Contents

1. Introduction

REAL-TIME EMBEDDED systems are usually characterized by timely computation, which is identified as *Deadline*, besides correct output of the computation [**?**]. They are applied in many safety-critical applications, e.g., the braking system in vehicles, besides applying the appropriate force on the wheel, the time taken to slow down (or halt) the vehicle must be timely otherwise accident can occur. Therefore, safety-critical real-time systems should be analyzed rigoruously for functional safety, for instance, the ISO 26262 standard "Road vehicles-Functional safety" [8] suggests early analysis of automotive systems, that is during requirements specification and software design, and the use of formal methods, which are mathematical techniques and tools that enable unambiguous specification and modeling, and rigorous analysis of systems [18].

In distributed computing, the safety-critical software is mapped on multiple hardware systems to capitalize on the computational power provided by the distributed architecture, e.g., the executing of the braking software on multiple electronic control units (ECU). Since the distributed software is normally exposed to a greater degree of permanent and transient faults, reliability of the safety-critical software should be maximized to improve dependability of the system which requires additional critical systems resource such as power and energy besides computational resources. However, the embedded hardware is usually resource constrained, therefore, the software should be efficiently mapped to the hardware to conserve critical system resources, but also to accommodate current and future growth of the software functionality.

In this thesis, we apply formal methods to improve the requirements specifications of safety-critical systems, and to analyze the functional and timing of the safety-critical software against the specifications. Safety-critical requirements specifications should be unambiguous, comprehensible, etc. In fact, according standards, e.g., ISO 26262, semi-formal or formal languages are recommended to specify safety-critical requirements. However, natural language is the de facto method to specify embedded systems requirements industry because it is intuitive and expressive. However, it is inherently ambiguous, consequently the natural language specifications are sometimes ambiguous, incomprehensible, inconsistent, etc [1]. Template-based specification and controlled natural language are the two most commonly used methods to improve requirements specifications. The template-based specification methods, e.g., requirements boilerplates [7],

etc., lack meta-model for extensible and the template selection is usually cumbersome. Controlled natural languages, e.g., Attempto [6][5], etc., mimic the intuitiveness of natural language and have formal semantics, however, lacks support for embedded systems, hence are less effective. In this thesis, we propose a constrained natural language which is domain-specific and uses the notion of boilerplates to facilitate reuse. The specifications have semantics in Boolean logic and description logic to enable rigoruous analysis via Boolean satisfiability and ontology, respectively.

The specifications are employed in subsequent system development including software design to verify the latter for correct functionality. The software design is usually modeled, simulated and analyzed before implementation. In this regard, Simulink is one of the most widely used development environment for multi-domain, multi-rate, discrete and continuous safety-critical systems in industry [9]. For this main reason, there is increasing interest in formal analysis of Simulink models [16][19]. Simulink Design Verifier, which is based on the Prover model-checker, is the de facto tool in the Simulink environment to formally verify Simulink design models. However, it has limited functionality, e.g., it supports only discrete models, has issues with scalability, and lacks verification of timed properties. We propose transformation of Simulink models to network of stochastic time, subsequently analyzing the latter via statistical model checking, which is uses traces of executions and statistical analysis technques, e.g., monte-carlo simulation, etc., to check properties, that is in contrast to exhaustive model checking. The statistical model checking is shown to scale well on complex and large problems unlike the exact model checking, which suffers from state-space explosion.

The software design is mapped to hardware, which should take into consideration effectiveness and efficiency. The software should be effectively mapped to the execution platform, that is satisfying the timing and reliability requirements of a distributed safety-critical software. The timing considers fixed-priority preemptive scheduling of tasks, the end-to-end delay of chains (or sequence of tasks), and the reliability maximization employs fault tolerance via active replication of software components on different computing units. Furthermore, the allocation should be efficient so that the power consumption is minimized to ensure extensibility of the software, that is to accommodate future growth of the software functionality, which is evident for instance in the automotive electical/electronic systems where hundreds of software functions are executed. The software-to-hardware allocation problem is NP-hard, so the exact methods, which deliver optimal solutions, e.g., branch and bound, dynamic programming, etc., are usually used for relatively small problems. On the other hand, heuristics, which deliver near-optimal solutions, are applied on large and complex problems. We propose formulation of the software allocation as an integer-linear programming, subsequently solve via branch and bound Furthermore, we

propose hybrid-particle optimization to solve large software allocation problems.

1.1 Research Contributions Overview

In this subsection, we give overview of the thesis contributions, and later in Section x, the contributions are further discussed in detail.

- **Formal Analysis of natural language requirements:** we propose a fairly expressive, flexible yet structured and domain-specific constrained natural language, called *ReSA* [14][15]. The language has semantics in Boolean and description logic to support for shallow and rigorous analysis, respectively. The Boolean specifications are checked for consistency using the satisfiability-modulo theory via the Z3 SMT solver. Whereas, the description logic is used to encode the specification as ontology, where we check consistency of the specifications at the lexical level using Reasoner (Inference engine) such Hermit. The ReSA tool, which consists of an editor and implements consistency-checking functionality, is integrated seamlessly into EATOP, which is an open source EAST-ADL IDE, to complement the requirements modeling.

- **Scalable analysis of Simulink models:** we propose a pattern-based, execution-order preserving automatic transformation of atomic and composite Simulink blocks into stochastic timed automata that can be formally analyzed using UPPAAL Statistical Model Checker [2]. Our method is scalable, and has been validated on industrial use cases [4]. The statistical model checker analyzes a state-transition system by conducting statistical analysis on the collected traces of the system executions, effectively mitigating the state-space explosion of (exact) model checking [12].

- **Efficient Power consumption ILP and metaheuristics:** we propose an integer-linear programming (ILP) model to the allocation of distributed software on the network of heterogeneous computing units, which have different processor speed, failure rate and power consumption specifications. The ILP implemented in JAVA using the ILOG CPLEX interface, and subsequently solved the CPLEX solver.

- **Validation on industrial use cases:** Our contributions such as its the ReSA language as well as the proposed formal analysis of Simulink model is validated on industrial use cases, which are provided

  Our solutions are evaluated on industrial automotive use cases and on a realistic benchmark. The formal analysis of the natural language

requirements specifications in ReSA and the formal analysis of Simulink models are evaluated on the adjustable speed-limiter (ASL) and brake-by-wire (BBW) systems provided by Volvo Group Trucks Technology (VGTT). ASL is a speed-limitation automotive function which controls the vehicle speed of Volvo trucks from speeding up, and is useful in roads where speed-limitation signs are in place. The ASL use case consists of around 300 requirements, which are specified in natural language, architectural models in EAST-ADL and Simulink models. The integrated software allocation is evaluated on the engine management system benchmark provided by Bosch [] provided for AUTOSAR applications. The benchmark consists of statistics of the schedulable objects, such as mean values, shars of timing specifications and activation mechanisms of the schedulable objects in the system.

1.2 Thesis Outline Overview

The thesis is divided into two parts. The first part is a summary of our research. It is organized as follows: in Chapter 2, we give the background information on description logic, Boolean satisfiability problem, Simulink, stochastic timed automata, and meta-heuristic optimization. In Chapter 3, we explain the research problem and outline the research goals. The thesis contributions are discussed in Chapter 4, followed by the related work in Chapter 5. In Chapter 3, we describe the research method applied to conduct the research. Finally, in Chapter 7, we conclude the thesis and outline possible directions for future work.

2. Preliminary

3. Problem Formulation

The thesis is motivated by the need for advanced (or rigorous) requirements specification, modeling and analysis of safety-critical automotive systems, essentially to improve the existing methods and tools of automotive systems development at VGTT and Scania.

It is also inspired by the increasing complexity of automotive functionality implemented by an electrical/electronic system inside a modern truck, which is resource constrained, hence the need for efficient use of critical system resources such as power and energy besides computation and communication resources. Thus, the *overall goal* of the thesis is to:

> *provide assurance and extensibility of safety-critical system design, at the various levels of abstraction, via formal analysis and optimization techniques*

The overall goal is refined via *research goals*, which state the needs or concerns that the thesis should address and are formulated as follows:

3.1 Research Goals

Safety-critical automotive systems are developed according to the ISO 26262 standard, including the development process, methods and tools, etc. The standard requires the use of semi-formal specification languages to specify requirements less ambigous and comprehensible specifications, which are usually constrained natural languages, such as templates, e.g., requirements boilerplates, etc., and controlled natural languages, e.g., Attempto, PING, etc.

The template-based methods inherently lack meta-model (or grammar), therefore is difficult to add new templates effectively, moreover, template selection is usually cumbersome. The existing controlled natural languages lack effective support of specifying embedded systems requirements.

Thus, the first research goal is to:

RG 1:	*reduce ambiguity and improve the comprehensibility of natural-language requirements using domain-specific knowledge of embedded systems.*

One of the mechanisms to improve natural language specifications is by constraining the language, including its syntax, semantics and the lexicon [10]. The design of a constrained natural language for the specification of requirements is not trivial mainly because: i) by constraining the language, its expressiveness and intuitiveness can be impaired [1][17], therefore, appropriate trade-offs should be made during the design in order to have a robust and effective specification language; ii) domain knowledge/expertise is highly needed.

Requirements should be analyzed in ensemble in order to detect errors that span multiple specifications, e.g., logical contradictions. However, natural language lacks formal (or precise and unambiguous) semantic, therefore is difficult to rigoruously analyze (or reason) natural-language requirements specifications. There are several methods to natural language semantics, of which logic is the most applied method. Thus, the second research goal is to:

RG 2:	*facilitate formal analysis of the requirements specifications through transformation to Boolean and description logics*

Natural language specifications are constructed from syntactic units, such as words, phrases, clauses, statements, etc. Consequently, rigorous analysis of specifications involve parsing and interpreting the syntactic units, which is a complex problem in computational linguistics [3]. The depth of the interpretation greatly affects the applicability of the methods, e.g., the propostional logic representation is simple and the analysis scales well, however, it is shallow as it abstracts the details. On the other hand, first-orde-logic representations are more rigor, thus enable thorough analysis, but are less tractable. Therefore, proper use of the methods is crucial to benefit from the semantics.

The software designs and software-design units (or behavioral models) should conform to the requirements specifications. In this thesis, we consider the software-design units are modeled in Simulink, which is the most widely used model-based development environment in industry to model and simulate the dynamics of multi-domain, discrete, continous embedded systems. Simulink also supports the generation of code from discrete Simulink models that directly execute on specific platforms, thus is crucial to conduct rigoruous analysis of Simulink models to reduce errors introduced at generated code.

The automotive Simulink models that we encounter at VGTT and Scania are in the scale of thousands blocks and are realize complex safety-critical functionality. The de facto Simulink analysis techniques, e.g., by type checking, simulation, and formal verification via the Simulink Design Verifier (SDV[1]) are not sufficient to address the full correctness of safety-critical real-time Simulink models. SDV lacks support for checking temporal correctness as specified in timed properties, e.g., in TCTL, and also lacks support for verifying continuous models and suffers from scalability due to its reliance on the exact model-checking [13]. In contrast to exact model checking, statistical model-checking collects sufficient traces of system simulations, and consequently applies statistical methods to verify properties. It scales better, and the accuracy of the analysis can be improved by taking many traces of simulations. Thus, the fourth research goal of the thesis is to:

RG 3:	*enable scalable formal analysis of multi-rate and hybrid Simulink models using statistical model-checking*

Simulink consists of connected and hierarchical Simulink blocks, which encode mathematical functions [9]. For industrial systems, the number of blocks in a Simulink model can be in the order of thousands, and the blocks can be triggered with different sampling frequencies for discrete blocks and without any sampling frequency for continuous blocks. Therefore, typical industrial Simulink models are usually complex and comprise mixed signals, multiple rates, discrete and continues Simulink blocks, making formal analysis challenging.

In the distributed computing, the automotive software is distributed on multiple computing units (or ECU). In this case, the greater failure risks of the distributed automotive software necessitate maximizing system reliability such by implementing fault tolerance, e.g., using redundant software software functionality on multiple ECU, which requires additional computation, I/O and more power. In this regard, the software-to-hardware allocation process plays a crucial role, that is effective and efficient allocation should satisfy the safety-critical software system requirements such as timing and reliability, but also should mimiize the resource consumptions.

Thus, the third research goal is to

RG 4:	*Minimize power consumption of distributed safety-critical software while satifying timing and reliability requirements, in the allocation process of software to hardware.*

[1] https://se.mathworks.com/products/sldesignverifier.html

Software allocation is NP hard and is difficult to find a solution in the general case. However, for less complex problems, exact methods, e.g., using integer-linear programming, etc., works, however, for large and complex probelms, the exact methods are limited, in stead, heauristics is usually applied. In this thesis, we assume, fixed-preemptive scheduling policy and timing analysis based on response time, furthermore, we consider end-to-end timing analysis simultaneously, as a result, which the software allocation is trivial.

In order to show the validity of our proposed solutions, a working prototype should be developed and should also evaluated on industrial uses cases. The validation should consider scalability and engineer-friendliness of methods and tools besides effectiveness. Thus, the last research goal is to:

RG 5:	*Provide automated and engineering-friendly support for the requirements specification, software allocation of embedded and formal analysis of Simulink models.*

Seamless integration of our proposed methods and tools into the existing development process require close cooperations between the domain experts and the practitioners. The role of the domain experts should be to simplify usage of the tools, e.g., by rendering their interface to exisiting once, etc., and the practitioners should cooperate from providing to materials to the validation of the tools, which is not trivial considering the challenge of forma methods, and companies culture for being restrctive.

4. Contributions

5. Research Method

6. Related Work

7. Conclusions and Future Work

Bibliography

[1] ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes –Requirements engineering. *ISO/IEC/IEEE 29148:2011(E)*, pages 1–94, 12 2011.

[2] Peter Bulychev, Alexandre David, Kim Gulstrand Larsen, Marius Mikučionis, Danny Bøgsted Poulsen, Axel Legay, and Zheng Wang. UPPAAL-SMC: Statistical Model Checking for Priced Timed Automata. *Electronic Proceedings in Theoretical Computer Science*, 2012.

[3] Alexander Clark, Chris Fox, and Shalom Lappin. *The Handbook of Computational Linguistics and Natural Language Processing*. 2010.

[4] P. Filipovikj, N. Mahmud, R. Marinescu, C. Seceleanu, O. Ljungkrantz, and H. Lönn. *Simulink to UPPAAL statistical model checker: Analyzing automotive industrial systems*, volume 9995 LNCS. 2016.

[5] Norbert E. Fuchs, Kaarel Kaljurand, and Tobias Kuhn. Attempto Controlled English for Knowledge Representation. pages 104–124. Springer, Berlin, Heidelberg, 2008.

[6] Norbert E Fuchs and Rolf Schwitter. Attempto Controlled English {(ACE)}. *CoRR*, cmp-lg/960, 1996.

[7] Elizabeth Hull, Ken Jackson, and Jeremy Dick. *Requirements Engineering*. Springer London, London, 2011.

[8] ISO26262 ISO. 26262: Road vehicles-Functional safety. Technical report, ISO/TC 22/SC 32 Electrical and electronic components and general system aspects, 2011.

[9] Thomas L. Harman James B. Dabney. *Mastering Simulink*. Pearson, 2003.

[10] Tobias Kuhn. A Survey and Classification of Controlled Natural Languages. *Computational Linguistics*, 40(1):121–170, 3 2014.

[11] Edward Ashford Lee and Sanjit Arunkumarr Seshia. *Introduction to Embedded Systems: A Cyber-Physical Systems Approach*. 2011.

[12] Axel Legay, BenoÃ®t Delahaye, and Saddek Bensalem. Statistical Model Checking: An Overview. pages 122–135. Springer, Berlin, Heidelberg, 2010.

[13] Florian Leitner and Stefan Leue. Simulink Design Verifier vs. SPIN a Comparative Case Study. *Proceedings of FMICS*, 2008.

[14] N Mahmud, C Seceleanu, and O Ljungkrantz. ReSA Tool: Structured requirements specification and SAT-based consistency-checking. In *2016*

Federated Conference on Computer Science and Information Systems (FedCSIS), pages 1737–1746, 9 2016.

[15] Nesredin Mahmud, Cristina Seceleanu, and Oscar Ljungkrantz. ReSA: An ontology-based requirement specification language tailored to automotive systems. In *2015 10th IEEE International Symposium on Industrial Embedded Systems, SIES 2015 - Proceedings*, pages 1–10, 2015.

[16] Karthik Manamcheri, Sayan Mitra, Stanley Bak, and Marco Caccamo. A step towards verification and synthesis from simulink/stateflow models. In *Proceedings of the 14th international conference on Hybrid systems: computation and control - HSCC '11*, page 317, New York, New York, USA, 2011. ACM Press.

[17] Andriy Myachykov, Christoph Scheepers, Simon Garrod, Dominic Thompson, and Olga Fedorova. Syntactic flexibility and competition in sentence production: The case of English and Russian. *Quarterly Journal of Experimental Psychology*, 2013.

[18] Gerard OâRegan. Concise guide to formal methods, 2017.

[19] Steve Sims and Daniel C. DuVarney. Experience report. In *Proceedings of the 2007 ACM SIGPLAN international conference on Functional programming - ICFP '07*, page 137, New York, New York, USA, 2007. ACM Press.

[20] Jiacun Wang. *Real-Time Embedded Systems*. 2017.

[21] Yanyun Wang. Developing Safety Critical Embedded Software under DO-178C, 2016.