFEST project, the system development has been carried out by adopting the V-model software development approach as follows. During the *Requirement and Analysis* phase, the WTS requirements (both functional and timing) have been documented. For the *Design* phase, component- and model-based approaches have been adopted, using the Simulink/MATLAB modeling environment. Subsequently, the implemented system, a combined FPGA and CPU solution, has been deployed on a heterogeneous hardware platform (Xilinx ZynQ 7000 product family).

In Fig. 4.10, we present a high-level view of WTS system as made of *Plant*, *Controller*, and Environment representing *Wind Profile* and *Resistive Load*. The *plant* subsystem consists of three components; *Servo*, *Rotor*, and *Generator*. The pitch of the turbine, determined by the *Controller* (described below), is actuated by the *Servo*. The *Rotor* produces the required *torque* to maximize the angular speed of the *Generator* (which produces the final voltage), based on the pitch value as well as the current wind speed (we assume a fixed *resistive load*).
The \textit{Rotor} optimizes the produced torque value based on the current angular speed of the \textit{Generator}.

The \textit{controller} subsystem for WTS consists of four main components: the \textit{Filter}, the \textit{Main Controller}, the \textit{Pitch Controller}, and the \textit{Park and Brake Controller}. The \textit{Filter Subsystem} is responsible for transducing, filtering and scaling the wind signal and plant signal (for instance the rotational speed of the turbine), which are used by the \textit{Main Controller} and the \textit{Pitch Controller}. Based on the inputs received through the \textit{Filter}, the \textit{Main Controller} directs the overall control. It oversees the performance and operations of the turbine in order to maximize the energy production and prevent any damage to the plant. Based on the wind and plant state, the controller determines the operational mode (i.e. \textit{park}, \textit{start}, \textit{generate}, or \textit{brake}) of the turbine. The \textit{Pitch Control} calculates the proper pitch i.e. angle to steer the rotor blades when the turbine starts up or generates power. The \textit{Pitch and Brake} controller determines if the turbine needs to brake or park, to ensure the safety of the wind turbine, for instance, during wind turbulences.

4.3.2 Formal Modeling using CCSL and Timed automata

Keeping in view the complexity of WTS, in particular the plant behavior with respect to the physical dynamics of the \textit{environment}, we develop formal models as described below:

- CCSL and Timed automata based semantic models for both \textit{plant} and \textit{controller} subsystems have been developed. The formal models represent the timed causality semantics of the corresponding \textit{model-of-computation} for both \textit{plant} (synchronous, reactive and
instantaneous) and the controller (asynchronous, time-triggered).

- As an effective abstraction for plant behavior, we integrate the simulation results to construct the partial functions (input values to output values) representing plant components. As a novel approach, addresses the limitations of modeling the complex behavior of the plant behavior for the wind turbine system.

- For the modeling of the main controller components, we apply the pattern-based approach e.g. the controller, run-to-completion, history patterns proposed in the thesis. The system’s interface behavior such as component activations are associated with executions of internal behavior; the latter is modeled using the real-time component behavior patterns proposed in the thesis.

In Fig. 4.11, we present an EAST-ADL-based plant model for WTS. The timing behavior and execution semantics of the components, e.g. ROTOR, are specified using CCSL as shown in Listing 4.1. This refers to the interface behavior of the component with respect to the underlying model-of-computation based on instantaneous executions. Similarly, we specify the execution semantics of the controller components, which are based on a time-triggered, data-flow based model-of-computation.

As a next step in the modeling process, we transform the CCSL specifications of the real-time components into timed automata. As shown in Fig. 4.12, the CCSL constraints for the ROTOR component (Listing 4.1) are transformed into timed automata. The automata can be composed to obtain a single timed automaton model for the ROTOR component. Similarly, we have developed timed automata models for remaining components of WTS. Please note that the timed automata in Fig. 4.12.(c) represents the partial function, i.e., writeTorque(), constructed using the simulated values of the WTS model. For controller components this corresponds to the execution of the associated behavior model (e.g. Fig. 4.13).
4.3 Industrial Case-Study: A Wind Turbine System - Formal Modeling and Verification

```plaintext
| CCSL clock RT_in;                      | // read (input) instants               |
| CCSL clock RT_out;                    | // write (output) instants             |
| CCSL clock RT_omega;                  | // activation (trigger) instants       |
| CCSL constraint                       |                                          |
| RT_omega                              | RT_in;                                 |
|                                        | // RT_omega coincidesWith RT_in         |
| RT_in                                 | RT_out;                                |
|                                        | // RT_in coincidesWith RT_out           |
```

Listing 4.1: CCSL specification of ROTOR.

Figure 4.12: Timed automata modeling for ROTOR: (a) RT_omega \(\square\) RT_in (b) RT_in \(\equiv\) RT_out (c) Modeling the partial function - An abstraction of ROTOR computation.

4.3.3 Formal Verification of WTS and Lessons Learned

For the Verification and Validation (V&V) of WTS, we use model-based techniques as follows: (i) simulation of the WTS functionality using the Simulink environment, and (ii) automatic model-based test-case generation with MaTeLo tool. However, the above techniques are not sufficient to ensure system predictability with respect to all possible system executions, hence formal verification is desirable, as described below.

Verification of functional properties: Verifying functional properties gives insight into the overall system (architectural) design. For instance, in the WTS

Figure 4.13: Functional behavior of the MainControl component.
case, it is useful to verify the following property: “if the wind speed is within the prescribed limits, the controller eventually moves to Generate mode”. The property can be formulated as a liveness property or leads to property (denoted by \( \rightarrow \), implemented as \( \Rightarrow \) in UPPAAL), as below.

\[
\text{(ws} \geq 5 \&\& \text{ws} \leq 20) \Rightarrow \text{state} = 2
\]  \hspace{2cm} (4.1)

**Verification of safety-critical properties:** One of the safety-critical requirements for the WTS is to fulfill the following property: “the wind turbine moves to Park mode, within 30s from detecting that the wind speed has crossed the upper limit of 20m/sec”. To verify the property (with respect to simulation data), we construct an observer automaton, for the property, as shown in Fig. 4.14, compose the observer with the system model, and verify that the corresponding invariant, property (4.2), holds for the composed model. Note that the urgent channel ‘U!’ forces the change from location B to A without any further delay, when the corresponding edge guard holds.

\[
A[]\text{obs.B implies } x \leq 30
\]  \hspace{2cm} (4.2)

![Observer Automaton](image)

Figure 4.14: An observer automaton to verify the safety-property: \( A[]\text{obs.B implies } x \leq 30 \)

**Lessons Learned.** While applying the modeling and verification techniques for the industrial case study as described above, we have made the following observations: while verification models are expressive in terms of system structure and precise timing behavior, simulation models are suitable to specify the plant and the environment (wind profile) modeling for WTS. Thus, the proposed verification approach, in this case, provides an enhanced simulation-based validation. However, some limitations of the approach do exist. The exhaustiveness of the verification is limited to partial functions constructed by using specific instance(s) of simulation. Hence, the approach may be similar to testing-based analysis (albeit model-based). Hence, we need strategies, e.g. choosing a suitable simulation step and data profiles, to generate simulation data for the system properties to be verified. Further, it may be noted that
the simulation-based verification approach presented above may be suitable for data-intensive control systems (e.g., hybrid systems), such as the wind turbine system case study presented in the thesis (PAPER G). On the other hand, control-intensive systems may be exhaustively modeled and verified, by using model-checking, independently of simulation.

4.4 Discussion

In the previous sections, we have presented the research contributions of this thesis with respect to the research goals presented in Chapter 3. We revisit the research goals below.

The first research contribution described in Section 4.1, that is Pattern-based Design Support for RTES addresses the research goals G1a and G1b of the thesis, recollected below.

**Research Goal G1a.** Provide a systematic design support for correlating structural and behavior models of real-time embedded systems.

**Research Goal G1b.** Apply separation-of-concerns for component-based designs to achieve both reusability and analyzability.

The research goal G1a is addressed through the component-based design patterns proposed in the thesis. The patterns correlate higher-level behavioral models and component-based designs that capture the overall timing behavior as well as functionality. The research goal G1b is addressed through the behavioral patterns proposed in the thesis. The patterns represent a separation-of-concerns approach for the abstract representation of functionality, execution behavior and timing. Further details of the contributions that address the above research goals can be found in Paper A and Paper C.

The second research contribution described in Section 4.2, that is ‘CCSL and TA Semantic Basis for Verification and Validation of RTES models’ addresses the research goal G2 of the thesis, presented in the previous chapter and recollected below.

**Research Goal G2.** To provide a formal semantics basis to specify real-time and timed causality aspects of structure and behavior models to enable exhaus-
The research goal \(G_2\) is addressed by providing a CCSL and timed automata based semantic basis for structural and behavioral models of RTES. The CCSL and timed automata are expressive and powerful semantic domains, as specification and modeling languages, respectively. The time model in CCSL is discrete and interactions are synchronous. On the other hand, the timed automata modeling is based on dense time and interactions are asynchronous. In this thesis, we have integrated CCSL and timed automata, thus combining the expressive power of CCSL with efficient chronometric modeling and verification using timed automata and UPPAAL. Further details of the contributions addressing the above research goals can be found in \textbf{Paper B}, \textbf{Paper D}, \textbf{Paper E} and \textbf{Paper F}.

The third and final research contribution described in Section 4.3, that is ‘Industrial Case-Study : A Wind Turbine System - Formal Modeling and Verification’ addresses the research goal \(G_3\) of the thesis, as recollected below.

**Research Goal G3.** To validate the proposed semantic-aware model-based framework by applying it on an industrial case-study.

The research goal \(G_3\) is partially addressed by using representative examples of RTES within the included papers, on which the framework has been applied. However, the major research validation comes from applying the proposed design and verification techniques on an industrial wind turbine system. During this work, we have observed both the benefits and limitations of the proposed techniques, as documented in the included papers, in particular \textbf{Paper G}. For instance, the proposed modeling techniques facilitate obtaining expressive models, whereas the benefits of formal verification can be combined with simulation-based validation techniques for enhanced system model analysis. Further details on research validation can be found in \textbf{PAPER G}. 
Chapter 5

Related Work

In this chapter, we present an overview of related works and compare the corresponding approaches with the design and analysis methods proposed in this thesis.

5.1 Design Support for Embedded Systems

The Statemate toolkit [22] is an early working environment for the development of complex reactive systems. Modularity of the system development is provided in terms of different views, such as structure, functionality, and behavior. Our approach for behavior specification of components (modules in Statemate) is similar to the Statecharts [23], the behavioral language of Statemate. Though not hierarchical, our FSM notation for component behaviors (see Section 4.1), combined with the patterns proposed in this thesis, is similar to the Statechart features such as run-to-completion and execution history.

There are a number of component-based development (CBD) frameworks for embedded systems described in the literature. The BIP framework and the toolkit IF [24] are intended for predictable embedded systems development by supporting correctness-by-construction and compositional verification. While BIP offers bottom-up design of systems, the component-based design support presented in the thesis (PAPER A) facilitates traditional top-down design, with support for formal modeling in Save-IDE [25], a precursor of ProCom framework, and model-checking based verification using the UPPAAL PORT toolkit [20, 26].
Pattern-based design support for RTES development has also been in the attention of fellow researchers, due to its immediate benefits. For instance, Sandén proposed the “state-machine” pattern [27], for designing concurrent real-time software in Ada [28]. Many possible implementations of the pattern, corresponding to concurrent, reactive, and time-triggered behaviors, are described. Also, patterns for non-functional aspects such as resource usage, quality-of-service have been proposed [29]. However, such patterns focus on the design or implementation phase of the system. The patterns proposed in this thesis support the design process, by directly mapping the specification aspects, that is, an abstract behavioral model, with associated timing constraints, into the corresponding design elements.

Maxwell et al. have proposed a formal framework [30] for heuristics-based transformation of architectural designs. The authors capture heuristics in a structured and formal manner, such that the architectural transformations can be performed for optimizing the non-functional qualities of a system. Denford et al. have proposed an architectural refinement method [31] that focuses on non-functional requirements e.g., reliability, performance, while still addressing the functional requirements. While these works focus on non-functional aspects other than timing, e.g. performance, we address architectural designs with timing constraints.

In this thesis, we have integrated high-level system behavior models, that is, mode behavior specifications (also referred as modemachines), in the model-based development process. The mode behavior specifications provide a very useful abstraction of the system to support the initial phase analysis as well as the design process. As related work, mode automata and the notion of running modes have been introduced in the domain of synchronous languages [32], to reduce the gap between the initial design of a system and the program written for it. The formalism has been proposed to support both dataflow as well as imperative styles. The mode behavior models used in this thesis correspond to the event-based, hierarchical, high-level control structure of the system and associated timing constraints.

As standard modeling environments, UML/MARTE and AADL (Architecture analysis and design language [33]) are extensively used in the designs of RTES [34, 35]. AADL supports the modeling of both software components such as thread, subprogram, process, and platform components, e.g., bus, memory, processor, and device. As a related modeling environment, EAST-ADL [12] is a layered architecture language for model-based development of automotive software. To address various concerns of system’s life-cycle development, it provides abstraction layers such as feature level, requirements, analysis, design,
and implementation. The design support that we have provided in this thesis is applicable to the above standard modeling environments and can facilitate enhanced modeling capabilities.

5.2 Formal Verification Support for Component-based RTES

The standard modeling environments such as UML, AADL and EAST-ADL lack a unified formal semantics. For instance, AADL introduces avoidable redundancies that obscure the model and may even lead to design inconsistence. To address this deficiency, the CCSL constraints have been used [35] to precisely specify both event, and time-triggered communications for AADL models, and to compute end-to-end flow latency. These works focus on models related to software and platform mapping. In our work (PAPER C), we offer formally verified support for component-based system design, in the form of patterns based on timing constraints. In the context of EAST-ADL, Mallet et. al. have described CCSL semantics for EastADL timing requirements [36]. This enables the use of MARTE tools for timing verification of EastADL requirements. In this thesis, we have extensively used CCSL for the precise specification of timed causality semantics of the structural and behavioral models of RTES. However, we have proposed transformation mechanisms into the real-time semantic domain of timed automata, enabling model-checking of RTES models annotated with CCSL constraints that offer a basis for specifying combined logical and chronometric time constraints. Thus, the thesis work extends the existing analysis frameworks towards model checking.

COMDES-II (Component-Based Design of Software for Distributed Embedded Systems) [37] is a development framework in which the functional units encapsulate one or more dynamically scheduled activities. Besides providing a clear separation of concerns (functional behavior from real-time behavior), in modeling, COMDES-II also offers support for formal analysis, by specifying the activity behavior in terms of hybrid state machines. In comparison to COMDES-II, we have also adopted heterogeneous modeling approach where functionality of a component is modeled using statemachine-based behavior models. However, our CCSL-based approach is highly expressive in terms of the required timing behavior. Koala [38] is a software component model, introduced by Philips Electronics, designed to build product families of consumer electronics. For Koala compositions, the non-functional information is exposed at the component’s interface. The prediction of non-functional properties is
carried out by measurements and simulations at the application level. The communication among SOFA components [39] can be captured formally, by traces, which are sequences of event tokens denoting the events occurring at the interface of a component. The behavior of a SOFA entity (interface, frame or architecture) is the set of all traces, which can be produced by the entity. In comparison, we have specified timed causality semantics of architectural and component-based designs using CCSL, which is both expressive and precise. ProCom’s precursor, SaveCCM, is also an analyzable component model for real-time systems [40]. SaveCCM’s semantics is defined by a transformation into timed automata with tasks, a formalism that explicitly models timing and real-time task scheduling. In comparison, we have defined semantics of ProCom using an intuitive notation (in PAPER B) with real-time features, which can be transformed into timed automata or other such semantic formalisms.

A process-algebraic approach to describing architectural behavior of component models is advocated by Allen and Garlan [41], and Magee et al. [42], who formalize the component behavior in CSP (Communicating Sequential Processes) and via a labeled transition system with a possibly infinite number of states. In comparison, we have proposed generic component-based patterns to facilitate intuitive designs as well as support the transformation to timed automata for formal verification.

In the context of EAST-ADL, several approaches have been proposed for timed automata based modeling. Qureshi et al. [43] present automata templates for EAST-ADL timing constraints. These modeling templates model various error scenarios and are based on informal semantics of the EAST-ADL architectural models. In comparison, the automata templates presented in this paper specify event chains and associated causality and temporal behavior. Kang et al. [44] present a method for formal modeling of EAST-ADL models, for verification using UPPAAL. Enoiu et al. [45], have provided a tool support for EAST-ADL models, also limited aspect of timing verification. In comparison to these works, in this thesis, we have addressed the analysis of timing specifications based on the notion of logical constraints. We have also presented the complementary use of different analysis approaches, such as simulation and model checking using UPPAAL, for timing specifications.
Chapter 6
Conclusions and Future Work

In this chapter, we present a summary of the thesis contributions, as well as corresponding limitations. Finally, we conclude the thesis work with an outline of the future work.

6.1 Summary and Conclusions

In this thesis, we have mainly addressed two research goals:

• A component-based design support to produce analyzable models of RTES.

• A real-time semantic basis to make formal verification of behavior and architectural models feasible.

As the first contribution of this thesis, we have presented component-based behavioral and design patterns, in order to produce manageable and easy-to-grasp design models for RTES. The recurring behaviors of real-time components, such as run-to-completion, history, timing etc, are specified using an intuitive finite state machine notation. We also describe how the behavior specifications are syntactically transformed into timed automata based modeling framework, for model checking using formal tools such as UPPAAL PORT and UPPAAL. As a modeling result, we have provided a design support for developing intuitive
and analyzable component-based designs for RTES. Further, we have integrated high-level specification models such as mode behavior models (also referred as *mode machines*) within the design process. The behavior models have been combined with CCSL constraints, thus providing expressive models to enhance the design and verification during early-phases of the system development. In this context, as a significant contribution of the thesis, we were able to provide a *model checking* - based verification technique to models of behavior enhanced with CCSL constraints, which are otherwise not directly verifiable by *model checking*.

Another line of contribution is our novel use of both CCSL and timed automata, the real-time formalisms, to enrich the model-based analysis of RTES. As a significant result, we have been able to bridge the CCSL and TA-based frameworks, by successfully mapping the synchronous and discrete chronometric semantics of CCSL into the asynchronous and dense-time semantics of timed automata. To demonstrate the benefits of the proposed transformation, we have verified both logical and chronometric properties for the structural and behavioral models of RTES. Hence, we have successfully achieved the research goal of providing a semantic basis, that can potentially help engineers who develop the models, as well as researchers investigating the corresponding analysis techniques, model-transformation tools, etc.

Finally, we have also validated the research results using relevant examples of RTES as well as an industrial case-study, a simplified *wind turbine system*. The pattern-based design support as well as the CCSL and timed automata based semantics have been found to be useful for the model-based development. Further, we have extended the existing simulation-based techniques towards exhaustive verification of critical system properties. We have also identified some limitations of the approach, such as the need for generating suitable simulation data w.r.t. the properties under verification.

## 6.2 Future Work

As future work, we intend to extend the pattern-based design support towards a structured design process with the help of necessary tool support. For the verification support, similar structured approach can be taken for the CCSL and timed automata semantics presented in the thesis. CCSL is a very expressive language (e.g. unbounded operators, discrete semantics, etc.) pose challenges to *model checking*. Hence a suitable subset of CCSL needs to be investigated to provide verifiable specifications by efficient transformations into timed au-
tomata. In the thesis, we have used the proposed semantic basis only to verify
the timed causality behavior of the system models. However this can be ex-
tended to computational aspects, as timed automata provides data variables and
update actions to represent the computation. This paves the way for extended
verification capabilities for CCSL-based system models. Finally, as identified
during the modeling and verification of the wind turbine case-study, techniques
in the domain of model-based testing may be investigated; for instance, identifi-
cation of suitable input values, w.r.t. the property under verification, to generate
simulation data to assist the verification process described in the thesis.
Bibliography


[33] Society of Automotive Engineers (SAE). Architecture analysis and design language (AADL), June 2006.


