

Mälardalen University Press Dissertations
No. xxx

A Resource-Aware Framework for Designing Predictable Component-Based Embedded Systems

Aneta Vulgarakis

2012



MÄLARDALEN UNIVERSITY

School of Innovation, Design and Engineering
Mälardalen University

Copyright © Aneta Vulgarakis, 2012
ISSN xxxx
ISBN xxx
Printed by Arkitektkopia, Västerås, Sweden

Abstract

Acknowledgements

Contents

1	Introduction	1
1.1	Research Motivation	1
1.2	Problem Statement and Research Goals	3
1.3	Contributions	6
1.4	Publications	9
1.4.1	Description of fundamental publications	9
1.4.2	Publications related to the thesis	16
1.5	Research Methodology	18
1.6	Thesis Outline	21
2	Background	25
2.1	Component-Based Development	25
2.2	Formal Models and Analysis Techniques	26
2.2.1	Timed automata	26
2.2.2	Priced timed automata	26
2.2.3	Model-checking technique	26
3	ProCom: A Component Model for Embedded Systems	27
3.1	Domain Requirements of the ProCom Component Model	28
3.2	Syntax and Informal Semantics	28
3.2.1	ProSys the upper layer	28
3.2.2	ProSave the lower layer	28
3.2.3	Integration of layers — combining ProSave and ProSys	28
3.2.4	Example: An Electronic Stability Control System	28
3.3	Formal Semantics of the ProCom Component Model	28
3.3.1	Underlying Formalism and Graphical Notation	28
3.3.2	Overview of ProCom formalization	28

3.3.3	Formal Semantics of Selected ProCom Architectural Elements	28
3.4	Summary	28
4	REMES: A Behavioral Model for Embedded Systems	29
4.1	Classes of Resources	30
4.2	Introducing REMES	30
4.3	Composition of REMES Models	30
4.4	Formal Analysis of REMES Models	30
4.4.1	Feasibility analysis	30
4.4.2	Optimal and worst-case resource consumption	30
4.4.3	Trade-off analysis	30
4.5	Example: A Temperature Control System	30
4.6	Summary	30
5	Integrating ProCom and REMES	31
5.1	Connecting component interfaces and REMES modes	31
5.2	Packaging ProCom components and REMES modes together	31
5.3	Example: A Turntable Drilling System	31
5.4	Summary	31
6	The REMES Tool-chain	33
6.1	The REMES Editor	33
6.2	An Automated Transformation from REMES into PTA	33
6.3	Summary	33
7	Case Study: Ericsson Nikola Tesla Demonstrator	35
7.1	Overview of the Validation Process	35
7.2	Description of the Demonstrator	37
7.3	The ProCom Architecture of the Demonstrator	38
7.4	REMES Modeling and Formal Analysis of the Demonstrator	40
7.4.1	The REMES model of the ENT demonstrator	41
7.4.2	Formal analysis goals	42
7.4.3	PTA model of the ENT demonstrator and analysis results	42
7.5	Summary	45
8	Related Work	47
8.1	Component Models for Embedded Systems	47
8.2	Resource-Aware Modeling and Analysis	47

9 Conclusions	49
9.1 Summary and Limitations	49
9.2 Future work	49
Bibliography	51
Index	57

List of Figures

1.1	Overview of the applied research process.	19
7.1	The system validation process.	36
7.2	The deployment architecture of the demonstrator.	37
7.3	The ProSys model of the ENT demonstrator.	39
7.4	<i>Pen</i> , <i>Client1</i> and <i>Server1</i> modeled in REMES.	43
7.5	PTA model of the <i>Pen_Input</i> component.	44
7.6	PTA model of the <i>Client1</i> component.	44
7.7	Optimal trace for processing four requests.	46

List of Tables

Chapter 1

Introduction

The following introduction will motivate the need for a new development method for embedded systems (Section 1.1), and then describe in detail the research problem tackled in this thesis and list the research goals (Section 1.2). Afterwards, it will point out the scientific contributions of the thesis (Section 1.3), before it will list the published papers that establish the contributions of the thesis (Section 1.4). Finally, it will present the research methodology used for answering the research problem (Section 1.5), and provide an outline of the thesis (Section 1.6).

1.1 Research Motivation

Embedded systems, such as mobile phones, car engines, elevators, etc., are part of our daily life, and we are increasingly depending on their reliability in operation. According to IEEE Glossary [1] "an embedded system is as a computer system that is part of a larger system and performs some of the requirements of that system". Like all computing systems, embedded systems consist of hardware and software integrations.

During recent decades, the amount of software in embedded systems is increasing at a breathtaking pace. For example, a modern upper-class car holds between a dozen and nearly 100 crosslinked electronic control units (ECU), each with a microprocessor software that amounts to about 1MByte compiled code [8]. This is comparable to what a typical desktop computer runs today. Reasons for this tremendous increase include the

demand for new functionality on the one hand, and the availability of powerful and cheap hardware on the other hand. In difference to a general purpose computer, an embedded system has to satisfy the following extra-functional requirements [13]:

- reaction requirements (such as deadlines and response time), which concern that the system interacts with the physical world in a timely manner, and
- execution requirements (such as limited computation power, memory space, and channel bandwidth), which concern the interaction of the system with the underlying platform.

The demanding extra-functional requirements of modern embedded systems coupled with the increasing complexity of the underlying software, stimulate application of techniques for managing complexity and for ensuring predictable system behavior. The existing theories and methods for software development, when applied to software design of embedded systems, reveal the two major challenges of embedded system design. The first challenge is to provide an artifact (an embedded computer system) that provides the specified services under given constraints. The second challenge is that relevant properties of this artifact need to be modeled at different levels of abstraction by a hierarchy of models of adequate simplicity [18, 30]. Accordingly, there is a need for improved software development techniques and processes that will let developers to tame software's growing complexity, while reducing time to market and development costs. *Component-based development* (CBD) aims to be a promising approach to handle the complexity associated with software development, reduce time to market, introduce structure and abstractions. It enables building components and software systems from pre-existing individual components. The underlying paradigm is that individual components are designed and developed to provide functionality that is potentially reusable for future systems.

The central point of CBD has been reuse, but for embedded systems the structure and abstractions introduced by components are equally important as a basis for construction of abstract formal models. An essential benefit of a formal model is that it forces one to specify the system in a precise and unambiguous way, which may reveal inconsistencies and gaps in the original informal description. Through abstraction formal models allow software engineers to focus on the critical issues

facing them. Through logical foundations they support predictable development already at early design time, where *predictability* concerns the possibility to guarantee absence or presence of certain properties, or to predict/guaranty value of a property. This avoids cost intensive redesigns of systems in late development phases. The predictability analysis should guide the design and selection of hardware and software system components. The final implementation of the system should be made as much as possible using automatic synthesis from formal models describing the system behavior in order to ensure implementations that are "correct by construction" [7].

1.2 Problem Statement and Research Goals

In the previous section, we have shown that the development of embedded systems is a challenging task, due to their growing complexity and the pervasive nature of their most critical property: resource-limitations. Resource-usage should be predicted and assessed already at the early design phases, since access to such information at early stages of design can help the designer to prevent resource conflicts at run-time. This can in turn, help to decrease the embedded system's development time and, consequently, reduce development costs.

Based on the above discussion, we identify our general research problem coming from the embedded systems practice as:

Need to address the complexity and resource limitations of embedded systems in a structural way and ensure predictability during early stages of system development.

In order to refine the general research problem, we narrow our focus from different perspectives. Firstly, we consider that in order to achieve predictability throughout the development of embedded systems, the designer needs to employ a design framework equipped with analysis methods and tools that can be applied at various levels of abstraction. These methods and tools should provide estimations and guarantees of relevant system properties.

Secondly, we have the opinion that the CBD principles introduce structure and abstraction, and enable reusability of various types of analysis, so we concentrate on designing embedded systems according

to the CBD standards.

Thirdly, in our view formal analysis of functionality, timeliness and resource usage are important complements to testing. A representative analysis goal is to verify the resource-wise feasibility property. Such property can state that the composition of the worst-case resource requirements of components stays within the available resources provided by the implementation platform, or that there exists an execution path that uses no more than the available resources to behave correctly.

Taking into account these objectives, we specify our refined problem statement as:

Develop a resource-aware framework encompassing modeling and analysis methods for component-based embedded systems.

Research Goals

Decomposing the refined problem statement, we formulate three research goals to be addressed in this thesis.

Research goal 1.

Component models are indispensable to CBD, as they define rules for constructing individual components and for assembling them into systems. The potential benefits of CBD are as attractive in the domain of embedded systems as they are in other areas of the software industry. Beside component models, component technologies form another central concept of CBD. They make use of component models in practice, that is, a particular component technology provides tools that enable development and deployment of systems that adhere to a corresponding component model. Although there exist several component models and technologies for the development of embedded systems (e.g., AUTOSAR [4], BlueArX [17], COMDES-II [16], Koala [28], Pecos [29], Robocop [22], Rubus [12], and SaveCCM [2]), CBD is still not broadly used in the embedded systems industry. An important reason for such limited success is the difficulty of providing solutions that meet typical embedded systems requirements.

Wolf [30] discusses about which domain specific requirements a component technology targeting embedded system development should be aware of. In the embedded systems domain, designing for predictability

requires architectures that meet both the corresponding functional requirements (e.g., expected services, functionality and features), as well as extra-functional ones (resource-feasibility, timing and/or reliability). In order to simplify analysis and help the intuition behind the embedded system's functioning, one could create a hierarchy of models that will allow them to reason about timed behavior, resource consumption and so on, without going down to the instruction level. For instance, architectural models may be used for modeling basic functionality, and behavioral models for modeling functional and extra-functional behavior. Also, embedded system developers must verify that applications meet their functional and extra-functional specification. All these requirements should be reflected in the component model. However, the specifications of many component models are defined informally and component models suffer from incomplete and imprecisely defined syntax and semantics. Formalization of a component model using formal methods can provide unambiguous system designs. Therefore, it is essential to associate the component model and its constructs with a formal semantics to which any design should conform. Such motivation justifies our first research goal:

Develop a formal description of a component model for real-time embedded systems.

(RG1)

Research goal 2.

The diversity of approaches on resource modeling and analysis existing in the literature [19, 20, 21, 9, 15, 11, 3, 25] indicate the difficulty of handling all relevant embedded resources within the same formal model. This calls for an innovative look on resource-aware design methods, based on the experience gathered from the existing modeling approaches. In order to properly specify and analyze embedded systems, the designer requires a modeling language that incorporates resources as primitive types, that is, built-in the model. Ideally the language should be unified to support modeling and analyzing functional and timing behavior too, besides the resource-wise behavior of the embedded system. This would allow both separation of concerns as well as simple model-to-model transformations, for analysis purposes. Accordingly, the second research goal can be formulated as:

Develop a behavioral language that will support modeling and formal analysis of functional, timing and resource-wise behavior of components and their compositions.

(RG2)

Research goal 3.

The usefulness, applicability, and scalability of embedded systems modeling languages and analysis methods can be exercised by performing their validation against measured, quantified behavioral properties. In order to illustrate, as well as validate the applicability of our design framework, we must perform a number of case-studies. Thus, our third research goal is:

Exercise the applicability of the proposed design framework in modeling and analysis of example embedded systems that are motivated by reality.

(RG3)

1.3 Contributions

In this section, we map the contributions of the thesis to the research problem and goals formulated earlier.

Research goal 1.

Develop a formal description of a component model for real-time embedded systems.

(RG1)

The contributions addressing RG1 are as follows:

- **The formally specified ProCom component model for embedded systems.** ProCom is particularly designed to target control-intensive distributed systems, which are special class of embedded systems that can be found in many products, such as vehicles, automation systems, or distributed wireless networks. In order to address the different concerns at different levels of granularity, ProCom is structured in two distinct, but related, layers

(ProSys and ProSave). The two layers differ in terms of granularity, architectural style and communication paradigm. The formalization of the ProCom component model is based on an extension of finite-state machines (FSM) and sets the ground for formal analysis of systems built out of ProCom elements. The proposed FSM language has notions of urgency, implicit timing and priorities. Its formal semantics is expressed in terms of timed automata with priorities [6] and urgent transitions [5]. The FSM language has graphical appeal, making it simpler than the corresponding timed automata model, and it abstracts from real-valued variables and synchronization channels.

Research goal 2.

Develop a behavioral language that will support modeling and formal analysis of functional, timing and resource-wise behavior of components and their compositions.

(RG2)

The contributions addressing RG2 are as follows:

- **The REMES behavioral language.** REMES is intended as a meaningful basis for modeling and analysis of resource-constrained behavior of embedded systems. REMES is a dense time state-based hierarchical behavioral language that has a notion of explicit entry- and exit points, continuous variables, flows and progress invariants, making it fit for component-based system modeling.
- **An integration of ProCom and REMES.** The integration is done via the ProCom's attribute framework, that enables a developer of a ProCom component to specify the corresponding behavior by pointing to a REMES model. Both the ProCom component and the associated REMES model are seen as a reusable unit of composition. To accomplish this, in this thesis, we propose a way of connecting ProCom and REMES together. The relation between the ports of the component and the variables in the REMES model is given by a mapping between the ProCom and REMES interface.
- **Performing resource-wise analysis.** We present a method for encoding the resource-wise analysis problem as a weighted sum

in which the variables capture the accumulated consumption of resources, respectively. Thus, we perform three types of analysis: feasibility analysis, optimal or worst-case resource consumption analysis, and trade-off analysis. Feasibility analysis checks whether the accumulated values of the resources consumed/used during all possible system behaviors are within the available resource amounts provided by the implementation platform. Optimal or worst-case resource consumption analysis returns the cost of the “cheapest”, and/or most “expensive” trace that will eventually reach some goal. This analysis may help in resolving the possible non-determinism in a component implementation. Trade-off analysis is a systemic approach to balancing trade-offs between conflicting resource requirements: memory vs. execution time, energy vs. memory, etc. The result of this analysis is the best alternative between the conflicting requirements.

- **A tool chain for the REMES language.** The tool-chain can be employed for construction and analysis of embedded systems. It consists of the following tools: (i) a REMES editor for modeling behaviors of embedded components, (ii) a REMES simulator to test timing and resource behavior prior to formal analysis, and (iii) an automated transformation from REMES to priced timed automata, needed for formal analysis. The REMES simulator is out of the scope of this thesis and therefore will only be shortly described.

Research goal 3.

Exercise the applicability of the proposed design framework in modeling and analysis of embedded system examples that are motivated by reality.

(RG3)

RG3 has been addressed with the following contribution:

- **Validating the resource-aware framework.** ProCom and REMES have been applied on simple, yet relevant “toy examples”: an electronic stability control system, a temperature control system and a turntable drilling system. We also show how to model extra-functional behavior, and verify the resulted behavioral models of a

component-based Ericsson Nikola Tesla prototype telecommunication system. The validation of our models is ensured by the actual values of timing, CPU, and memory usage in our models, measured by Ericsson researchers on the prototype's source code.

Hence, all three research goals have been targeted, and as such also the refined problem statement "*develop a resource-aware framework encompassing modeling and analysis methods for component-based embedded systems*" has been addressed. Needless to say, we have provided one solution to the research problem, out of a possibly large pool of valid solutions.

The resource-aware framework that we present in this thesis includes two parts:

1. The formally specified ProCom component model that fulfills the requirements coming from the embedded systems domain;
2. The REMES behavioral language for describing component's and system's functional and extra-functional behavior (such as timed behavior and resource consumption), associated analysis techniques for various resource-wise properties, and a set of tools implementing the former.

1.4 Publications

This section presents planned and published papers related to the thesis. The publications are divided into two categories: (i) papers that are fundamental for the thesis contributions; and (ii) papers that are related to the thesis.

1.4.1 Description of fundamental publications

Licentiate thesis

- *A Resource-Aware Component Model for Embedded Systems*. Aneta Vulgarakis. Licentiate Thesis, ISBN 978-91-86135-37-9, Mälardalen University Press, September 2009.

Summary: In this thesis we introduce the ProCom component model for building embedded systems, as well as the REMES behavioral language for describing the internal behavior of components.

Usage in the thesis: This doctoral thesis is a continuation of the research work presented in the licentiate thesis. In the doctoral thesis we extend the REMES behavioral language, introduce an algorithm for automatic transformation from REMES to priced timed automata, show a tool for modeling and analysis of REMES models, present an integration of ProCom and REMES, and validate the REMES behavioral language.

Journals

- **paper A.** *Resource-Oriented Modeling and Formal Analysis of Embedded Systems Behavior*. Marin Orlić, Aneta Vulgarakis, Cristina Secleanu, and Paul Pettersson. To be submitted to Journal of Systems and Software.

Summary: This paper is based on the work presented in papers E and G. Additionally, the paper presents an extension of the REMES behavioral language, reveals a solution for the problem with access to shared variables of REMES modes, and presents an algorithm for transformation of REMES modes to priced timed automata.

Contribution: I and Marin Orlić are the main authors of this paper. I am responsible for addressing the problem with access to shared variables of REMES modes. Together me and Marin Orlić have formally defined the automated transformation from REMES into priced timed automata.

Usage in the thesis: This paper is a basis for Chapter 4 and Chapter 6. It describes the extended version of the REMES behavioral language, the algorithm for transforming REMES modes into priced timed automata, and the /remes tool-chain.

- **paper B.** *A Classification Framework for Component Models*. Ivica Crnković, Séverine Sentilles, Aneta Vulgarakis, and Michel Chaudron. IEEE Transactions on Software Engineering. October, 2011.

Summary: This paper presents a survey of a number of component models, described and classified with respect to a three-dimensional classification framework, which groups different aspects

of the development process of component models. As such, this classification framework identifies common characteristics as well as differences between selected component models. The results of the comparison have led to some observations which are discussed in this paper.

Contribution: This paper was written with an equal contribution of the first three authors. All the coauthors have contributed with ideas, discussions, and reviews. I was responsible mainly for the lifecycle dimension and shared the responsibility with Séverine Sentilles for collecting, analyzing and classifying in tables the included component models. The classification framework was developed in several iteration steps including observations and analysis. It was discussed with several CBD and empirical software engineering researchers and experts from different engineering domains.

Usage in the thesis: This paper is used in Chapter 8 for describing the state of the art of component models for embedded systems. In addition, the knowledge gained from this paper is used as a basis for designing the ProCom component model, presented in Chapter 3.

Conferences and workshops

- **paper C.** *Validation of Embedded Systems Behavioral Models on a Component-Based Ericsson Nikola Tesla Demonstrator.* Aneta Vulgarakis, Cristina Seceleanu, Paul Pettersson, Ivan Skuliber and Darko Huljenić. 11th International Conference on Quality Software (QSIC 2011), IEEE, Madrid, Spain, July, 2011.

Summary: In this paper, we show how to model extra-functional behavior, and verify the resulted behavioral models of a component-based Ericsson Nikola Tesla prototype telecommunications system. The models are described in our REMES language, with Priced Timed Automata semantics that allows us to apply UPPAAL- based tools for model-checking the system's response time and compute optimal resource usage traces. The validation of our models is ensured by using actual values of timing, CPU, and memory usage in our models, measured by Ericsson researchers on the prototype's source code.

Contribution: I was the main author of this paper. I contributed

to this paper with modeling and analyzing the ENT system. All the coauthors have contributed with valuable discussions and reviews. The requirements and measurements of the ENT system were given by the last two coauthors of this paper.

Usage in the thesis: This paper is a basis for Chapter 7, and describes the validation of the REMES behavioral language on the ENT system.

- **paper D.** *Integrating Behavioral Descriptions into a Component Model for Embedded Systems*. Aneta Vulgarakis, Séverine Sentilles, Jan Carlson, and Cristina Seceleanu. 36th Euromicro Conference on Software Engineering and Advanced Applications (SEAA 2010), IEEE, Lille, France, September, 2010.

Summary: In this paper, we show how the ProCom component model can be combined with the REMES behavioral language. This permits analysis of system properties, while also supporting reuse of behavioral models when components are reused.

Contribution: I was the main driver of this paper. I proposed a way of mapping the ProCom component interface onto the entry and exit variables of REMES modes, such that the two models become connected. Séverine Sentilles was in particular responsible for implementing this connection through the ProCom attribute framework. I was also responsible for exemplifying the connection on a turntable system. All the coauthors have contributed with valuable discussions and reviews.

Usage in the thesis: This paper is a basis for Chapter 5 where the integration of ProCom and REMES is presented.

- **paper E.** *REMES Tool-chain - A Set of Integrated Tools for Behavioral Modeling and Analysis of Embedded Systems*. Dinko Ivanov, Marin Orlić, Cristina Seceleanu and Aneta Vulgarakis. 25th IEEE/ACM International Conference on Automated Software Engineering (ASE 2010), Antwerp, Belgium, September, 2010.

Summary: In this paper, we present our REMES tool-chain that can be employed for construction and analysis of embedded behavioral models. The tool-chain consists of the following tools: (i) a REMES editor for modeling behaviors of embedded components, (ii) a REMES simulator to test timing and resource behavior

prior to formal analysis, and (iii) an automated transformation from REMES to priced timed automata, needed for formal analysis.

Contribution: I and Marin Orlić were the main authors of this paper. I was the REMES tool-chain leader and supervisor, and contributed with suggesting a design of the REMES editor and the REMES meta-model. I and Marin Orlić developed an algorithm for transforming REMES into priced timed automata. Dinko Ivanov developed the REMES editor and Marin Orlić developed the REMES simulator. Cristina Seceleanu coordinated the work on the REMES tool-chain.

Usage in the thesis: This paper is a basis for Chapter 6 where the REMES editor and the transformation from REMES to priced timed automata are presented.

- **paper F.** *Formal Semantics of the ProCom Real-Time Component Model*. Aneta Vulgarakis, Jagadish Suryadevara, Jan Carlson, Cristina Seceleanu, and Paul Pettersson. 35th Euromicro Conference on Software Engineering and Advanced Applications (SEAA 2009), IEEE, Patras, Greece, August, 2009.

Summary: In this paper, we define the formal semantics of the ProCom component model in a small but powerful finite state-machine based formalism, with notions of urgency, timing, and priorities. As such, the formalism provides an unambiguous description of the modeling elements of ProCom, sets the ground for formal analysis using other formalisms, and provides an intuitive and useful description for both practitioners and researchers.

Contribution: I was the main author of this paper. I and Jagadish Suryadevara contributed with defining a formal semantics of the ProCom component model and exemplifying it on the modeling elements of ProCom. All the coauthors have contributed with valuable discussions and reviews. The paper proceeded from a technical report that was written together with Jagadish Suryadevara.

Usage in the thesis: This paper is used in Chapter 3 for describing the formal semantics of the ProCom component model.

- **paper G.** *REMES: A Resource Model for Embedded Systems*. Cristina Seceleanu, Aneta Vulgarakis, and Paul Pettersson. 14th

IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2009), IEEE, Potsdam, Germany, June, 2009.

Summary: This paper introduces the model REMES for formal modeling and analysis of both functional and extra-functional behavior of interacting embedded components. REMES is a state-based behavioral language with support for hierarchical modeling, resource description, continuous time, and notions of explicit entry and exit points that make it suitable as a semantic basis for component-based modeling of embedded systems. The analysis of REMES-based systems is placed around a weighted sum in which the variables capture the accumulated consumption of resources, respectively.

Contribution: This paper was written with equal contribution from all the authors. I particularly worked on the classification of the resources and specified, modeled in REMES, and analyzed in UPPAAL CORA [27] the TCS system presented as a case study in the paper.

Usage in the thesis: This paper is a basis for Chapter 4 where the REMES behavioral language is introduced.

- **paper H.** *A Component Model for Control-Intensive Distributed Embedded Systems.* Séverine Sentilles, Aneta Vulgarakis, Tomáš Bureš, Jan Carlson, and Ivica Crnković. 11th International Symposium on Component Based Software Engineering (CBSE 2008), Karlsruhe, Germany, October 2008.

Summary: In this paper, the two-layered ProCom component model for design and development of control-intensive distributed embedded systems is introduced. ProCom takes into account the most important characteristics of these systems and employs the concept of reusable components throughout the whole development process, from early design to deployment. The two-layered model is developed to efficiently cope with different design paradigms that exist at different abstraction levels of embedded systems (high level view of loosely coupled subsystems and a low-level view of control loops controlling a particular piece of hardware). Additionally it provides ground for analysis and predicting properties (e.g., timed behavior and resource consumptions) in such systems.

Contribution: This paper was written with equal contribution from all the authors, and proceeded from a technical report that was written together with all the authors. I took part in the discussions and contributed with writing and improving parts of the paper, particularly in the discussions about the semantics of the component model, analysis and predicting properties and the related work section. The ProCom component model that we describe in this paper was developed in several iteration steps resulting from the conducted discussions between the authors.

Usage in the thesis: This paper is a basis for Chapter 3 where the ProCom component model is introduced.

- **paper I.** *Embedded Systems Resources: Views on Modeling and Analysis*. Aneta Vulgarakis and Cristina Seceleanu. 1st IEEE International Workshop On Component-Based Design Of Resource-Constrained Systems (CORCS 2008), IEEE, Turku, Finland, July, 2008.

Summary: In this paper, we discuss several representative frameworks that model and estimate resource usage of embedded systems, identifying their advantages and limitations. As such, we divide the variety of approaches existing in the literature into three distinctive categories: code-level resource modeling and analysis of component assemblies, UML-based description of embedded resources and higher-level formal approaches based on temporal logics and process algebras. In the end, we present the resource-aware development view that we are adopting throughout the rest of the thesis.

Contribution: This paper was written with equal contribution from all the authors. I was specifically working on the code-level and UML- based resource modeling and analysis.

Usage in the thesis: This paper is used in Chapter 8 for describing the state of the art of embedded systems resources modeling and analysis. In addition, the knowledge gained from this paper is used as a basis for designing the REMES behavioral language, presented in Chapter 4.

1.4.2 Publications related to the thesis

Journals

- *Applying REMES Behavioral Modeling to PLC Systems.* Aneta Vulgarakis and Aida Čaušević. *Mechatronic Systems*, vol 1, nr 1, p40-49, Faculty Of Electrical Engineering, University Sarajevo, December, 2009.

Conferences and workshops

- *Classification and Survey of Component Models.* Ivica Crnković, Aneta Vulgarakis, Mario Žagar, Ana Petričić, Juraj Feljan, Luka Lednicki, and Josip Maras. DICES workshop @ SoftCOM 2010, Bol, Croatia, September 2010.
- *Towards Simulative Environment for Early Development of Component-Based Embedded Systems.* Marin Orlić, Aneta Vulgarakis, and Mario Žagar. 15th International Workshop on Component-Oriented Programming (WCOP 2010), Prague, Czech Republic, June, 2010.
- *Applying REMES Behavioral Modeling to PLC Systems.* Aneta Vulgarakis and Aida Čaušević. 22nd International Symposium on Information, Communication and Automation Technologies (ICAT 2009), IEEE, Sarajevo, Bosnia Herzegovina, October 2009.
- *Towards a Unified Behavioral Model for Component-Based and Service-Oriented Systems.* Aida Čaušević and Aneta Vulgarakis. 2nd IEEE International Workshop On Component-Based Design Of Resource-Constrained Systems (CORCS 2009), IEEE, Seattle, Washington, July, 2009.
- *Towards a Resource-Aware Component Model for Embedded Systems.* Aneta Vulgarakis. Doctoral Symposium of 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC 2009), IEEE, Seattle, Washington, July, 2009.
- *A Component Model Family for Vehicular Embedded Systems.* Tomáš Bureš, Jan Carlson, Séverine Sentilles, and Aneta Vulgarakis. 3rd International Conference on Software Engineering Advances (ICSEA 2008), IEEE, Sliema, Malta, October 2008.

- *A Classification Framework for Component Models*. Ivica Crnković, Michel Chaudron, Séverine Sentilles, and Aneta Vulgarakis. 7th Conference on Software Engineering and Practice in Sweden (SERPS 2007), Göteborg, Sweden, October 2007.
- *A Model-Based Framework for Designing Embedded Real-Time Systems*. Séverine Sentilles, Aneta Vulgarakis, and Ivica Crnković. Work-In-Progress (WIP) track of the 19th Euromicro Conference on Real-Time Systems (ECRTS), Pisa, Italy, July 2007.

MRTC reports

- *Connecting ProCom and REMES*. Aneta Vulgarakis, Séverine Sentilles, Jan Carlson, and Cristina Seceleanu. MRTC report ISSN 1404-3041 ISRN MDH-MRTC-244/2010-1-SE, Mälardalen Real-Time Research Centre, Mälardalen University, May, 2010.
- *ProCom: Formal Semantics*. Jagadish Suryadevara, Aneta Vulgarakis, Jan Carlson, Cristina Seceleanu, and Paul Pettersson. MRTC report ISSN 1404-3041 ISRN MDH-MRTC-234/2009-1-SE, Mälardalen Real-Time Research Centre, Mälardalen University, March, 2009.
- *REMES: A Resource Model for Embedded Systems* Cristina Seceleanu, Aneta Vulgarakis, and Paul Pettersson. MRTC report ISSN 1404-3041 ISRN MDH-MRTC-232/2008-1-SE, Mälardalen Real-Time Research Centre, Mälardalen University, October, 2008.
- *ProCom – the Progress Component Model Reference Manual, version 1.0*. Tomáš Bureš, Jan Carlson, Ivica Crnković, Séverine Sentilles, and Aneta Vulgarakis. MRTC report ISSN 1404-3041 ISRN MDH-MRTC-230/2008-1-SE, Mälardalen Real-Time Research Centre, Mälardalen University, June 2008.
- *Towards Component Modelling of Embedded Systems in the Vehicular Domain*. Tomáš Bureš, Jan Carlson, Séverine Sentilles, and Aneta Vulgarakis. MRTC report ISSN 1404-3041 ISRN MDH-MRTC-226/2008-1-SE, Mälardalen Real-Time Research Centre, Mälardalen University, April 2008.
- *Progress Component Model Reference Manual - version 0.5*. Tomáš Bureš, Jan Carlson, Ivica Crnković, Séverine Sentilles, and Aneta

Vulgarakis. MRTC report ISSN 1404-3041 ISRN MDH-MRTC-225/2008-1-SE, Mälardalen Real-Time Research Centre, Mälardalen University, April 2008.

1.5 Research Methodology

Depending on the kind of problem to solve and the context of the problem, different research methodology can be used. Research methods and research methodology are two terms that are often interchangeably used. Strictly speaking there is a slight difference between the two. Research methods aim to find solutions to research problems and they describe the steps that one should follow to solve a given problem. Example research methods are: conducting experiments, testing, surveys, interviews, lessons learned, critical analysis of the literature and the like. We refer the reader to [14] for a summary of computing research methods. On the other hand, the Merriam-Webster dictionary defines methodology as a "a body of methods, rules, and postulates employed by a discipline: a particular procedure or set of procedures". In other words, methodology is the general plural term for all the individual research methods one has chosen, but there are certain types of methodology which encompass and use specific methods within them e.g., quantitative/qualitative methodologies.

In our view, a research process describes the stages for conducting a research; it starts with defining a problem, and ends with proposing a solution for that problem. During the research process one may use one or combine several research methods in order to address a certain research goal. The use of one or more research methods to address a certain research goal may create several research results. The research process that is used in this thesis is presented in Figure 1.1 . It consists of four main stages as follows: identification of a general research problem, identification of a refined research problem in a research setting, studying the refined problem and validation. As such, the process begins with identification and formulation of a general research problem from embedded systems practice, and the ultimate goal is to provide a solution to this practical problem. The solution is obtained in a research setting by refining and narrowing down the general problem, studying the refined problem, and finally validation. Solving the research problem

is not a straightforward process but an iterative one, allowing feedbacks between steps. First the research problem is decomposed into research goals, which are clarified, formulated, studied, refined, and even sometimes left aside. When the research results are mature enough, we move to the validation stage that make us examine the validity of our research results. In using this research process, the validation of the results is crucial in both research and industry settings. If the validation stage fails, the research goals and results need to be revisited, improved, polished, and if necessary discarded.

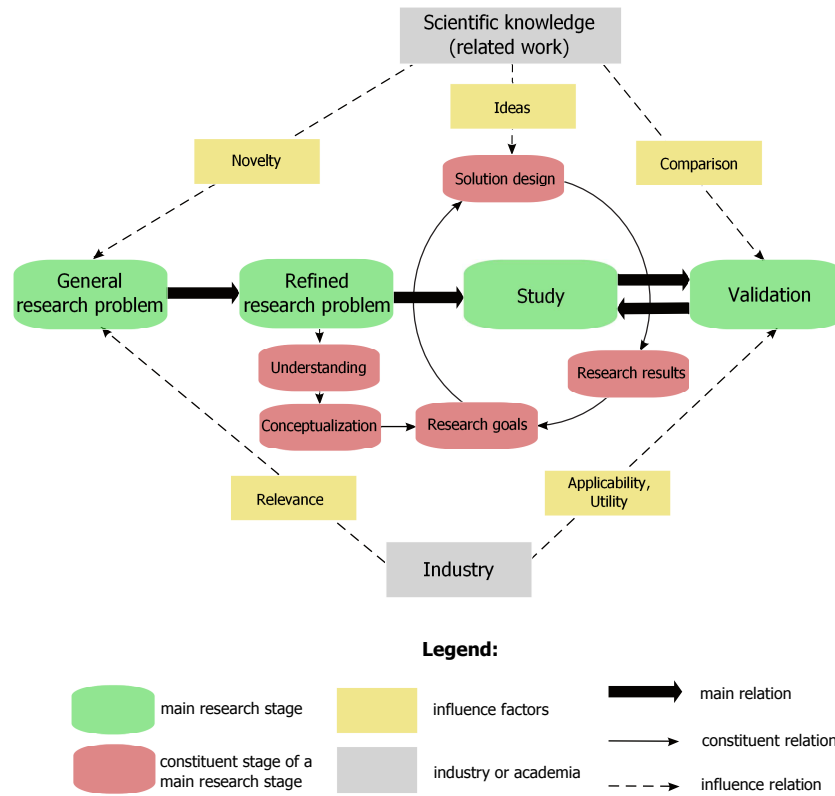


Figure 1.1: Overview of the applied research process.

We considered the general research problem, the need to address the

complexity and resource limitations of embedded systems in a structural way and ensure predictability during early stages of system development, and transferred the problem to a research setting (see Section 1.2). In order to understand the problem both from industrial and scientific perspective, we performed information gathering and studied the state of the art and state of the practise covering previous work done on the research problem. In scientific research, the role of previous work is to give a background for the research problem, and especially explicate the industrial relevance and scientific novelty of the research. During this stage we used the research method that is close to the so called *critical analysis of literature* [31] method. This method is a historical one that aims to provide an exhaustive summary of literature relevant to a research problem, by collecting and analyzing data from published materials. The analysis part provides the opportunity to draw conclusions from a broad range of approaches. We performed our literature review in several iterations, and we discussed the concluded results. In difference to the traditional critical analysis of literature, we did not identify a list of databases for searching related work, and we did not classified the papers covering the related work according to their citation indexes. The investigation of the related work resulted in two papers: paper B and paper I (see Section 1.4). As a result we have studied several (embedded systems') component models and a number of frameworks that model and estimate resource usage of embedded systems.

On this basis we moved to the next stage of our research method - studying the research problem. During this stage we used the *proof of concept* (also known as proof of principle) research method [10]. It involves creating solutions, methodologies, concepts, and techniques in an iterative manner. Note that this research method has a lot in common with software development [23], as in software development the goal is to create a working software system.

Our studying research stage included several iterations where the research results were improved through discussions and analysis. First we conceptualized the problem and expressed it as research goals, presented in Section 1.2. Then, we moved to addressing the research goals by developing solutions, presenting achieved research results and comparing these research results with the research goals. In developing our solutions we drew ideas from the related work. In papers A, D, E, F, G and H we presented our research results on developing a resource-aware framework encompassing modeling and analysis methods for component based em-

bedded systems. We proposed a language for component-based design of embedded systems (ProCom), and a resource-aware behavioral language for describing component's and system's functional and extra-functional behavior (REMES).

The last stage of our research process is validation. Out of the many existing validation techniques [26], in our research we use validation by *persuasion*, *analysis*, and *example* validation. Firstly, we give an explanation and persuade the reader that it is reasonable to use our resource-aware framework in addressing the general research framework. Secondly, by developing examples and performing formal analysis we show how the research results work in practise and whether they can be found satisfactory. According to Shaw [26] the validation described in this thesis covers *toy*-, as well as *slice of life examples*. A toy example presents a simplified example, which might have been motivated by reality, where as a slice of life example is a system that the author has developed. As such, our research results were illustrated on simple yet relevant “toy examples” presented in papers D, G, and H. Accordingly, in paper H we exemplified the ProCom component model on an electronic stability control system of a car. Further in paper A and G, we performed a small case study demonstrating the principles of our resource modeling and analysis approach. The case study was conducted on an abstracted version of the internal design of a temperature control system for heat producing reactor. In paper D, we exemplified our resource-aware framework on a turntable example system, which we modeled as a collection of ProSys components that we connected to their associated behavioral REMES models. Finally, in paper C, we showed how to model extra-functional behavior, and verify the resulted behavioral models on a slice of life component-based Ericsson Nikola Tesla (ENT) telecommunications system. The salient point of our model, which enables its validation, is the fact that we built it by using the timing and resource values extracted from the actual prototype implementation of the ENT system. The REMES behavioral language and the associated analysis techniques were compared with the related work and showed to be applicable for the development and analysis of the ENT system.

1.6 Thesis Outline

The outline of the rest of the dissertation is as follows.

- **Chapter 2 - Background** introduces basics in the areas of component-based development, and formal modeling and analysis of software systems. Section 2.1 discusses concepts of components, component-based systems and component models. Section 2.2 gives an overview of (priced) timed automata and model checking, as they will be used throughout this thesis.
- **Chapter 3 - ProCom: A Component Model for Embedded Systems** proposes a two-layer component model for design and development of control-intensive distributed embedded systems. The upper layer - ProSys - is presented in Section 3.2.1, and the lower layer - ProSave - is described in Section 3.2.2. Section 3.2.3 defines the relation between the two layers. The formal semantics of ProCom (Section 3.3) is defined by using a finite state machine (FSM) underlying formalism with notions of urgency, timing and priority, in which the semantics of each ProCom element is defined as a translation relation from ProCom to the FSM language. The ProCom model and its formal semantics are illustrated through a number of interesting examples.
- **Chapter 4 - REMES: A Behavioral Model for Embedded Systems** introduces the behavioral modeling language REMES for formal modeling and analysis of embedded resources such as storage, energy, communication, and computation (see Section 4.2 and 4.3). Section 4.1 presents a classification of embedded resources, based on their rate of consumption over time, and the attribute of being referable, or not. Section 4.4 shows how a number of important resource analysis problems can be formalized in the framework of (multi-)priced timed automata. Finally, Section 4.5 demonstrates the principles of REMES on a temperature control system for a heat producing reactor.
- **Chapter 5 - Integrating ProCom and REMES** proposes a way of mapping the ProCom component interface onto the entry and exit variables of REMES modes (see Section 5.1), such that the two models become connected. Section 5.3 demonstrates the mapping on a turntable example system, which is modeled as a collection of ProSys components that are connected to their associated behavioral REMES models. The packaging of a ProCom component and its REMES behavioral model together is done through the Pro-

Com's attribute framework (see Section 5.2), which will only be shortly described since it is out of the scope of this thesis.

- **Chapter 6 - The REMES Tool-chain** presents the tool-chain for the REMES language, which can be used for the construction and analysis of embedded system behavioral models. Section 6.1 describes the REMES editor, which is a graphical user interface that allows designing REMES behavioral models. Section 6.2 reveals an algorithm for transformation of REMES modes into priced timed automata.
- **Chapter 7 - Case Study: Ericsson Nikola Tesla Demonstrator** presents a case study where REMES is applied to model and analyze a telecommunication system by Ericsson Nikola Tesla (see Section 7.4). The ProSys layer of ProCom is used to model the architecture of the ENT demonstrator, as shown in Section 7.3.
- **Chapter 8 - Related Work** gives a brief survey and relates the contributions presented in the thesis to relevant research, subdivided into two sections. Section 8.1 covers the state of the art of component models for embedded systems. Section 8.2 glances through several representative frameworks that model and estimate resource usage of embedded systems, pointing out advantages and limitations.
- **Chapter 9 - Conclusions** ends our dissertation with conclusions, enumerates the limitations of our results and lists future research directions.

Chapter 2

Background

This chapter introduces important technical concepts used throughout the remainder of this thesis. It provides an introduction to component-based development (Section 2.1) and to formal models and analysis techniques (Section 2.2). However, for more information on component principles and technologies, we refer to ..., and for details on formal models and analysis techniques to ... or

2.1 Component-Based Development

The key principle of component-based development (CBD) is to build software systems from existing software units, termed components, that are developed separately with reuse and integration in mind.

There are many definitions available for components [2], [3], but all seem to agree on that a component is a unit of composition and component-based development are the process of integrating components. Furthermore, one can state that a component provides its functionality via its interfaces. In other words, a component representing a source will implement a producer interface and a component representing a sink will implement a consumer interface.

2.2 Formal Models and Analysis Techniques

2.2.1 Timed automata

2.2.2 Priced timed automata

2.2.3 Model-checking technique

Model checking is a method of automatically verifying concurrent systems in which a finite state model of a system is compared with a correctness requirement. The process of model checking can be separated into system modeling, requirement specification and verification. It has a number of advantages over other traditional approaches. This method has been used successfully in practice to verify complex circuit design and communication protocols.

Chapter 3

ProCom: A Component Model for Embedded Systems

- 3.1 Domain Requirements of the ProCom Component Model
- 3.2 Syntax and Informal Semantics
 - 3.2.1 ProSys the upper layer
 - 3.2.2 ProSave the lower layer
 - 3.2.3 Integration of layers — combining ProSave and ProSys
 - 3.2.4 Example: An Electronic Stability Control System
- 3.3 Formal Semantics of the ProCom Component Model
 - 3.3.1 Underlying Formalism and Graphical Notation
 - 3.3.2 Overview of ProCom formalization
 - 3.3.3 Formal Semantics of Selected ProCom Architectural Elements
- 3.4 Summary

Chapter 4

REMES: A Behavioral Model for Embedded Systems

4.1 Classes of Resources

4.2 Introducing REMES

4.3 Composition of REMES Models

4.4 Formal Analysis of REMES Models

4.4.1 Feasibility analysis

4.4.2 Optimal and worst-case resource consumption

4.4.3 Trade-off analysis

4.5 Example: A Temperature Control System

4.6 Summary

Chapter 5

Integrating ProCom and REMES

- 5.1 Connecting component interfaces and REMES modes
- 5.2 Packaging ProCom components and REMES modes together
- 5.3 Example: A Turntable Drilling System
- 5.4 Summary

Chapter 6

The REMES Tool-chain

6.1 The REMES Editor

6.2 An Automated Transformation from REMES
into PTA

6.3 Summary

Chapter 7

Case Study: Ericsson Nikola Tesla Demonstrator

7.1 Overview of the Validation Process

The validation process that we use in our case study is iterative, allowing feedbacks between steps. It consists of four steps (see Figure 7.1) as follows.

- **I step.** Based on the system functional requirements the designer builds the ProCom architectural model of the system. Similarly, the verification experts uses both the functional- and resource requirements (such as timing, memory, etc.) to develop the REMES behavioral model of the system.
- **II step.** During this step an interface mapping between the ProCom architectural- and the REMES behavioral model is performed, as described in Chapter 5.
- **III step.** The ProCom architectural- and the REMES behavioral model are together transformed to priced timed automata (PTA)

model for formal analysis. The architectural model gives information about the order of execution of the REMES modes modeling the behavior of the components.

- IV step.** We assume that we have hardware abstraction that provides us with global available resources (e.g., memory budget, cpu load, bandwidth of the communication network, etc.). To perform model-checking, a PTA model of the system is fed into UPPAAL, together with a hardware abstraction and a desired property (requirement) expressed in a temporal logic. UPPAAL then automatically traverses the system's state space in an exhaustive manner. If an invariant property is satisfied, the tool notifies that the verification finished successfully, or if the invariant property is violated, it reports one of the traces that violates the property as a counter-example to the model. For reachability properties the opposite is true i.e., a trace is reported when the property is satisfied.

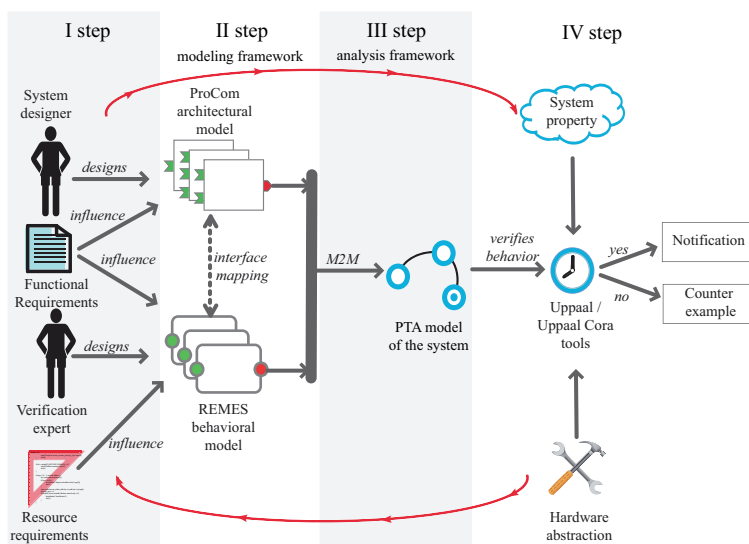


Figure 7.1: The system validation process.

7.2 Description of the Demonstrator

Ericsson Nikola Tesla's (ENT) demonstrator is a prototype of a telecommunications system. It is designed according to current telecommunications industry's trends of adapting horizontal development (systems built from reusable components) methodologies instead of traditionally used vertical ones (systems built from ground-up in-house, now called legacy systems). The organization of the demonstrator is shown in Figure 7.2 from the perspective of deployment architecture.

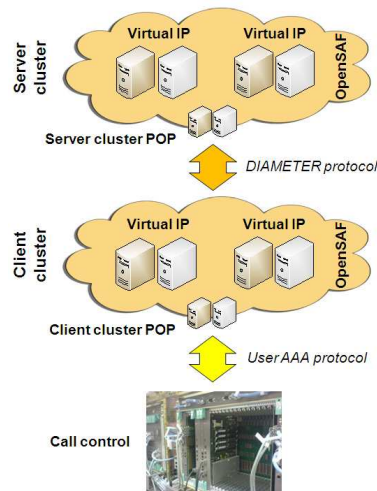


Figure 7.2: The deployment architecture of the demonstrator.

In the demonstrator, a new telecommunications service is created with horizontal development. This new service is added to existing, so called *basic service* that was created over the years with vertical development. More precisely, the basic service performs typical call control functionality: decoding of addressing information and routing calls from one end-point to another. When a special kind of processing is needed, it generates events that result with requests (messages) that are being redirected into the *extension service*. The extension service processes messages generated by the basic service by performing an AAA (authentication, authorization and accounting) functionality that conforms

to the widely accepted Internet standard called DIAMETER [24]. The result of the processing are also messages that are sent back to the basic service.

The extension service is realized as *clients-* and *servers cluster*, which communicate via DIAMETER protocol. They should assure high levels of performance through round-robin load balancing, and availability through redundancy. Implementation of high availability and reliability is facilitated with the use of *OpenSAF*. Previous experiments performed by Ericsson researchers show negligible impact of OpenSAF to the overall performance of the demonstrator [32]. Thus, we omit OpenSAF from experiments shown in this paper.

Pen is a third-party open-source load balancer that was customized for the purpose of load balancing stateful AAA protocol between the basic service and the extension service. *Pen* maintains the information (e.g., IP addresses and ports) about which call control node is communicating with which DIAMETER *client* and uses round-robin method for choosing which client will serve a given request. DIAMETER client receives AAA requests through the AAA protocol between the basic service and the extension service, transforms them into DIAMETER-based AAA requests and sends these DIAMETER-based AAA requests to the DIAMETER servers cluster.

DIAMETER *relay* is a DIAMETER protocol functionality that is used for balancing the load among DIAMETER servers. Similar to *Pen*, it uses round-robin method for choosing which server will serve a given request. Since each DIAMETER message contains full address information about communicating peers, it just transmits the response received from DIAMETER server to corresponding DIAMETER client that originated the initial request. DIAMETER *server* receives DIAMETER-based AAA requests originated on DIAMETER clients. It processes these requests and returns the results to the relay. Since the original request contained the information about which client created it, the relay knows to which client the response must be sent to.

7.3 The ProCom Architecture of the Demonstrator

In this section, we describe a software architecture of the ENT demonstrator that adheres to the ProCom component model. The internal

design of the ENT demonstrator is modeled by two larger subsystems: *Basic Service* and *Extension Service*, as depicted in Figure 7.3. The interfaces of the subsystems are expressed in terms of message ports.

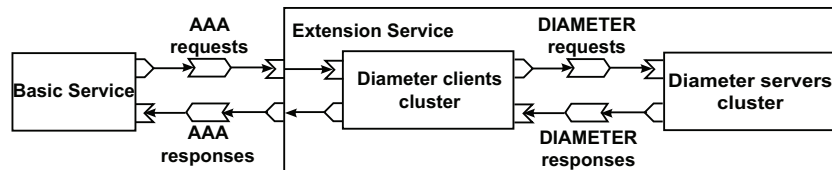


Figure 7.3: The ProSys model of the ENT demonstrator.

Basic Service is an existing legacy ProSys component. *Extension Service* is a subsystem composed of two smaller ProSys components: *Diameter clients cluster* and *Diameter servers cluster*. We consider that there are four clients (resp. servers) in *Diameter clients cluster* (resp. *Diameter servers cluster*). Each of these ProSys components may be further decomposed into either smaller ProSys components, or into ProSave components, depending on the level of complexity of the functionality, and the possibility for distribution. Accordingly, the *Diameter clients cluster* component is built from *Pen* ProSave component and four *Client* ProSave components. Similarly, the *Diameter servers cluster* component is made of *Relay* ProSave component and four *Server* ProSave components.

The component *Basic Service* sends *AAA requests* to *Extension Service*. These requests are forwarded to *Diameter clients cluster* component. Inside this cluster, the *Pen* component is responsible with forwarding these messages in a round-robin fashion to each of the four clients. The *Diameter clients cluster* is the client side of the DIAMETER protocol. Thus, inside *Diameter clients cluster* component *AAA requests* are transformed into *DIAMETER requests* and are forwarded to the *Diameter servers cluster* component. *Relay*, similarly to *Pen*, forwards the *DIAMETER requests* messages in a round-robin manner to each of the four servers. The servers process these requests and return *DIAMETER responses* to *Relay* that forwards them to *Diameter clients cluster*. In the end, *Diameter clients cluster* component transforms *DIAMETER responses* into *AAA responses* and sends them back to *Basic Service*.

7.4 REMES Modeling and Formal Analysis of the Demonstrator

As previously mentioned, the demonstrator used as the case-study in this paper addresses the telecommunications industry's trend of adopting the component-based system design paradigm, that is, construct the system out of reusable components, as a combination of in-house but also third party components. The most important requirement imposed on the demonstrator is to ensure an acceptable performance of the extension service, that is, handling 100 calls per second. For this to happen, and assuming a linear timing behavior per bursts of requests, the end-to-end response time of, say, 500 AAA requests should be less or at most 5 seconds.

To verify this, and, at the same time, validate the abstract descriptions, we have considered in our behavioral models (REMES and the corresponding PTA) the actual source code measured values of the authorization request, and authorization answer response times, respectively, as well as their respective CPU load, and memory usage, for each component of the demonstrator: Pen, DIAMETER client, DIAMETER relay, and DIAMETER server.

Before embarking upon formal validation, we need to check the absence of deadlocks, property specified in UPPAAL as follows:

$$A \square \text{not deadlock}$$

By using the measured values of the demonstrator's extra-functional attributes, we aim at:

- model-checking the REMES system model's capacity (number of handled requests per second), as well as
- computing an optimal execution trace for the overall consumption of resources (CPU and memory).

Verifying the demonstrator's capacity is a crucial performance requirement of the system, and we will next show that we actually deliver a performance guarantee, since model-checking is an exhaustive verification technique. To accomplish the response time verification, we define a global clock variable that stores the elapsed time from the start time of sending the authorization request to the Pen, until the request is served

and returns to the call controller in the basic service. Computing optimal resource-aware traces relies on a weighted sum representation of the resource function, which accounts for both types of resources simultaneously, allowing the designer to set the level of criticality for both resources (identical weights meaning equal importance).

7.4.1 The REMES model of the ENT demonstrator

We model the functional, timing and resource usage behavior of the ENT components as models in REMES. Due to space limitation, here we only present the REMES models of the *Pen* component, one of the clients from *Diameter clients cluster* and one of the servers from *Diameter servers cluster*, depicted in Figure 7.4(a), 7.4(b), and 7.4(c), respectively. *Relay* has similar behavior as the *Pen* component.

In the ENT demonstrator, we make use of two resources: memory and CPU. We assume CPU as a continuous resource, and we treat memory as a discrete resource. Note that in the current version of the demonstrator the clients and the servers are homogenous. From the measurements performed on the source code, we have concluded that *Relay* is the slowest component in the ENT demonstrator and the one that consumes the most resources. Moreover, since the servers are homogenous, we have noticed that *Relay*'s round-robin load balancing protocol always sends messages coming from the first-, second-, third- and fourth- client to the first-, second-, third- and fourth- server, respectively.

The *Pen* mode is made up of two submodes: *Pen_Input* and *Pen_Output*. *Pen* starts executing by entering *Pen_Input* mode and its submode *Receive_IP*, where it reads instantaneously the addresses of the clients. *Pen* may be reentered in case the *Basic Service* component sends a new request (depicted with the Boolean variable *req*) or in case one of the clients is ready to send a response (i.e., at least one of the Boolean variables *client1prio*, *client2prio*, *client3prio* or *client4prio* is evaluated to true). *Basic Service* may send requests to *Pen* only when *Pen* is free (captured with the Boolean variable *penfree*).

The *Pen_Input* mode is responsible for sending requests to the clients in a round-robin fashion. We use the variable *counter* to ensure the round-robin principle. For example, *Pen* is ready to send a message to *Client1* when the guard *client1prio* and $(counter \bmod 4 == 0)$ is evaluated to true. The *Pen_Output* mode receives responses from the clients and forwards them to *Basic Service*.

Client1 receives requests from *Pen*, processes them, and forwards the processed requests to *Relay*. Later, *Relay* sends responses to the requests back to *Client1*. *Client1* sends on the responses to *Pen*. Note that *Client1* can process only one request at a time. The fact that *Client1* has to send back response to *Pen* before receiving a new request is depicted with the Boolean variable *client1prio*.

The *Client1* component remains in the non-lazy mode *Waiting_for_Pen* until receiving a *send_c1* message from *Pen*. When this happens, the component goes via sequence of submodes: *Client1_to_Relay*, *Waiting_for_Relay* and *Client1_to_Pen*. *Client1* stays in modes *Client1_to_Relay* and *Client1_to_Pen* as long as their invariants hold (i.e., until $t \leq 25$ and $t \leq 103$, respectively). The submode *Waiting_for_Relay* is exited when a *relay_to_c1* message arrives.

The *Server1* component receives requests from *Relay*, processes them, and sends them back to *Relay*. *Server1* is entered when *send_c1* becomes true. The *Server1* mode is exited after 68 time units.

7.4.2 Formal analysis goals

7.4.3 PTA model of the ENT demonstrator and analysis results

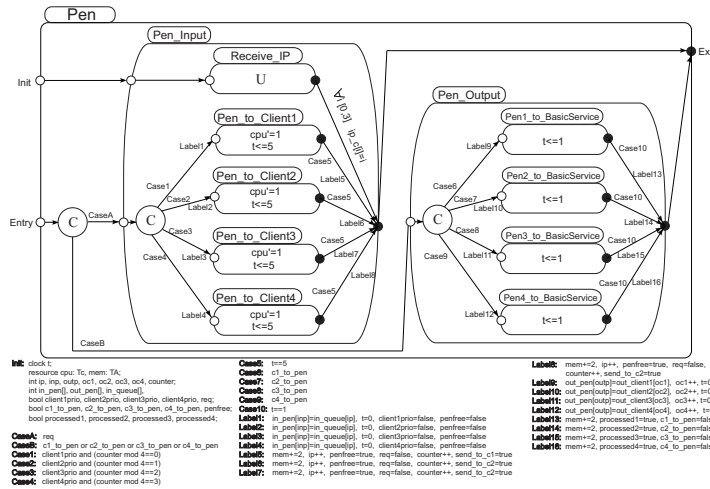
We have analyzed the REMES-based ENT demonstrator, by semantically translating it into a network of PTA models, in CORA¹. Here, we present only the PTA models of the *Pen_Input* submode and the *Client1* mode, shown in Figure 7.5 and 7.6, respectively.

The PTA of *Pen_Input* has six locations: *Start*, *Receive_IP*, *Waiting_for_BasicService*, *Pen_to_Client1*, *Pen_to_Client2*, *Pen_to_Client3* and *Pen_to_Client4*. The synchronization between *Basic_Service* and *Pen_Input* is modeled with channel *req*. Similarly, the synchronization between *Pen_Input* and *Client1* is modeled with channel *send_c1*. The selection of the clients is controlled by the variables *client1prio*, ..., *client4prio* and *counter*.

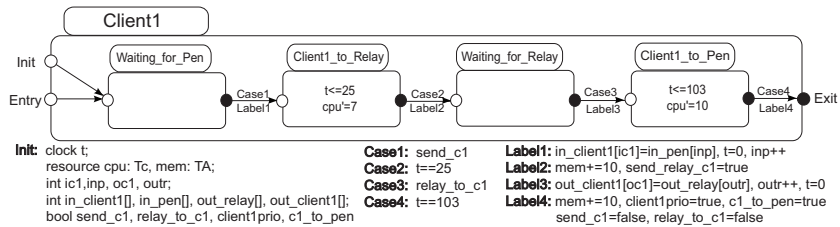
The PTA of *Client1* consists of five locations: *Start*, *Waiting_for_Pen*, *Client1_to_Relay*, *Waiting_for_Relay* and *Pen_to_Client1*. The synchronization between *Client1* and *Relay* is modeled by using two channels *send_relay_c1* (models requests sent from *Client1* to *Relay*), and

¹See the web page www.uppaal.org for more information about the CORA tool. CORA is a branch of the UPPAAL tool for cost optimal reachability analysis.

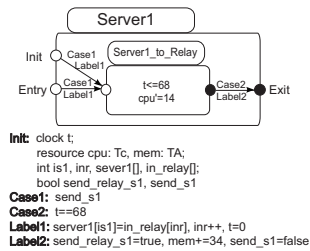
7.4 REMES Modeling and Formal Analysis of the Demonstrator 43



(a) The *Pen* component modeled in REMES.



(b) The *Client1* component modeled in REMES.



(c) The *Server1* component modeled in REMES.

Figure 7.4: *Pen*, *Client1* and *Server1* modeled in REMES.

44 Chapter 7. Case Study: Ericsson Nikola Tesla Demonstrator

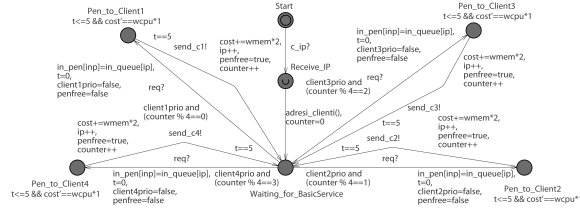


Figure 7.5: PTA model of the Pen_Input component.

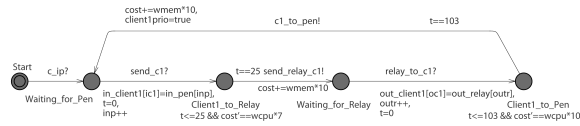


Figure 7.6: PTA model of the Client1 component.

c1_to_relay (models responses sent from *Relay* to *Client1*). The synchronization between *Client1* and *Pen_Output* is modeled by using channel *c1_to_pen*.

In our analysis model, we consider CPU to be a more critical resource than memory. The cost model that we use is derived from the measurements carried out on the actual source code. The resource-usage cost is influenced by the weights of CPU and memory, and the consumed resources of all transitions and locations. In the ENT demonstrator, we consider the following total cost function

$$c_{tot} = w_{cpu} \times c_{cpu} + w_{mem} \times c_{mem}$$

where $w_{cpu} = 2$ and $w_{mem} = 1$, and c_{cpu} and c_{mem} are the accumulated consumed amounts of CPU and memory, respectively.

After providing CORA with the PTA model of the ENT demonstrator, we were able to study the minimum cost reachability problem i.e., to find an execution trace of the system that results in the minimum possible total resource cost. For illustration, let's check for an optimal trace satisfying the reachability property: $E \langle \rangle (processed[3] == 4)$, that is, a trace in which four requests are eventually processed by the ENT demonstrator. The execution trace that was found by CORA is presented in Figure 7.7 and the cost of this best trace is 173308. We use the value

781 ms (from code measurements) as time needed for the basic service to process 500 requests.

From our analysis model we were also able to determine the time needed for processing a certain number of requests. For response time only, we have performed the analysis in UPPAAL (in order to get a TA model we have removed the costs from the PTA model). UPPAAL has calculated that the time needed for handling 1 request is 3,42 time units (ms). If we consider that the processing time grows linearly, then for processing 500 requests our UPPAAL model of the ENT demonstrator needs 1710 time units. This number is just slightly higher than the source code measured value, that is, 1690 ms. The verification result shows that the capacity of the demonstrator (extension service alone) is actually greater than the required 100 requests per second. Also, this result concludes our behavioral model formal validation, regarding the end-to-end response time, which has been a central design issue of the demonstrator.

7.5 Summary

Chapter 8

Related Work

- 8.1 Component Models for Embedded Systems
- 8.2 Resource-Aware Modeling and Analysis

Chapter 9

Conclusions

This chapter summarizes the contributions of the thesis, and describes how they relate to existing work in the area. It also presents a number of directions in which this work could be extended in the future.

9.1 Summary and Limitations

9.2 Future work

Bibliography

- [1] IEEE Standard Glossary of Software Engineering Terminology. *IEEE Std 610.12-1990*, page 1, 1990.
- [2] Mikael Åkerholm, Jan Carlson, Johan Fredriksson, Hans Hansson, John Håkansson, Anders Möller, Paul Pettersson, and Massimo Tivoli. The SAVE approach to component-based development of vehicular systems. *Journal of Systems and Software*, 80(5):655–667, May 2007.
- [3] Hany H. Ammar, Vittorio Cortellessa, and Alaa Ibrahim. Modeling Resources in a UML-Based Simulative Environment. *Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications (AICCSA 2001)*, pages 405–410, 2001.
- [4] AUTOSAR Development Partnership. AUTOSAR – Technical Overview V2.2.1, 2008. http://www.autosar.org/download/AUTOSAR_TechnicalOverview.pdf.
- [5] Johan Bengtsson, W. O. David Griffioen, Kåre J. Kristoffersen, Kim Guldstrand Larsen, Fredrik Larsson, Paul Pettersson, and Wang Yi. Verification of an Audio Protocol with Bus Collision Using UPPAAL. *Proceedings of the 8th International Conference on Computer Aided Verification (CAV 1996)*, pages 244–256, 1996.
- [6] Alexandre David, John Håkansson, Kim Guldstrand Larsen, and Paul Pettersson. Model Checking Timed Automata with Priorities using DBM Subtraction. *Proceedings of the 4th International Conference on Formal Modelling and Analysis of Timed Systems (FORMATS 2006)*, pages 128–142, September 2006.

- [7] Stephen Edwards, Luciano Lavagno, Edward A. Lee, and Alberto Sangiovanni-Vincentelli. Design of Embedded Systems: Formal Models, Validation, and Synthesis. *Proceedings of the IEEE*, 85(3):366–390, mar 1997.
- [8] Ulrik Eklund and Carl Magnus Olsson. A Case Study of the Architecture Business Cycle for an In-Vehicle Software Architecture. In *Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture (WICSA/ECSA)*, pages 91–100, 2009.
- [9] Alexandre V. Fioukov, Evgeni M. Eskenazi, Dieter K. Hammer, and Michel R. V. Chaudron. Evaluation of Static Properties for Component-Based Architectures. *Proceedings of 28th EUROMICRO conference, Component-based Software Engineering track*, pages 33–39, 2002.
- [10] Dawn G. Gregg, Uday R. Kulkarni, and Ajay S. Vinze. Understanding the Philosophical Underpinnings of Software Engineering Research in Information Systems. *Information Systems Frontiers*, 3(2):169–183, 2001.
- [11] Object Management Group. UML Profile for Schedulability, Performance and Time Specification. Version 1.1, formal/05-01-02. 2005.
- [12] Kaj Hänninen, Jukka Mäki-Turja, Mikael Nolin, Mats Lindberg, John Lundbäck, and Kurt-Lennart Lundbäck. The Rubus Component Model for Resource Constrained Real-Time Systems. *Proceedings of the 3rd IEEE International Symposium on Industrial Embedded Systems*, June 2008.
- [13] Thomas A Henzinger. Two Challenges in Embedded Systems Design: Predictability and Robustness. *Philosophical Transactions of the Royal Society - Series A: Mathematical, Physical and Engineering Sciences*, 366(1881):3727–3736, 2008.
- [14] Hilary J. Holz, Anne Applin, Bruria Haberman, Donald Joyce, Helen Purchase, and Catherine Reed. Research Methods in Computing: What are they, and how should we teach them? *SIGSE/SIGCUE Joint Conference on Integrating Technology into Computer Science Education*, 38:96–114, 2006.

- [15] Merijn De Jonge, Johan Muskens, and Michel Chaudron. Scenario-Based Prediction of Run-time Resource Consumption in Component-Based Software Systems. *Proceedings of the 6th ICSE Workshop on Component-based Software Engineering (CBSE6)*, pages 19–24, 2003.
- [16] Xu Ke, Krzysztof Sierszecki, and Christo Angelov. COMDES-II: A Component-Based Framework for Generative Development of Distributed Real-Time Control Systems. *Proceedings of the 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 199–208, 2007.
- [17] Ji Eun Kim, Rahul Kapoor, Martin Herrmann, Jochen Haerdlein, Franz Grzeschniok, and Peter Lutz. Software Behavior Description of Real-Time Embedded Systems in Component Based Software Development. *Proceedings of the 11th IEEE Symposium on Object Oriented Real-Time Distributed Computing (ISORC 2008)*, pages 307–311, 2008.
- [18] Hermann Kopetz. The Complexity Challenge in Embedded Systems Design. *Proceedings of the 11th IEEE International Symposium on Object/Component/Service-Oriented Real-time Distributed Computing (ISORC 2008)*, May 2008.
- [19] Insup Lee, Jin-Young Choi, Hee-Hwan Kwak, Anna Philippou, and Oleg Sokolsky. A Family of Resource-Bound Real-Time Process Algebras. *Proceedings of the 21st International Conference on Formal Techniques for Networked and Distributed Systems (FORTE 2001)*, pages 443–458, 2001.
- [20] Insup Lee, Anna Philippou, and Oleg Sokolsky. A General Resource Framework for Real-Time Systems. *Proceedings of the 9th International Workshop on Radical Innovations of Software and Systems Engineering in the Future (RISSEF 2002)*, pages 234–248, 2002.
- [21] Insup Lee, Anna Philippou, and Oleg Sokolsky. Resources in Process Algebra. *Journal of Logic and Algebraic Programming*, 72(1):98–122, 2007.
- [22] Hugh Maaskant. *A Robust Component Model for Consumer Electronic Products*, volume 3 of *Philips Research*, pages 167–192. Springer, 2005.

- [23] Esperanza Marcos. Software Engineering Research versus Software Development. *ACM SIGSOFT Software Engineering Notes*, 30(4):1–7, 2005.
- [24] Dinko Matijašević, Igor Gizdić, and Darko Huljenić. Mechanisms for Diameter service performance enhancement. In *Proceedings of the 17th International Conference on Software, Telecommunications and Computer Networks - SoftCOM*, 2009.
- [25] Martin Ouimet, Kristina Lundqvist, and Mikael Nolin. The Timed Abstract State Machine Language: An Executable Specification Language for Reactive Real-Time Systems. *Proceedings of the 15th International Conference on Real-Time and Network Systems (RTNS 2007)*, 2007.
- [26] Mary Shaw. What Makes Good Research in Software Engineering? *Software Tools for Technology Transfer*, 4(1):1–7, 2002.
- [27] UPPAAL CORA. <http://www.cs.aau.dk/~behrmann/cora/>, accessed April 2011.
- [28] Rob van Ommering, Frank van der Linden, Jeff Kramer, and Jeff Magee. The Koala Component Model for Consumer Electronics Software. *Computer*, 33(3):78–85, 2000.
- [29] Michael Winter, Christian Zeidler, and Christian Stich. The PECOS software process. *Workshop on Components-based Software Development Processes, ICSR*, 2002.
- [30] Wayne Wolf. Embedded Computing - What Is Embedded Computing? *IEEE Computer*, 35(1):136–137, 2002.
- [31] Marvin V. Zelkowitz and Dolores Wallace. Experimental Validation in Software Engineering. *Information and Software Technology*, 39:735–743, 1997.
- [32] Marijan Zemljić, Ivan Skuliber, and Saša Dešić. Utilization of Open-Source High Availability Middleware in Next Generation Telecom Services. In *Proceedings of the 17th International Conference on Software, Telecommunications and Computer Networks - SoftCOM*, 2009.

Index

- component models, 4, 47
- component-based development, 2, 25
- contribution, 6
- embedded systems, 1
- formal analysis, 4, 26, 30
 - feasibility, 30
 - optimal, 30
 - tradeoff, 30
- model checking, 26
- predictability, 3
- ProCom, 28
 - attribute framework, 31
 - formal semantics, 28
 - integration with REMES, 31
 - ProSave, 28
 - ProSys, 28
- publications, 9
- REMES, 30
 - editor, 33
 - formal analysis, 30
 - integration with ProCom, 31
 - tool-chain, 33
 - transformation to PTA, 33
 - validation, 35
- research methodology, 18
- research methods, 18
 - critical analysis of literature, 20
 - proof of concept, 20
 - validation, 21
 - analysis, 21
 - example, 21
 - persuasion, 21
- research problem, 3
- research process, 18
- resource modeling, 5, 47
- timed automata, 26
 - priced, 26

Index

- component models, 4, 47
- component-based development,
 - 2, 25
- contribution, 6
- embedded systems, 1
- formal analysis, 4, 26, 30
 - feasibility, 30
 - optimal, 30
 - tradeoff, 30
- model checking, 26
- predictability, 3
- ProCom, 28
 - attribute framework, 31
 - formal semantics, 28
 - integration with REMES, 31
 - ProSave, 28
 - ProSys, 28
- publications, 9
- REMES, 30
 - editor, 33
 - formal analysis, 30
 - integration with ProCom, 31
 - tool-chain, 33
 - transformation to PTA, 33
 - validation, 35
- research methodology, 18
- research methods, 18
 - critical analysis of literature,
 - 20
 - proof of concept, 20
 - validation, 21
 - analysis, 21
 - example, 21
 - persuasion, 21
- research problem, 3
- research process, 18
- resource modeling, 5, 47
- timed automata, 26
 - priced, 26

