Mälardalen University Press Licentiate Theses
No. 150

SATISFYING NON-FUNCTIONAL REQUIREMENTS IN MODEL-DRIVEN DEVELOPMENT OF REAL-TIME EMBEDDED SYSTEMS

Mehrdad Saadatmand

2012

School of Innovation, Design and Engineering

ISBN 978-91-7485-066-6
ISSN 1651-9256
Printed by Mälardalen University, Västerås, Sweden

Abstract

Design of real-time embedded systems is a complex and challenging task. Part of this complexity originates from their limited resources which incurs handling a big range of Non-Functional Requirements (NFRs). Therefore, satisfaction of NFRs plays an important role in the correctness of the design of these systems. Model-driven development has the potential to reduce the design complexity of real-time embedded systems by increasing the abstraction level, enabling analysis at earlier phases of development and code generation. In this thesis, we identify some of the challenges that exist in model-driven development of real-time embedded systems with respect to NFRs, and provide techniques and solutions that aim to help with the satisfaction of NFRs. Our end goal is to ensure that the set of NFRs defined for a system is not violated at runtime.

First, we identify and highlight the challenges of modeling NFRs in telecommunication systems and discuss the application of a UML-based approach for modeling them. Since NFRs have dependencies, and the design decisions to satisfy them cannot be considered in isolation, we propose a model-based approach for trade-off analysis of NFRs. The approach enables the comparison of different design models with respect to the satisfaction level of their NFRs. Following the issue of evaluating the interdependencies of NFRs, we also propose solutions for establishing and maintaining balance between different NFRs. In this regard, we categorize our suggested solutions into static and dynamic methods. The former refers to a static design and set of features which ensures and guarantees (by construction) the balance of NFRs, while the latter means establishing balance at runtime by reconfiguring the system and runtime adaptation. Finally, we discuss the role of the execution platform in preservation and monitoring of timing properties in real-time embedded systems and propose an approach to enrich the platform with necessary mechanisms for monitoring them.

To my parents...

Acknowledgments

I would like to start by thanking and appreciating the work of all the people at IDT department for making it a very friendly and cooperative research and educational environment which I believe can serve as a role model for other institutes.

Many special thanks to my main supervisor, Mikael Sjödin, who has been very supportive and encouraging beyond just academical work and also to my assistant supervisors Antonio Cicchetti and Radu Dobrin for all their guidance. Working with you all is a great pleasure.

Thanks to Per Wolde who has been a great friend of mine since the very first days that I came to Sweden and his support for me to start this work. To my office mates, Federico and again Antonio, with whom I have had great memories especially from all the travels that we had together; thank you.

I would like to also thank my managers and colleagues at Enea & Xdin: Barbro Claesson, Erik Netz, Anders Törnqvist, Thomas Barnå, Celi, Mathias L., Daniel B., Joel H., Detlef; also all the great people at IDT whose acquaintance I treasure: Barbara, Farhang, Moris, Saad, Rafia, Åsa, Gunnar, Malin, Monika, Carola, Ingrid, Susanne, Andreas J., Thomas N., Ivica, Dag, Gordana, Jan C., Jan G., Hans, Frank L., Paul, Cristina S., Kristina L., Mats, Björn, Lars, Adnan, Aida, Aneta, Séverine, Nima, Jagadish, Yue Lue, Thomas L., Etienne, Mikael Å., Juraj, Josip, Ana, Luka, Leo, Hüseyin , Hongyu, Kivanc, Andreas G., Daniel, Sasi, Sara, Stefan, Abhilash...

A very special thanks to Nazanin for being by my side with her positive spirit; my life in Sweden would have not been the same without you.

My deepest gratitudes to my family who have always been there for me no matter what. Without them I could have never reached this far.

Mehrdad Saadatmand
Västerås, May, 2012

List of Publications

Papers Included in the Licentiate Thesis [1] [2]

Paper A *UML-Based Modeling of Non-Functional Requirements in Telecommunication Systems*. Mehrdad Saadatmand, Antonio Cicchetti and Mikael Sjödin. The Sixth International Conference on Software Engineering Advances (ICSEA 2011), Barcelona, Spain, October, 2011.

Paper B *Modeling and Trade-off Analysis of NFRs*. Mehrdad Saadatmand, Antonio Cicchetti and Mikael Sjödin. MRTC report ISSN 1404-3041 ISRN MDH-MRTC-267/2012-1-SE, Mälardalen Real-Time Research Centre, Mälardalen University, April, 2012. [3]

Paper C *Modeling Security Aspects in Distributed Real-Time Component-Based Embedded Systems*. Mehrdad Saadatmand and Thomas Leveque. The Ninth International Conference on Information Technology : New Generations (ITNG), Las Vegas, Nevada, USA, April, 2012.

Paper D *Design of Adaptive Security Mechanisms for Real-Time Embedded Systems*. Mehrdad Saadatmand, Antonio Cicchetti and Mikael Sjödin. The Fourth International Symposium on Engineering Secure Software and Systems (ESSoS), Eindhoven, The Netherlands, February, 2012.

Paper E *The Role of Schedulers in Model-Driven Development of Real-Time Systems*. Mehrdad Saadatmand, Mikael Sjödin and Naveed Ul

[1] A licentiate degree is a Swedish graduate degree halfway between M.Sc. and Ph.D.

[2] The included articles have been reformatted to comply with the licentiate layout.

[3] Under submission for conference publication. Also partially published as a WiP paper in the Sixteenth IEEE International Conference on Emerging Technology & Factory Automation (ETFA11), Toulouse, France, September, 2011.

Mustafa. MRTC report ISSN 1404-3041 ISRN MDH-MRTC-264/2012-1-SE, Mälardalen Real-Time Research Centre, Mälardalen University, March, 2012.

Additional Papers, Not Included in the Licentiate Thesis

- *On Generating Security Implementations from Models of Embedded Systems*. Mehrdad Saadatmand, Antonio Cicchetti and Mikael Sjödin. The Sixth International Conference on Software Engineering Advances (ICSEA 2011), Barcelona, Spain, October, 2011.

- *Enabling Trade-off Analysis of NFRs on Models of Embedded Systems*. Mehrdad Saadatmand, Antonio Cicchetti and Mikael Sjödin. The Sixteenth IEEE International Conference on Emerging Technology & Factory Automation (ETFA11), WiP session, Toulouse, France, September, 2011.

- *A Methodology for Designing Energy-aware Secure Embedded Systems*. Mehrdad Saadatmand, Antonio Cicchetti and Mikael Sjödin. The Sixth IEEE International Symposium on Industrial Embedded Systems (SIES11), Västerås, Sweden, June, 2011.

- *Toward a Tailored Modeling of Non-Functional Requirements for Telecommunication Systems*. Mehrdad Saadatmand, Antonio Cicchetti and Mikael Sjödin. The Eighth International Conference on Information Technology : New Generations (ITNG), Las Vegas, USA, April, 2011.

- *On the Need for Extending MARTE with Security Concepts*. Mehrdad Saadatmand, Antonio Cicchetti and Mikael Sjödin. The Second International Workshop on Model Based Engineering for Embedded Systems Design (M-BED 2011), Grenoble, France, March, 2011.

Contents

I

Thesis

Chapter 1

Introduction

The main goal in the design of a software system is to deliver a product which satisfies all the requirements of different stakeholders. Requirements are generally categorized into functional and non-functional. Although non-functional requirements usually receive less attention in the design of many types of systems (for example in desktop applications) they play an important role in certain domains; particularly, real-time embedded systems. Satisfaction of non-functional requirements, such as timing, is critical in these systems and can determine the success or failure of the final product. In this thesis, we look at different challenges of satisfying non-functional requirements within the context of model-driven development of real-time embedded systems. In our work which extends from the system model down to the execution platform, we propose solutions and methods that help to better satisfy non-functional requirements.

1.1 Basic Terms and Definitions

In this section, we provide definitions for some of the key terms that are used throughout the thesis.

We consider two categories of requirements: *functional* and *non-functional*. In the simplest form, functional requirements are those which define *what* the system should do, while the term non-functional requirement is used for requirements which specify *how* a system should perform, or as suggested in [1] "a non-functional requirement is an attribute of or a con-

straint on a system". There is a big number of suggested definitions for non-functional requirements which are discussed in [2]. These requirements are usually described with terms that end with 'ility' such as availability, 'ity' such as atomicity while a few other ones such as performance and user-friendliness do not follow this pattern. According to the IEEE standards, 610.12-1990 and ISO/IEC/IEEE 24765:2010(E) [3, 4], the following definitions are provided for requirement, functional and non-functional requirement, and also *derived requirement*:

- Requirement:
  1. a condition or capability needed by a user to solve a problem or achieve an objective.
  2. a condition or capability that must be met or possessed by a system, system component, product, or service to satisfy an agreement, standard, specification, or other formally imposed documents.
  3. a documented representation of a condition or capability as in (1) or (2).
  4. a condition or capability that must be met or possessed by a system, product, service, result, or component to satisfy a contract, standard, specification, or other formally imposed document.
- Functional Requirement:
  1. a statement that identifies what a product or process must accomplish to produce required behavior and/or results.
  2. a requirement that specifies a function that a system or system component must be able to perform.
- Non-Functional Requirement: a software requirement that describes not what the software will do but how the software will do it (i.e., design constraints). Examples: software performance requirements, software external interface requirements, software design constraints, and software quality attributes. Non-functional requirements are sometimes difficult to test, so they are usually evaluated subjectively.
- Derived Requirement:
  1. a lower-level requirement that is determined to be necessary for a top-level requirement to be met.

2. a requirement that is not explicitly stated in customer requirements, but is inferred from contextual requirements (such as applicable standards, laws, policies, common practices, and management decisions) or from requirements needed to specify a product or service component.

In this thesis, we use the term extra-functional as a synonym to non-functional, however, to be consistent with the style that is used in the literature, we use it as an adjective for properties as in the phrase "extra-functional properties". This brings us to the next term which is *property*. It is defined in IEEE standard ISO/IEC/IEEE 24765:2010(E) [3] as:

> A responsibility that is an inherent or distinctive characteristic or trait that manifests some aspect of an object's knowledge or behavior (responsibility: A generalization of properties (attributes, participant properties, and operations) and constraints. An instance possesses knowledge, exhibits behavior, and obeys rules. These are collectively referred to as the instances responsibilities. A class abstracts the responsibilities in common to its instances. A responsibility may apply to each instance of the class (instance-level) or to the class as a whole (class-level) [5]).

In this work, we distinguish between requirements and properties by considering the former as an expression of a *need* (possibly informal), and the latter as a statement that is usually asserted formally and can be therefore proven and analyzed (e.g., calculated response time of a component). This implies that "a requirement can require that a certain property holds (e.g., absence of deadlock, meeting deadline, not overflowing a queue, etc.) and that in order for a property to hold a number of requirements may have to be met, which we normally neither express nor assert formally"[1].

Based on these definitions, we use the term *satisfaction* for requirements and *preservation* for extra-functional properties (i.e., to keep properties within their acceptable and valid ranges and protect them from violation). Accordingly, "response time of a component should not exceed 2ms" is a requirement, while "response time of component A never exceeds 2ms" and "response time of component B is equal to 1ms" are expressions of properties in a system.

Balancing trade-offs among requirements can incur adjusting system properties that are related to those requirements. Moreover, we consider a relationship between an NFR and extra-functional properties of the system in the sense that, in order to satisfy an NFR, its related extra-functional properties

[1] These definitions have been provided and formulated with the help of Prof. Tullio Vardanega.

should have valid values. For example, to satisfy performance requirements of a real-time system, execution and response time values (among others) should remain within a valid range. Moreover, a property per se does not tell much about its validity. Only when a property is considered along with its related NFR or NFRs, it becomes possible to evaluate whether it is valid for a specific system and design or not. For example, a component with the worst-case execution time of 100ms can be acceptable in designing one system but the same component can violate the requirements of another system and may not be appropriate for it.

1.2 Background and Motivation

Embedded computer systems are systems that are designed to operate as part of other devices. These systems are usually designed for specific and dedicated functions, and interact with their external environment through sensors and actuators [6, 7]. This interaction often brings along real-time requirements. However, besides real-time requirements, resource constraints in these systems also introduce other limitations with respect to properties such as energy consumption, performance, and safety. Due to resource constraints that these systems have, the correct functionality of the whole system depends heavily on the satisfaction of its NFRs. On the other hand, NFRs are often interconnected and cannot be considered in isolation. An example of such interconnection and dependency can be observed more explicitly between security and performance requirements in a system. Therefore, in satisfying a requirement, its impacts on other requirements should also be taken into account, and trade-off analysis among NFRs needs to be done [2, 8]. Handling a big range of requirements, establishing balance among them and performing trade-off analysis contribute to the high design complexity of real-time embedded systems and make it a challenging task [7].

Model-Driven Development (MDD) is a promising approach in raising abstraction level and reducing design complexity of real-time embedded systems, which also enables analysis of the system at earlier phases of development. This helps with the identification of problems in the system design before reaching the implementation phase [9, 10]. The implementation of the system can also be generated from the design models through a set of model transformations. In this context, non-functional requirements are captured by expression of extra-functional properties that are attached to model elements. In an integrated model-driven and component-based approach, the model elements

that extra-functional properties are annotated on can be components.

However, in order to ensure correctness of the generated system, it is important to ensure that in each step, extra-functional properties that are specified on the model are preserved. This means that for each transformation, input properties should be preserved in the output of the transformation. It also includes the system execution at runtime which is the result of executing the generated source code. This implies that the execution platform should be able to actively monitor and enforce extra-functional properties at runtime. To this end, the execution platform also requires to be *semantically aware* of the specified properties and related events in order to monitor them and detect their probable deviations.

1.3 Problems and Contributions Overview

There are several challenges in satisfying non-functional requirements using a model-driven development method for real-time embedded systems. These challenges include (but are not limited to) issues such as how to model NFRs, how to model their interdependencies, and also how to ensure that the implementation that is generated from the model behaves as expected during execution and not in a way that impairs the satisfactions of NFRs at runtime.

As the first step, we introduce an approach using Unified Modeling Language (UML) for modeling non-functional requirements and extra-functional properties in telecommunication systems (as an example and sub-domain of real-time embedded systems). It is done by suggesting adoption from and combining several already existing UML-based languages. In satisfying an NFR, its impacts on other NFRs should also be taken into account. To enable model-based trade-off analysis of NFRs, we propose a UML profile for modeling dependencies and impacts of NFRs and functional parts that contribute to the satisfaction of each NFR in a positive or negative way. The concepts that are provided in this UML profile are based on the Soft-goal Interdependency Graph (SIG) approach that is used in the NFR Framework [11].

As an example of interdependency of NFRs, we consider security and timing requirements in real-time embedded systems. To satisfy security requirements certain mechanisms, such as encryption, should be applied. However, adding such security mechanisms may result in the violation of the timing requirements of the system. In the context of an integrated model-driven and component-based approach, we propose an approach to automatically derive the component model of a system that includes components implementing se-

curity mechanisms (from an original component model without security components). This is done by identifying and annotating sensitive data flows in the model. The derived component model uses the same meta-model as the original model, and therefore, enables using the same timing analysis applicable to the original model. This way, timing impacts of added security components can be analyzed at earlier phases and before the implementation phase.

Modeling NFRs and including them along with functional elements, and analyzing the model do not guarantee that the system that is generated from the model is capable of satisfying those NFRs. Violations in this respect can not only happen during transformations to generate intermediate models and finally code, but also during execution of the generated code on the platform. To mitigate such violations, extra-functional properties of the system can be enforced and monitored during runtime. Upon detection of such violations, adaptation and reconfiguration of the system may be performed as a counteractive measure. We have developed an adaptive approach for balancing the security level of a system with its timing requirements at runtime. The approach is based on keeping a history of the timing behaviors of encryption algorithms and using a weaker but less time-consuming one when a timing violation occurs. The introduced adaptation mechanism is suitable for complex real-time systems where timing analysis is not practical or not much information about timing characteristics of each individual task is available.

On the other hand, in order to monitor and detect violation of extra-functional properties at runtime, the execution platform should have certain capabilities. We discuss these capabilities for timing requirements which are of great importance in real-time embedded systems. We also introduce the concept of second-layer scheduler for monitoring of real-time events, such as deadline misses and execution time overruns, on top of a real-time operating system with a priority-based preemptive scheduling policy to provide such capabilities.

1.4 Thesis Outline

The thesis is organized into two parts:
Part I includes three chapters. Chapter 1 provided an introduction to the thesis and formulated the research problem. In Chapter 2, the research overview describing detailed research goals and contributions is offered. In Chapter 3 we summarize the work and present suggestions for the future work.
Part II presents the technical contributions of the thesis in detail in the form of

research papers which are organized in Chapters 4-8.

Chapter 2

Research Overview

In this thesis, we target some of the challenges related to the satisfaction of non-functional requirements in the context of model-driven development of real-time embedded systems. Towards this goal, we also discuss techniques that help with the preservation of extra-functional properties in these systems and mitigate their deviations from the desired behavior. The importance of this research objective becomes more obvious and is motivated considering the following points:

- Satisfaction of non-functional requirements is important in the design of real-time embedded systems. A system design which cannot satisfy its non-functional requirements (e.g., timing requirements) can mean failure of the end product.

- The non-functional requirements are mapped to architectural models and captured by extra-functional properties.

- Analysis is performed on models of embedded systems in order to calculate extra-functional properties of the system and its components, such as response time, and to evaluate satisfaction feasibility of its non-functional requirements. Analyses are based on some assumptions and preconditions. Violation of these assumptions equals to the violation and invalidation of the analysis results.

- At runtime, several factors such as transient loads, difference between the ideal execution environment (taken into account for analysis) and the

actual one, can lead to violation of the assumptions that were used to perform analysis [12].

- It may not be practical and/or economical to perform analysis on all types of extra-functional properties. In such cases, runtime monitoring and handling of violations can be used to *preserve* extra-functional properties.

In this context, the term *preserve* that we use in our work implies and is based on the assumption that there is some knowledge about valid and invalid values (e.g., range of values) that an extra-functional property can have in a specific design.

Satisfaction of non-functional requirements in real-time embedded systems has been considered in this thesis in the context of model-driven development and particularly Model-Driven Architecture (MDA) methodology that is suggested by Object Management Group (OMG) [13]. UML is the key modeling language at the heart of MDA. Figure 2.1 depicts the MDA design flow that defines the development context of this thesis.

As shown in figure 2.1, different types of analyses are performed at different levels throughout the transformation chain for code generation to ensure correctness of the design. The idea here is to provide a read-only Platform-Specific Model (PSM) to the user, and also make the manual editing of the code unnecessary. This is important in order to maintain design consistency. As a consequence, the results of the analyses are propagated back to the Platform-Independent Model (PIM), so that the user can identify parts of the model that need to be modified in order to achieve the desired behavior.

2.1 Research Goals

Aligned with the context that was described in the previous section the following research goals have been defined:

G1: Evaluation of UML approaches for modeling NFRs in telecommunication systems: In order to include NFRs in the design models of a system, the modeling language that is used to design the system should provide concepts for modeling of NFRs. Considering the characteristics of telecommunication systems and the problems that were observed in developing such systems in terms of their NFRs, we have proposed a UML-based approach using a combination of several existing UML profiles [14] for modeling of NFRs

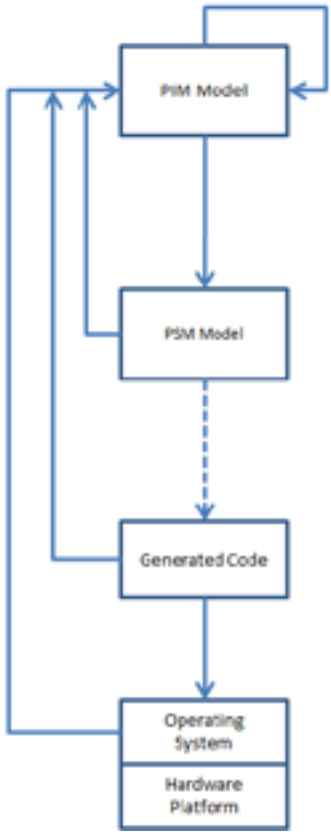

Figure 2.1: Model-driven development context

in these systems. We did this by looking at the EAST-ADL [15] modeling language which is defined for automotive domain using a similar approach and by adopting from a set of different UML profiles. We have also touched upon benefits and drawbacks of using UML profiles for defining new modeling concepts compared to defining a Domain-Specific Language (DSL) [16] from scratch.

G2: Providing an approach for model-based trade-off analysis of NFRs: To model interdependency of NFRs and identify their impacts on each other,

we have proposed a UML profile. The profile offers necessary concepts to model an NFR along with it refinements which can include one or several other NFRs as well as functional parts that contribute to its satisfaction. This way, it enables to create a hierarchy of NFRs, form child-parent relationship among them, and also establish relationships to the functional elements in the model that provide realization and implementations of NFRs.

To enable trade-off analysis of NFRs in a quantitative manner, we introduce numerical properties as part of the defined stereotypes in the profile. This allows to calculate the satisfaction level of an NFR by taking into account both the contribution degree of each of its children NFRs and any negative or positive impact that other NFRs in the system may have on it.

G3: Identification and development of methods for balancing extra-functional properties of real-time embedded systems: Timing properties are of utmost importance in real-time embedded systems. Also as mentioned earlier, interdependencies of different extra-functional properties cannot be neglected in the design of these systems. Therefore, for this research goal, we have specifically looked at the interdependency of timing and security properties as an example of such relationships and how we can establish balance among them. Basically, we categorize the approaches for the aforementioned problem into *static* and *dynamic* (adaptive), which are explained below and may also be applicable for other extra-functional properties as well.

- One approach to establish balance between timing and security properties in a system is to identify parts of the system (i.e., sensitive data) in the model that need to be protected, add security features to protect them, and finally perform timing analysis on the derived model to ensure that the added security features do not violate the timing requirements. This method leads to a *static* design in the sense that a specific security mechanism, which is analyzed, and thus, known to execute within its allowed time budget is always used in each execution. We have proposed and implemented a method to automatically derive the component model of the system including components that implement its security requirements. The original component model of the system is used as input for a transformation that considers the sensitive data flows in the original system model and adds appropriate security components. The key ideas here are to facilitate implementation of security features for non-security experts, bring security considerations into earlier phases of development, and thus most importantly enable timing analysis of the

system including security components at the model level. This helps designers to identify problems and imbalance between timing and security at the model level and fix it before reaching the implementation phase.

- The static method may not be practical for systems with high complexity which are hardly analyzable or systems with unknown timing behaviors of their components. Instead, for such systems, a *dynamic* way to select appropriate security mechanisms, based on the state of the system, can be used to adapt its behavior at runtime, and stay within the timing constraints. To this end, we have suggested an approach, along with its implementation for selecting appropriate encryption algorithms at runtime (in terms of their timing behaviors) in an adaptive way. In this approach, the timing behavior of each execution of the encryption procedures is logged, and used as feedback for selecting a more suitable encryption algorithm in the next execution.

G4: Identification and development of methods for runtime monitoring of extra-functional properties, focusing on timing properties: The above mentioned methods are not enough per se for runtime preservation and enforcement of extra-functional properties and require some support mechanisms from the underlying platform, which brings us to the next problem that exists in achieving this goal. Many of the real-time operating systems that are used as execution platform provide support for a priority-based scheduling paradigm. However, in such platforms, there is often no explicit specification on how to define and introduce different types of real-time tasks (i.e., periodic, sporadic, aperiodic) in the system. Therefore, for example, periodic behavior is actually implemented by using timer interrupts and this way a periodic task is created. This means that the platform has no concept of periodic task and there is no observable runnable entity in the system as a periodic task. Since the platform has no awareness about the type of task it is scheduling and its timing parameters such as deadline and worst-case execution time, monitoring and detection of real-time events such as deadline misses and execution time overruns are not supported and need to be implemented by end users in arbitrary ways. Monitoring of timing behavior of the system at runtime is necessary in order to ensure preservation of timing requirements. We have proposed the idea of the second-layer scheduler which enables users to specify real-time tasks in detail using their timing parameters (period, deadline, etc.). By having awareness about the type of each real-time task that is defined in the system, the second-layer scheduler is capable of providing detailed monitoring information. This monitoring information, among others, include deadline misses, execution time overruns,

task preemptions, and completion times.

2.2 Research Contributions

The following published research results cover the research goals described in the previous section, as summarized in Table 2.1.

Research Goals	Papers
G1	A
G2	B
G3	C, D
G4	E, D

Table 2.1: Mapping of published papers to research goals

- **Paper A: UML-Based Modeling of Non-Functional Requirements in Telecommunication Systems**; Mehrdad Saadatmand, Antonio Cicchetti, Mikael Sjödin; The Sixth International Conference on Software Engineering Advances (ICSEA 2011), Barcelona, Spain, October, 2011

  Abstract: In this paper, we propose a UML-based solution, consisting of different modeling languages, to model non-functional requirements in telecommunication domain, and discuss different challenges and issues in the design of telecommunication systems that are related to these requirements.

  Contribution: I have been the initiator and main author of the paper.

- **Paper B: Modeling and Trade-off Analysis of NFRs**; Mehrdad Saadatmand, Antonio Cicchetti, Mikael Sjödin; MRTC report ISSN 1404-3041 ISRN MDH-MRTC-267/2012-1-SE, Mälardalen Real-Time Research Centre, Mälardalen University, April, 2012

  Abstract: In this paper, we focus on establishing balance and enabling trade-off analysis of Non-Functional Requirements(NFR) and identify what information about NFRs is required in order to perform trade-off analysis. We propose and explain an approach to incorporate this information into system models in order to enable trade-off analysis. Our approach is based on UML profiling method to annotate model elements with necessary information.

Contribution: I have been the initiator and main author of the paper.

- **Paper C: Modeling Security Aspects in Distributed Real-Time Component-Based Embedded Systems**; Mehrdad Saadatmand, Thomas Leveque; 9th International Conference on Information Technology : New Generations (ITNG), Las Vegas, Nevada, USA, April, 2012

  Abstract: This paper introduces concepts and mechanisms that allow to model security specifications and derive automatically the corresponding security implementations by transforming the original component model into a secured one taking into account sensitive data flows in the system. The resulted architecture ensures security requirements by construction and is expressed in the original meta model; therefore, it enables using the same timing analysis and synthesis as with the original component model.

  Contribution: I have been the initiator and co-author of the paper. The implementations were mainly done by the main author.

- **Paper D: Design of Adaptive Security Mechanisms for Real-Time Embedded Systems**; Mehrdad Saadatmand, Antonio Cicchetti, Mikael Sjödin; 4th International Symposium on Engineering Secure Software and Systems (ESSoS'12), Eindhoven, The Netherlands, February, 2012

  Abstract: In this paper, we target the timing requirements of real-time embedded systems, and introduce an approach for choosing appropriate encryption algorithms at runtime, to achieve satisfaction of timing requirements in an adaptive way, by monitoring and keeping a log of their behaviors. The approach enables the system to adopt a less or more time consuming (but presumably stronger) encryption algorithm, based on the feedback on previous executions of encryption processes. This is particularly important for systems with high degree of complexity which are hard to analyze statistically.

  Contribution: I have been the initiator and main author of the paper.

- **Paper E: The Role of Schedulers in Model-Driven Development of Real-Time Systems**; Mehrdad Saadatmand, Mikael Sjödin, Naveed Ul Mustafa; MRTC report ISSN 1404-3041 ISRN MDH-MRTC-264/2012-1-SE, Mälardalen Real-Time Research Centre, Mälardalen University, March, 2012

  Abstract: Model-driven development is a promising approach to cope with the design complexity of these systems. It helps to raise abstrac-

tion level, perform analysis at earlier phases of development, and enables generation of code from the models. In this context, capabilities of schedulers, as part of the underlying platform, play an important role. They can affect the complexity of code generators, and how the model is implemented on the platform. Also, the way a scheduler monitors timing behaviors of tasks, and schedules them can facilitate extraction of runtime information. This information is then used as feedback to the original model, in order to identify parts of the model that may require to be re-designed and modified. This is especially important for round-trip support in model-driven development of real-time systems. In this paper, we describe our work in providing these features by introducing a second layer scheduler on top of OSE real-time operating systems scheduler. The approach contributes to the predictability of the system by bringing more awareness to the scheduler about the type of real-time tasks (i.e., periodic, sporadic, and aperiodic) that are to be scheduled, and the information that should be monitored and logged for each type.

Contribution: I have been the initiator and main author of the paper.

2.3 Research Methodology

Figure 2.2 depicts and summarizes the key steps that have taken place in performing this research work.

Identification of NFRs in telecommunication systems, the general and specific challenges related to them, and the investigation of the state of the art and practice to understand what has been done to cope with those challenges resulted in a set of preliminary research goals. Providing and implementing solutions for those research goals also required investigation of the state of art and practice for other challenges and problems as well. The future work and directions which have been identified after implementing a solution for a research goal also resulted in new research goals which in turn required more investigation.

Figure 2.2: Research Methodology

Chapter 3

Conclusions

In this research work, we focused on the importance of NFRs in real-time embedded systems. We discussed different steps and points during the process of model-driven development of these systems where problems leading to the violation of the requirements can occur. It was shown how at the system model level, NFRs can be modeled along with other parts of the system, and traceability links among them can be established. Since the satisfaction of an NFR, especially in real-time embedded systems, can often affect the satisfaction of other NFRs of the system, we proposed a generic approach that enables system designers to compare design models with respect to the satisfaction level of their NFRs and perform trade-off analysis. This is achieved by considering the interdependencies of NFRs as well as the impact of the functional parts. This empowers system designers to make better decisions before continuing with the rest of the development process and generating implementations.

Focusing on timing and security requirements, two approaches for establishing balance between these NFRs were also introduced. These approaches target the problem of interdependency of NFRs and ensuring the balance among them. We believe that the suggested solutions can also be considered and modified for some other NFRs as well, but this needs to be further evaluated.

Regarding the proposed second layer scheduler, while it adds the missing necessary mechanisms that are required for detailed monitoring of timing events, its added overhead and performance costs should also be taken into account. For different systems, this additional overhead may be or not be acceptable. This solution is especially interesting for RTOSes where it is not

possible or not desirable to modify the kernel and the core scheduler. On the other hand, an alternative solution would be to instead include the features that we introduced as part of the second layer scheduler inside the core scheduler which helps with the reduction of the overheads. One point to note is that any added feature such as monitoring capabilities will have its performance costs anyway. This emphasizes the importance of efficient monitoring of system properties in real-time embedded systems which deserves further studies.

3.1 Future Work

While here we mainly looked at different steps during model-driven development of real-time embedded systems where violation of NFRs can occur, to further mitigate such violations it is also important to look at the transitions between each of these steps which in this context is model transformation (both model-to-model and model-to-text transformations). Investigation of transformation rules that preserve extra-functional properties of the system and thus can contribute to the satisfaction of NFRs is an interesting extension to this work. Such transformations that can be proven and provide preservation guarantees are of special interest in the development of safety-critical systems and in certification of the development process.

Also, by introducing our approach for trade-off analysis of NFRs and comparing different design alternatives, it would be interesting as a future work to study methods that help with the optimization of design models with respect to their NFRs. One of the problems in this direction that needs to be considered is the scalability of the approach for large systems and issues of state explosion type. Providing a more precise and detailed approach for quantification of NFRs is another topic for further research. Moreover, performing trade-off analysis of NFRs at runtime to re-configure the system according to different states/modes and match different Quality of Service (QoS) levels (e.g., if the available energy level goes below a certain limit) is another interesting future work.

Here we took timing and security requirements as an example of NFRs, to discuss dependencies and their impacts on each other and provided solutions (static and dynamic) for establishing balance among them. Another direction of this work could be to evaluate the applicability of the suggested methods for other NFRs such as timing and energy consumption.

Regarding the runtime monitoring of extra-functional properties, there are some issues that deserve special attention as future work. For some properties

and in some systems, the difference between the time point when the value of a property is requested and the time when the value is actually monitored and obtained can affect the accuracy and usefulness of the monitored value. This is important considering that the monitoring task needs also to compete with the (main) tasks that implement an application. Therefore, for such situations, we are considering to address this problem by providing a priority-based monitoring in the sense that by assigning priorities for different properties, the user can specify that the accuracy of which properties to monitor are more important for him. Based on these priorities, the monitoring task can perform differently to increase the accuracy of the monitored value while reducing the response time for obtaining it. We leave the implementation and evaluation of this method as a future work.

Bibliography

[1] Martin Glinz. On non-functional requirements. In *15th IEEE International Requirements Engineering Conference*, pages 21–26, New Delhi, India, October 2007.

[2] Lawrence Chung and Julio Cesar Prado Leite. Conceptual modeling: Foundations and applications. chapter On Non-Functional Requirements in Software Engineering, pages 363–379. Springer-Verlag, Berlin, Heidelberg, 2009.

[3] Systems and software engineering – Vocabulary (IEEE Standard). *ISO/IEC/IEEE 24765:2010(E)*, 15 2010.

[4] IEEE Standard Glossary of Software Engineering Terminology. *IEEE Std 610.12-1990*, 1990.

[5] IEEE Standard for Conceptual Modeling Language Syntax and Semantics for IDEF1X/Sub 97/ (IDEF/Sub Object/). *IEEE Std 1320.2-1998*, 1998.

[6] S. Heath. Embedded systems design. EDN series for design engineers, ISBN: 9780750655460. Newnes, 2003.

[7] Thomas Henzinger and Joseph Sifakis. The embedded systems design challenge. In Jayadev Misra, Tobias Nipkow, and Emil Sekerinski, editors, *FM 2006: Formal Methods*, volume 4085 of *Lecture Notes in Computer Science*, pages 1–15. Springer Berlin / Heidelberg, 2006.

[8] Luiz Marcio Cysneiros and Julio Cesar Sampaio do Prado Leite. Nonfunctional requirements: From elicitation to conceptual models. In *IEEE Transactions on Software Engineering*, volume 30, pages 328–350, 2004.

[9] Bran Selic. The pragmatics of model-driven development. *IEEE Software*, 20:19–25, September 2003.

[10] Martin Törngren, DeJiu Chen, and Ivica Crnkovic. Component-based vs. model-based development: A comparison in the context of vehicular embedded systems. In *Software Engineering and Advanced Applications, 2005. 31st EUROMICRO Conference on*. IEEE, August 2005.

[11] Lawrence Chung, Brian A. Nixon, Eric Yu, and John Mylopoulos. *Non-Functional Requirements in Software Engineering*, volume 5 of *International Series in Software Engineering*. Springer, 1999.

[12] S.E. Chodrow, F. Jahanian, and M. Donner. Run-time monitoring of real-time systems. In *Real-Time Systems Symposium, 1991*, pages 74 –83, dec 1991.

[13] Model-Driven Architecture (MDA). `http://www.omg.org/mda/`, Accessed: February 2012.

[14] Bran Selic. A systematic approach to domain-specific language design using UML. In *Proceedings of the 10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing*, Washington, DC, USA, 2007. IEEE Computer Society.

[15] EAST-ADL Specification V2.1. `http://www.atesst.org`, Accessed: November 2011.

[16] Ingo Weisemöller and Andy Schürr. A comparison of standard compliant ways to define domain specific languages. pages 47–58, Berlin, Heidelberg, 2008. Springer-Verlag.