
# LICENTIATE THESIS PROPOSAL

# Ontology-based Specification of Structured Requirements and Scalable Model checking of Embedded Systems Models

Nesredin Mahmud
School of Innovation, Design and Engineering (IDT)
Mälardalen University
nesredin.mahmud@mdh.se
November 2016

**Main Supervisor: Cristina Seceleanu**
**Co-supervisor: Guillermo Rodriguez-Navas**

## Abstract

*Safety-critical embedded systems, such as safety functions in automotive systems, control systems in industrial automation, chemical processes, avionics, medical equipment are developed in such a way that software bugs that could lead to catastrophes, like loss of human lives or property damage, are eliminated or at least reduced already at design time. The complexity of developing these systems is growing overtime due to increasing number of software functions running on constrained hardware resources. Therefore, a systematic approach is required in order to address the aforementioned challenges, and improve software quality. Model-based development and formal methods are viable approaches to address the challenges. These approaches have been used for decades to systematically model, and analyze safety-critical systems at stages before implementation. Formal methods is used to precisely specify embedded systems mathematically, thus enabling automated analysis using techniques such model checking. However, applications of formal methods are scarcely found in industry due to several reasons. First, the mathematical notation that formal methods use for specification is not common to software developers; second, tools that automate the application of formal methods are rarely integrated with industrial practices, in a seamless way. The integration of Prover's Plug-in products into MATLAB/Simulink, or the SCADE toolchain are successful examples of such integrations of formal methods with modeling languages used in industry.*

*The Achilles heel of embedded systems' development is for long considered to be the requirements capturing and specification. Usually, system requirements are captured informally using natural language, e.g., English. However, due to the inherent ambiguity, as well as flexibility of natural language, it is likely that some requirements are imprecisely specified, especially for large and complex systems. Moreover, natural language requirements are not amenable to analysis based on formal techniques. In contrast to natural language, semi-formal specification methods provide a more structured, and precise specification, e.g., tabular specifications, requirements boilerplates, properties specification patterns. However, these benefits are at the expense of less flexibility, and less engineering friendliness to natural language. Though many of the semi-formal specification methods have formal semantics, e.g., Property Specification Patterns (PSP) in several temporal logics, Software Cost Reduction (SCR) using the four-variable model, these methods still lack the flavor of natural language syntax. To address such challenges, in this thesis we propose requirements specification language, ReSA, and an automated consistency-checking technique for requirements expressed in ReSA. The language extends requirements boilerplate with domain knowledge for effective specification and analysis of requirements. The specifications in ReSA are transformed to propositional logic in order to be checked for logical consistency using Boolean Satisfiability Problem (SAT) technique. Further, ReSA specifications are transformed to temporal logic properties that can be further fed into a model checker for formal analysis of behavioral models.*

*In industry, such behavioral models are often described in diagrammatic forms in environments such as MATLAB/Simulink, which provide means for creating executable specifications that can be simulated and analyzed for design errors, but also from which code is being generated. Therefore, ensuring that Simulink models meet the associated requirements is necessary for correct functionality. To address the above desideratum, we propose a formalization of Simulink blocks as stochastic timed automata that we show how to statistically model check using UPPAAL SMC. Our transformation preserves the execution order of Simulink blocks, and uses a set of timed automata patterns for encoding the behavior of discrete and continuous Simulink blocks. Our techniques proposed in this thesis are demonstrated using prototype toolchains that provide automated support for specifying requirements in ReSA, and for checking the logical consistency of the boolean equivalent formulas of the ReSA requirements. We validate our approaches on industrial use cases from the automotive industry, that is, Adjustable Speed Limiter (ASL) and Brake-By-Wire(BBW) from Volvo Group Trucks Technology.*

# 1 Introduction

Embedded systems are ubiquitous, thus are found in different industrial domains, e.g., automotive, avionics, industrial automation, chemical processes domains. These systems execute dedicated software (embedded software) on resource-constrained hardware platforms. For safety-critical applications, such as anti-lock braking and flight control, embedded software programs are tested extensively to reduce, or eliminate errors during development. As the complexity of embedded software programs increase at a fast pace due to increasing number of functions running on the constrained hardware platforms, the likelihood of not capturing all errors by testing is higher than before. Therefore, appropriate methods and tools are required to effectively develop and analyze embedded systems, which should enable engineers to capture non-trivial software bugs, e.g., ambiguities and inconsistencies in requirements specifications, violation of safety and timing requirements. One way to accomplish this is by applying formal methods during model-based embedded software development. Formal methods use mathematical theories to specify systems. In industry formal methods are rarely applied in the development of embedded systems mainly due to lack of efficient methods and tools that meet industrial needs, e.g., cost-effectiveness, engineering friendliness, etc.

In a model-based development approach, engineering artifacts, such as system requirements, architectural and behavioural models, are pervasive in model-based embedded systems development. Requirements specifications describe functional and extra-functional requirements that a system needs to satisfy, and most specifications of requirements in industry are expressed using natural language, such as English. Natural language is flexible and intuitive for engineers to use; however, it is inherently ambiguous [37][34], thus could lead to ambiguous requirements specifications. If we consider the following bogus requirement expressed in English: `Cruise Control System shall limit vehicle speed, and also Adjustable Speed Control`. The requirement is syntactically ambiguous, because the dependent clause "and also Adjustable Speed Control", is not clearly conjugated to the independent clause "Cruise Control System shall limit vehicle speed". The interpretations are either `Cruise control System shall limit vehicle speed and Adjustable Speed Control`, or that `Adjustable Speed Control shall limit vehicle speed`. Moreover, natural language is too liberal so requirements could contain also inconsistent statements, especially for complex systems specifications. In order to fix such problems, several meetings, discussions, and rewriting of requirements is a common practice in industry, usually reducing efficiency in project management. As opposed to natural language, semi-formal specification methods, such as requirements boilerplate [21][27][28], property specification pattern [22][11][33], tabular specification [16], are less ambiguous, and allow more precise specification at the expense of less flexibility and intuitiveness.

Given the above described issues, in this licentiate proposal, we describe our contributions with respect to structuring requirements specifications and checking logical consistencies of requirements. Our structured requirements specification approach combines the advantages of requirements boilerplate with domain knowledge of requirements specification in embedded systems. We provide a meta model for our requirements boilerplate to support the creation and extensibility of boilerplates, and an ontology for capturing domain knowledge at system level. The ontology is an artifact that consists of system level taxonomies (e.g., System, Parameter, Device, etc) and syntactic relations for the semantic validation of requirements specifications. The contribution the core part of our proposed language ReSA - a constrained natural language for the specification of requirements in embedded systems. The language is also extended to express requirements for EAST-ADL levels of abstractions such as vehicle level, analysis level, and design level. Before verifying if a model satisfies requirements, it is effective to check first if the requirements are consistent to each other. For this, we check the logical consistency of requirements expressed in ReSA using the Boolean Satisfiability Problem (SAT) technique. If the conjunction of all analyzed requirements modeled as Boolean formulas is satisfied, an interpretation that satisfies the the specifications is generated, otherwise, the set of requirements is not satisfiable, hence inconsistent. Once requirements are consistent, we provide a mechanism to express properties in ReSA in order to simplify the complexity of temporal notations. This is accomplished by automatically transforming ReSA specifications to properties expressed in temporal logic such as Timing Computational Tree Logic (TCTL), and Weighed Metric Temporal Logic (WMTL). This approach should allow engineers to express temporal properties close to natural language, and use the properties in the verification of behavioural models using model checking.

The behavioural modeling language, MATLAB/Simulink [1] is a popular framework for model-based embedded software development in industry. The Simulink framework uses a block diagram environment for modeling, and provides a simulation platform to observe the dynamics of a system before its actual implementation. Further, Simulink allows static-code analysis, and formal verification using the Simulink Design Verifier (SDV) model checker, as it is often the case that code is generated directly from Simulink models , consequently their correctness assurance is a must. The existing tool SDV enables the identification of hidden design errors, such as integer overflow, out-of-bound array access, dead logic, etc. However, the

verifier lacks some capabilities, e.g., deadlock detection, modeling and checking of temporal properties, and also has issues regarding scalability. Recent endeavors propose other solutions for verifying Simulink models, such as validation of Simulink Diagrams using PVC [35], statistical model checking using Plasma Lab [8], transformation of Simulink models into hybrid system, etc. However, none of these have been applied on industrial system models, therefore their scalability has not been evaluated. In this thesis, We propose a pattern-based transformation of Simulink models into timed automata with stochastic semantics, suitable for simulation and formal analysis using the statistical model checker UPPAAL SMC [9]. In our transformation, we also propose a fattening algorithm for composite Simulink blocks that preserves the block execution order generated by Simulink. Statistical model checking provides a viable alternative to the state-space explosion of model checking for large scale software programs [14][36]. Last but not least, we validate the approach by applying on industrial Simulink models of Adjustable Speed Limiter (ASL) and Brake-By-Wire (BBW).

The rest of this proposal is organized as follows: In Section 2 we define the research goals, and scientific contributions of the thesis, and this followed by the research method in Section 3. In Section 4, we show the outline of the thesis, followed by the progress and time plan in Section 5. Finally, we discuss the related work in Section 6, after which, in Section 7, we present the conclusion and directions for future work.

## 2 Research Description and Scientific Contributions

The research problems are illustrated using research goals. The goals are targets met by the scientific contributions made during the course of the research through scientific publication of papers, and deliverables (documentation, prototype toolchains) to our VeriSpec project (http://www.es.mdh.se/projects/343-VeriSpec). In this section, we present the research goals, the papers included in the thesis, and our contributions that address the research goals.

### 2.1 Research Goals

The research is conducted in partnership with Volvo Group Trucks Technology(VGTT), and Scania, Sweden, within the VeriSpec project. Therefore, the research goals are catered to to address some of the challenges that the heavy-vehicle automotive industry is facing during the development of their systems. However, our goals and contributions fit a larger number of embedded systems domains. Out of these challenges, we focus on the specification and analysis of requirements and verification of automotive behavioural models. Our overall research goal is as follows:

**Overall Goal:** Provide engineers with effective and mathematically rigorous software quality improving methods and tools for the development of embedded systems based on formal techniques.

Since the overall goals is too broad to address as such, we refine it into five subgoals that are discuss subsequently. The first subgoal addresses the challenges of requirements specification in the development of embedded systems in industry. Most requirements in industry are expressed using natural language, consequently, they are prone to bad quality, e.g., incomprehensibility, ambiguity, inconsistency, etc. In contrast, requirements expressed in formal notations such as Z, temporal logic, etc, are unambiguous. However, specifying requirements in such languages require special training of engineers, which incurs cost increases. Therefore, an intermediate requirements specification language that could restrict natural language to prevent ambiguous specifications, while allowing for requirements consistency analysis would be beneficial. Consequently, our first subgoal is stated as follows:

**Subgoal 1:** Improve the structure and comprehensibility of embedded systems requirements specifications via requirements boilerplates that employ domain-specific knowledge.

In this subgoal, we address the challenge using boilerplates that bear the advantages of natural language (i.e., syntactic and semantic similarity), and by incorporating domain-specific knowledge represented formally using an ontology. The ontology validates the semantic correctness of requirements specification via type checking, and semantic relations between instances of the ontology concepts. Since the main goal aims at providing software quality improving methods for embedded systems, the engineering artifacts need to be verified against a set of desired properties.

In order to be used further for verification, the structured requirements need to be given precise meaning, such that they can be used as properties that behavioral models can be checked against. Likewise, other artifacts such as behavioral models need to be given formal semantics as a prerequisite for rigorous analysis. Therefore, our second subgoal is stated below:

**Subgoal 2:** Enable the precise description of selected engineering artifacts such as requirements specifications and behavioural models in mathematically-defined frameworks, as a prerequisite to formal analysis of the artifacts.

Once we enable the precise description of engineering artifacts, we need to be able to check and analyze these artifacts in order to depict potential trouble spots early on. Since Simulink is a development environment often used in industry, but also a platform for generating code from models, one has to be equipped with adequate techniques to analyze and get insight into its behavior. Therefore, we formulate the third subgoal as follows:

**Subgoal 3:** Devise techniques for checking the consistency of requirements specifications, and for scalable formal analysis of behavioral models such as Simulink models.

Ensuring the quality of engineering artifacts is crucial not just for the correct implementation of embedded systems, but also for improved documentation, and effective communication among stakeholders. Moreover, identification of errors at early stages of software development is easier, and less costly [6]. In this research, we tackle the problem of logical inconsistencies in requirements specifications, and this is followed by verification of behavioural models against those specifications using statistical model checking. The challenges associated to the above subgoal are twofold: first, the robustness and and scalability of the techniques is crucial, as these techniques are applied for large scale applications in industry; second, the feedback provided at the end of the analysis should be usable to the engineer. Hence, root causes of possible inconsistencies or the sources of counter-examples need to be localized in order to improve the usefulness of the methods. Once the problems are located, the engineer may opt to modify either the requirements specifications, or the behavioural model to assure the consistency between these artifacts. Without automated support, the techniques are hardly usable, especially for large industrial systems, and their effectiveness is therefore compromised. To address this challenge, we propose a new research subgoal, as follows:

**Subgoal 4:** Provide effective and engineering-friendly automated toolchain support for the specification and analysis of requirements and verification of behavioural models.

One of the main problems with adopting formal techniques in industry is lack of suitable engineering tools. The tools need to be engineering friendly, effective, and easy to integrate to existing tools. Therefore, it is crucial to spend time with engineers, understand existing tools, and learn the development processes. Our automated toolchains should consist of functions for the specification and analysis of requirements, and verification of behavioural models based on formal techniques. Further, the toolchains should hide the complexity of formal techniques while providing a user-friendly interface to the engineer. According to the ambition of our main goal, we need to ensure that the toolchain support is applicable, usable and can handle the challenges of industrially-sized. The final subgoal states as follows:

**Subgoal 5:** Validate the proposed quality improving techniques, and their automated support on various industrial use cases.

In order to check the effectiveness of our research artifacts, we need to validate them on industrial systems. Meeting this subgoal entails discovering the strengths, as well as limitations of our proposed solutions, and the contexts in which they work properly. The use cases consist of two operational automotive safety functions, including Adjustable Speed Limiter (ASL) from AB Volvo, and one academic use case system, that is the Brake-By-Wire system from AB Volvo. The uses cases consist of engineering artifacts, such as requirements documents in textual forms, state machines, high-level diagrams, and design models in Simulink. The challenge is to formulate effective methodologies to conduct the validation process that allow us to objectively, and critically evaluate our toolchain, thus our approach within the scope of the overall goal. Moreover, the validation process should incorporate mechanisms to capture feedbacks from engineers for further development of our approaches and the toolchain itself.

## 2.2 Papers Included in the Thesis

The papers that are going to be included in the licentiate thesis are as follows.

### 2.2.1 Paper A

**ReSA: An ontology-based requirement specification language tailored to automotive systems**
Nesredin Mahmud, Cristina Seceleanu, Ljungkrantz Oscar

**Abstract:** *Automotive systems are developed using multi-leveled architectural abstractions in an attempt to manage the increasing complexity and criticality of automotive functions. Consequently, well-structured and unambiguously specified requirements are needed on all levels of abstraction, in order to enable early detection of possible design errors. However, automotive industry often relies on requirements specified in ambiguous natural language, sometimes in large and incomprehensible documents. Semi-formal requirements specification approaches (e.g., requirement boilerplates, pattern-based specifications, etc.) aim to reduce requirements ambiguity, without altering their readability and expressiveness. Nevertheless, such approaches do not offer support for specifying requirements in terms of multi-leveled architectural concepts, nor do they provide means for early-stage rigorous analysis of the specified requirements. In this paper, we propose a language, called ReSA, which allows requirements specification at various levels of abstraction, modeled in the architectural language of EAST-ADL. ReSA uses an automotive systems' ontology that offers typing and syntactic axioms for the specification. Besides enforcing structure and more rigor in specifying requirements, our approach enables checking refinement as well as consistency of requirements, by proving ordinary boolean implications. To illustrate ReSA's applicability, we show how to specify some requirements of the Adjustable Speed Limiter, which is a complex, safety-critical Volvo Trucks user function.*

**Status:** Published in the 10th IEEE International Symposium on Industrial Embedded Systems (SIES), 2015, IEEE.
**My contribution:** I was the main driver of the paper. I developed the specification of the language, including the ontology for the requirements specification. The other two authors contributed with important ideas, and comments.

### 2.2.2 Paper B

**ReSA Tool: Structured Requirements Specification and SAT-based Consistency-checking**
Nesredin Mahmud, Cristina Seceleanu, Ljungkrantz Oscar

**Abstract:** *Most industrial embedded systems requirements are specified in natural language, hence they can sometimes be ambiguous and error-prone. Moreover, employing an early-stage model-based incremental system development using multiple levels of abstraction, for instance via architectural languages such as EAST-ADL, calls for different granularity requirements specifications described with abstraction-specific concepts that reflect the respective abstraction level effectively. In this paper, we propose a toolchain for structured requirements specification in the ReSA language, which scales to multiple EAST-ADL levels of abstraction. Furthermore, we introduce a consistency function that is seamlessly integrated into the specification toolchain, for the automatic analysis of requirements logical consistency prior to their temporal logic formalization for full formal verification. The consistency check subsumes two parts: (i) transforming ReSA requirements specification into boolean expressions, and (ii) checking the consistency of the resulting boolean expressions by solving the satisfiability of their conjunction with the Z3 SMT solver. For validation, we apply the ReSA toolchain on an industrial vehicle speed control system, namely the Adjustable Speed Limiter.*

**Status:** Published in the 36th IEEE Software Engineering Workshop, 2016, IEEE CS.
**My contribution:** I was the main driver of the paper. I developed the toolchain, and the other two authors contributed with important ideas and comments.

### 2.2.3 Paper C

**Simulink to UPPAAL Statistical Model Checker: Analyzing Automotive Industrial Systems**
Predrag Filipovikj, Nesredin Mahmud, Raluca Marinescu, Cristina Seceleanu, Oscar Ljungkrantz, Henrik Lönn

**Abstract:** *The advanced technology used for developing modern automotive systems increases their complexity, making their correctness assurance very tedious. To enable analysis by simulation, but also enhance understanding and communication, engineers use MATLAB/Simulink modeling during system development. In this paper, we provide further analysis means to industrial Simulink models by proposing a patternbased, execution-order preserving transformation of Simulink blocks into the input language of UPPAAL Statistical Model checker, that is, timed (or hybrid) automata with stochastic*

Table 1: Scientific contribution of the included papers to the research goals

| | Subgoal 1 | Subgoal 2 | Subgoal 3 | Subgoal 4 | Subgoal 5 |
|---|---|---|---|---|---|
| Paper A | X | | | | |
| Paper B | | X | X | X | X |
| Paper C | | X | X | X | X |
| Paper D | | X | | | |

*semantics. The approach leads to being able to analyze complex Simulink models of automotive systems, and we report our experience with two vehicular systems, the Brake-by-Wire and the Adjustable Speed Limiter.*

**Status:** In Proceedings of the 21st International Symposium on Formal Methods (FM), 2016, LNCS, Springer.

**Comment:** The first three co-authors contributed equally to writing the paper. Technically, I equally contributed with proposing the pattern-based semantics of Simulink blocks, together with Predrag Filipovikj. I implemented the patterns library for the Simulink blocks, and manually encoded the BBW system. Predrag Filipovikj developed the flattening algorithm, and implemented the tool for the automatic transformation of Simulink models into a network of timed automata with stochastic semantics. Raluca Marinescu contributed with analyzing the BBW system, Cristina Seceleanu contributed with designing the methodology, writing the abstract and introduction, and with useful ideas and comments. The industrial coauthors provided the use cases and commented on the final draft.

#### 2.2.4 Paper D

**Automated Transformation of Specifications in Constrained Natural Language into Temporal Logic Requirements**
Nesredin Mahmud, Cristina Seceleanu

**Abstract:** *Formal methods allow rigours analysis using automated techniques, e.g., theorem proving, model checking. They may be used to find hidden design errors by exploring the program's state space exhaustively, thus augmenting software testing. In order to facilitate their application in industry, formal languages need to be engineering friendly, which is not the case for most formal languages. For instance, the temporal logics, such as Linear Temporal Logic (LTL), Computational Tree Logic (CTL), Property Specification Language (PSL), are frequently used to describe temporal properties in model checking, however, their syntax is not natural to engineers. As a result, it is challenging, and timing consuming for engineers to express properties (or requirements) using such temporal logics, and comprehend what the properties really mean. In this paper, we propose property specification using constrained natural language that have interpretations in Timed CTL (TCTL). TCTL is the extended version of CTL with timing semantics, and it widely used to express properties in model checking for timed systems.*

**Status:** In progress. To be submitted to COMPSAC 2017, January 2017.
**My contribution:** I will be the main author of the paper, with respect to developing the transformation algorithm, and to writing the article.

### 2.3 Scientific Contributions

In this subsection, we present a summary of the contributions of the thesis. Each contribution is related to the goals, and the included papers described in Subsection 2.1 and **??**, respectively. Table 1 shows the relationship between the goals and the papers.

**Contribution 1:** A domain-specific language (ReSA) for the structured specification of embedded systems requirements based on boilerplates and a system-level ontology for requirements specification.

The contribution is included in Paper A. We propose a requirements specification language, ReSA. The syntax and semantics of the language resembles natural language (English). The ReSA language allows for more structured and constrained specifications, as well as more precise, at the expense of flexibility if compared to natural language. The following example shows an instantiation of the ReSA language:`if state# <System> is <State>; and <User> <ActOnInDev> <InDevice>;then actioin# <System> shall be <State>; within <time><unit>; endif`. The language uses boilerplates as the primitives based on which requirements are specified. The boilerplates are found within `#` and `;` delimiters, and the keywords are `if`, `then`, and `endif`. A boilerplate, such as `<System> shall be <State>`, consists of dynamic syntactic elements (`<System>`, `<State>`), and a static syntactic element (`shall be`). The language is constrained through a system-level ontology of requirements specifications. A system-level ontology captures the high level concepts, relations among instances of the concepts, and axioms that describe allowed relations in the ontology. The ontology for the requirements specification contains system concepts, syntactic relations and axioms that govern requirements specification, e.g., the axiom `term:system->term:State` states "an instance of System precedes an instance of State" in specification, and e.g., `Cruise Control:system shall be activated:state` is a valid instantiation. The ontology is developed through empirical analysis on use cases from automotive industry, and discussions with engineers.

**Contribution 2:** Formal semantic definition of engineering artifacts such as requirements specification in ReSA and Simulink blocks.

Contribution 2, which is presented in Papers B, C, and Paper D, addresses Subgoal 2.

**2.1. Formal semantics of ReSA requirements in Boolean logic and TCTL:** In Paper B, we show the transformation of requirements specifications in ReSA to intermediate propositional formulas, and eventually to SMT-LIb2 format for consistency checking using the Z3 SMT solver. In Paper C, we the show transformation of requirements specifications in ReSA into temporal specifications in TCTL. The temporal specifications act as properties to be verified using model checking on timed automata models, such as UPPAAL models. For example, the following requirement expressed in ReSA:

ReSA

if mode# <vehicle> is in <"running"> mode; and
interaction# <driver> <presses> <ASLButton>;
then
state# <ASL> shall be <activated>; within <250><ms>;
endif
*/*Original Format - with tags*/*

(OR)

if vehicle is in running mode and driver presses ASLButton
then
ASL shall be activated within 250 ms
endif
*/*Plain Format - without tags*/*

is transformed into specifications expressed in propositional and temporal logics as follows:

Propositional Logic:

$(p1 \wedge p2) \mapsto c3$

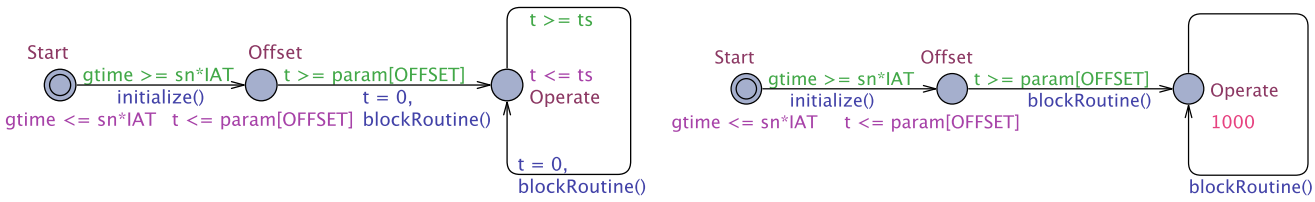
;where, $p1$ = "<vehicle> is in <"running"> mode;",
$p2$ = "<driver> <presses> <ASLButton>;",
$p3$ = "<ASL> shall be <activated>; within <250><ms>;"

Timed CTL:

$AG((p1 \wedge p2) \mapsto AFp3_{\leq t})$

;where, $t$ = "within <250><ms>;"

**2.2. A Pattern-based Transformation of Simulink models into the Stochastic Timed Automata Framework:** The semantics of Simulink models is given by transforming them into network of Stochastic Timed Automata (STA). A Simulink model represents the behaviour of an embedded system using communicating functional blocks known as Simulink blocks. In our approach, the transformation steps are described as follows: (i) first, we identify the most used Simulink blocks by analyzing two automotive use cases: BBW and ASL of Volvo GTT, and based on this we classify the blocks as atomic and composite blocks, but also as continuous-time and discrete-time blocks based on their execution semantics; discrete-time blocks execute periodically with sample time $Ts$, whereas, Continuous-time blocks execute on infinitely small time intervals that approximate continuous behaviour; (ii) second, we propose a tuple definition to an atomic Simulink block as follows: $B = \langle V_{in}, V_{out}, V_D, t_s, Init, blockRoutine \rangle$, where: $V_{in}$, $V_{out}$, $V_D$, are set of input, output, data variables, respectively, $t_s$ is a sample time, $Init$ is the initialization function and $blockRoutine$ is the function that maps input and data variables to




(a) Pattern for discrete blocks

(b) Pattern for continuous blocks

Figure 1: Our Proposed Timed Automata Patterns

output variables; (iii) third, we propose Timed Automata (TA) patterns for the continuous-time and discrete-time blocks as shown in Figure 1; (iv) fourth, a flattening algorithm is developed in order to transform a hierarchical Simulink model into a flat model; (v) as a last step, we carry out the validation of our approach by formally verifying industrial Simulink models of Adjustable Speed Limiter (ASL) and Brake-By-Wire (BBW)System.

**Contribution 3:** Automated consistency checking of ReSA requirements, and statistical model checking of Simulink models.

This contribution is found in Paper B, and Paper C, and addresses Subgoal 3. In paper B, we show a technique for checking the consistency of requirements using the SAT technique, that is, the consistency checking problem is formulated as a boolean satisfiability problem, that is, if the conjunction of all requirements $\Psi = \psi_i \dots \psi_n$ is satisfiable, and the requirements are considered consistent, and hence an interpretation (or a model) that satisfies the requirements can be generated from a SAT solver. Otherwise, the requirements are inconsistent; therefore, one or more of the requirements need to be modified.

In paper C, we formally verify the networks of stochastic TA that result out of the transformation of Simulink models that we described above. We employ UPPAAL SMC to statistically model check Simulink models against functional and timing properties. We apply the approach on two industrial use cases, that is, the Adjustable Speed Limiter (ASL) and Brake-by-Wire (BBW) systems from Volvo GTT. A sample of the requirements of the ASL and BBW systems that we analyze with UPPAAL SMC are:

| Requirement | Example |
|---|---|
| R1_BBW (End-to-end deadline): | The time needed for a brake request to propagate from the brake pedal sensor to the wheel actuator should not exceed 200 ms. |
| R2_BBW (Functional requirement): | If the slip rate exceeds 0.2, then the applied brake torque shall be set to 0. |
| R1_ASL (Min.speed limit): | The ASL-EM controller shall be able to handle road speed limit requests down to 5 km/h. |
| R2_ASL (Lowest speed limit): | When several road speed limit sources are active at the same time, ASL-EM shall use the lowest speed limit value. |
| R3_ASL (Max. latency): | The maximum latency of the ASL-EM block shall be 20 ms. |

**Contribution 4:** Toolchains for the specification and consistency checking of requirements, and formally verifying Simulink models.

The contribution is shown in Papers B, and D, and addresses Subgoal 4. In Paper B, we propose a prototype ReSA toolchain with main features for specifying requirements in ReSA language, and consistency checking of the Z3 SMT solver. The toolchain is developed on Eclipse framework, and employs the XText, and Z3 SMT solver technologies for developing a domain-specific language, and satisfiability checking, respectively. In Paper D, we show a prototype tool for formal verification of Simulink models using UPPAAL SMC. Figure 4 shows the workflow when using our toolchains to specify requirements in ReSA and analyze the requirements using the Z3 SMT Solver. Further, the workflow shows transformation of a Simulink model into the language UPPAAL SMC, then checking the formalized Simulink model using UPPAAL SMC model checker against functional and timing properties expressed in ReSA.

**Contribution 5:** Applying on two industrial uses cases: Adjustable Speed Limiter and Brake-by-Wire System.

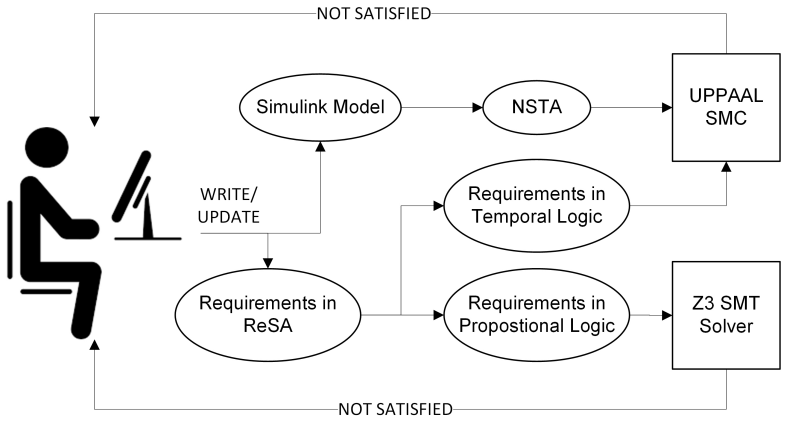



Figure 2: Consistency Checking of requirements and Verification of Simulink Model

The contribution is shown in Paper B, where we presented the feedback of ReSA toolchain from VGTT engineers. The process involved engineers from various departments, and roles, e.g., testing department, system architect, R&D using interviews, practice according to predefined scenarios, and questionnaire.

## 3 Research Method

Research methods provide systematic approaches to conduct research and solve problems. Research can be conducted through conceptual analysis, concept implementation, case study, field experiment, simulation, etc. In computing, a typical research process involves problem definition, data collection, data analysis and implementation, followed by evaluation and discussion of the result obtained from the implementation [19]. Figure 3 illustrates our research process. The research idea has its origin from industrial problems in Scania and Volvo GTT. The problems originated from lack of methods and tools to analyze software systems that help detect software bugs at early stages of software development. Due to the increasing safety functions in vehicles, and future autonomous vehicles, the need for sophisticated development and analysis methods and tools is crucial. The research has required several meetings with researchers from both companies, and investigation of existing methods and tools in industry (state of practice), and academia (state of art). In the mean time, we have organized several meetings to refine the research idea, and come up with research goals.

The research process has involved hands on experience with existing and operational software industrial engineering tools in order to understand, and analyze existing functions, determine lacking features, and reflect over the tools' engineering usability for feature improvements. This has helped the team understand imminent problems and issues in the automotive industry in particular, but also other ES industries. The proposed research solution is discussed, scrutinized from industrial interest before their implementation. The implementation consists of design, and prototype development of our proposed solutions. The implementation is internally evaluated for its correctness before exposing it for validation in industry. We have used automotive uses cases from Volvo VGTT to do the evaluation. The validation of the prototype toolchain has got its initial evaluation by engineers at Volvo GTT. This initial validation was accompanied by interview, questionnaire, and hands on exercise on the prototype toolchain. In the due course, we have written and published several scientific papers on the research topic. The research process is iteratively conducted for every research goals identified in the process.

## 4 Thesis Outline

The thesis is a collection of papers. The outline is as follows:

**I Thesis**

**1 Introduction**

**2 Preliminaries**

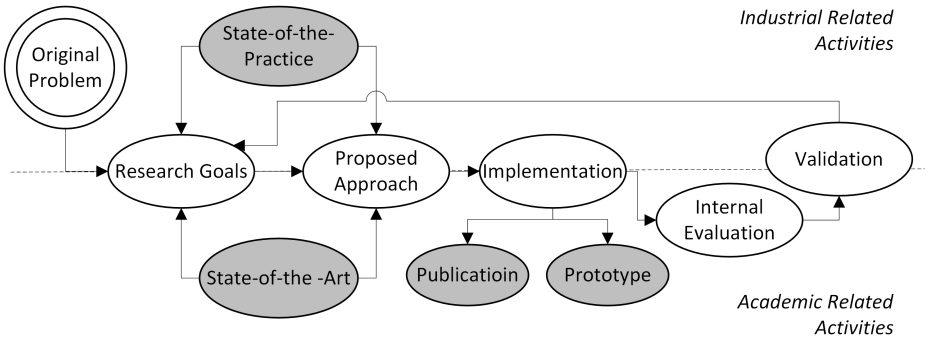



Figure 3: The research process of the thesis

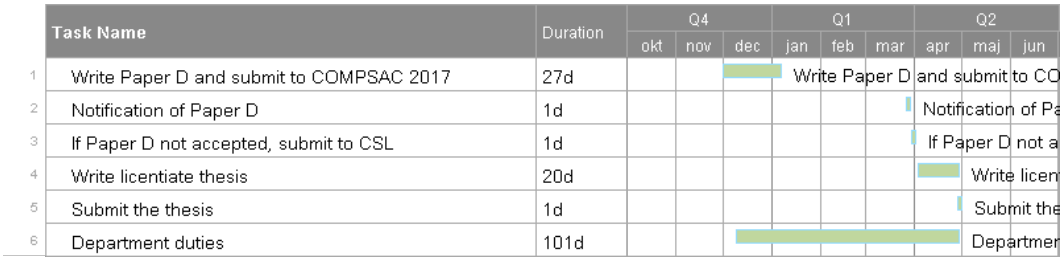



Figure 4: Time Plan

# 5 Progress and Time Plan

## 5.1 Courses

The licentiate degree requires at least 45 ECTS. This requirement will be fulfilled by the thesis defense time. The summary of the courses is listed in Table 2.

Table 2: List of courses taken

| **Courses** | **Credits** | **Status** |
|---|---|---|
| Project in Advanced Embedded Systems | 7.5 | Finished |
| Advanced Component-based Software Engineering | 7.5 | Finished |
| Research Planning | 4.5 | Finished |
| Research Methods | 7.5 | Finished |
| Advanced Validation and Verification | 7.5 | Finished |
| Principles of Cyber-physical Systems | 7.5 | Finished |
| International Summer School Marktoberdorf, July 29 to August 10, 2014 | 3.0 | Finished |
| Fourth Summer School on Formal Techniques May 19 - May 23, 2014 Menlo College, Atherton, CA | 2.0 | Finished |
| **Total Credits** | **47** | |

## 5.2 Activities and Time Plan

The remaining activities and the time plan are listed below:

## 5.3 Third Cycle Outcomes

| 1. Knowledge and understanding | Completed |
|---|---|
| 1.1a Knowledge and understanding of the research area | YES |

I have taken the following graduate courses:

- Project in Advanced Embedded Systems, 7.5
- Advanced Component-based Software Engineering, 7.5
- Research Planning, 4.5
- Research Methods, 7.5
- Industrial Systems Cloud Computing, 7.5
- Principles of Cyber-physical Systems, 7.5
- Advanced Validation and Verification, 7.5
- Safety-critical Systems Engineering, 7.5 (In Progress)

I have attended the following schools, workshops and conferences:

- The 36th IEEE Software Engineering Workshop (SEW-36), Gdansk, Poland, 11 - 14 September, 2016
- SIES'2015: 10th IEEE International Symposium on Industrial Embedded System
- IEEE Emerging Technology and Factory Automation (ETFA 2014), 16-19 September Barcelona, Spain

I have attended the following webinars, seminars, licentiate and doctoral proposals and presentations:

- licentiate/PhD seminars and presentations
- lectures given by guest professors at MDH
- seminars in other research groups at MDH
- seminar in other places (Volvo VGTT, Scania, SICS Open House)

| | |
|---|---|
| 1.1b Specialist knowledge in a defined part of the research area | YES |

I have taken the following graduate courses:

- dvanced Validation and Verification, 7.5 (ongoing)
- Safety-critical Systems Engineering, 7.5 (ongoing)

I have visited VGTT, GÃűteborg several times, and have:

- Spent time investigating existing engineering tools, and development methods
- Interviewed senior developers, group managers, and leaders from various departments
- Worked on Automotive case study, and applied research result on the use case

| | |
|---|---|
| 1.1c Deep knowledge of research methods in general, and of research methods in the specific research area | YES |

I have taken the following graduate courses:

- dvanced Validation and Verification, 7.5 (ongoing)
- Safety-critical Systems Engineering, 7.5 (ongoing)

I have visited VGTT, GÃűteborg several times, and have:

- Spent time investigating existing engineering tools, and development methods
- Interviewed senior developers, group managers, and leaders from various departments
- Worked on Automotive case study, and applied research result on the use case

Remaining Activities

- Paper C publication
- Paper D writing and publication

| **2. Skills and abilities** | **Completed** |
|---|---|
| 2.1a Critically and independently identify and formulate research questions | YES |
| • I have written scientific articles and papers indicated in Section 2.2<br>• I prepared basic or advanced level thesis proposal:<br>title: Automated Transformation of Requirements from Natural language to Formal specification<br>• I have written this thesis proposal | |
| 2.2b Contribute to the development of knowledge | YES |
| I evaluated ReSA toolchain with engineers at VGTT, Göteborg<br>I supervised basic and advanced level students:<br>• *Object Detection For High Productivity, 2015*<br>• *Automated Transformation of Requirements from Natural language to Formal specification, 2016* (Ongoing) | |
| 2.2 Orally and in writing present and discuss research results with the national and international scientific community, and the society in general | YES |
| I have attended the following scientific community for seminar, conference, workshop<br>• 19th IEEE International Conference on Emerging Technologies and Factory Automation, Barcelona<br>• The 36th IEEE Software Engineering Workshop (SEW-36), Gdansk<br>• Fuse, Joint Seminar on Functional Safety, Virtual Integration and Model Based Engineering, Göteborg<br>• I participated in VGTT, Scania södertälje workshop, and meetings | |
| 2.3 Independently conduct research and development | YES |
| • As main driver, I have been responsible for research, and publication of the scientific papers listed in Section 2.2<br>• I conducted research visits at VGTT, Scania södertälje | |

| 3. Skills and abilities | Completed |
|---|---|
| 3.1 Demonstrate ability to make ethical assessments in own research | YES |
| I have participated in the planning of my own research<br>I have taken courses related to research and planning<br>• Research Methods Course, 7.5credit<br>• Research Planning Course, 4.5credit | |
| 3.2 Demonstrate understanding of science's role and use in society, including its possibilities and limitations, and responsibility of its use | YES |
| • I participated in co-production, and project activities listed in Section 1.1a<br>• I participated in Seminars listed in Section 1.1a | |
| 3.3 Identify one's need of further knowledge and take responsibility for one's learning | YES |
| • As main driver, I have been responsible for research, and publication of the scientific papers listed in Section 2.2<br>• I conducted research visits at VGTT Göteborg, Scania Södertälje | |

### 5.4 Thesis Opponent and Committee

For the thesis defense we propose the preliminary committee as follows:

1. Opponent: Dr.habil. Bernhard Schätz, Technical University of Munich, Germany
2. Committee members:
   (a) Prof. Björn Lisper, MDH
   (b) Assoc. Prof. Marco Autilli, University of L'Aquila, Italy
3. Reserve: To be decided

## 6 Related Work

In this section, we present the related work in semi-formal specification of requirements, formal semantics of natural language , analysis and verification of models using formal techniques.

### 6.1 Semi-formal Specification of Requirements

In industry, requirements are usually expressed in natural language. However, natural language is inherently ambiguous, as a result, it could lead to erroneous requirements specification. Formal languages, such as temporal logic, Z notation are commonly used in the specification of safety critical embedded systems. However, such specification languages are symbolic, hence, difficult to be used and comprehended by engineers. As a result, semi-formal specification methods, such as Requirements Boilerplates, Specification Pattern System (SPS), tabular notation, requirements specification modeling language, are used to reduce the complexity of formal specification language, at the same time, improve the quality of the specifications. Heitmeyer [17] proposed tabular notation based Software Cost Reduction method to specify requirements in embedded systems. The tabular specification method is based on a state machine, and consists tables, such as, conditional, event, model transition tables. The SPS method [10] is a collection of patterns for specifying timing requirements overtime [23][15][13], and these patterns are defined over various temporal logics, such as LTL, TCTL, MTL . A structured English language accompanies these patterns, e.g., `Globally, it is never the case that P holds` means $AG\neg P$ in CTL, and $\Box\neg P$ in LTL, where $P$ is a propositional formula. A more intuitive method for specifying requirements relies on *requirements boilerplates*, which are close to natural language, but provide structure to requirements and guide the specification of properties, e.g., `if <Button> is pressed then <system> shall be <state>`. Due to the benefits of employing boilerplates in specification, in this thesis we take them one step further by proposing ReSA, a structured language for specifying requirements that extends the boilerplate approach towards a multi-abstraction level specification of requirements, supported by a well-defined ontology that includes domain-specific concepts.

### 6.2 Defining Formal Semantics to Engineering Artifacts

Natural language is flexible, and intuitive to engineers who work in requirements specification. However, it is lacks formal semantics, therefore, is not amenable to automated analysis. Defining formal semantics to natural language is challenging due the fact that natural language is too flexible (the same idea can be conveyed using different statements), and its lexicon can be used in ways that give rise to different interpretations. Several research endeavors exist on the formalization of natural language; it ranges from constraining the language using domain knowledge [4], to ad hoc mappings of natural language strings to strings in formal logic [3][12]. In our approach, we use domain knowledge that is represented using an ontology of system-level requirements specification, and propose the constrained natural language ReSA for specifying requirements in a structured and controlled way. Further, we develop a translation from substrings of ReSA language to propositional logic and temporal logic (TCTL). The benefit of this approach is twofold: first, the language allows an engineer to express a natural language rendering specification that can be shown to be consistent (in Z3), and second, second, the language can be used to express properties that can be verified with model checking tools that use timed automata as the input language (e.g. UPPAAL and UPPAAL SMC).

### 6.3 Formal Analysis of Requirements, and Verification of Embedded System Models

Constance et. al [18] provide a consistency checking method using the SCR model. The consistency check function checks the syntax, completeness of the variables inside the SCR tables, teachability of modes in the mode transition tables, and so forth. Amalinda et al. [32] proposed a consistency checking property which is typical in real-time systems, called rt-consistency. Further, the authors demonstrated a case where requirements can be consistent, but rt-inconsistent, simultaneously. However, the scalability of such approach to a larger industrial use case is limited due to the lack of tool support for requirements expressed in temporal logic extended with timing values. In our approach, a consistency checking problem is reduced to a satisfiability problem, that is, we check if the requirements expressed in the propositional formulas entail logical contradictions. This problem can easily be solved using existing SAT tools, e.g. Z3 that we employ in our work, for larger industrial uses cases. We have applied ReSA and our consistency checking procedure on 304 requirements of industrial system ASL, and the result is given within few seconds.

Several works have already tackled the formal analysis of Simulink models. Barnat et al. [5] and Meenakshi et al. [29] propose transformations that target only Simulink blocks with discrete-time behavior. The work of Agrawal et al. [2] focuses on the transformation of Simulink into networks of automata, without providing concrete means for formal verification. Miller [30] investigates how translating Simulink to Lustre enables formal verification with a constellation of model checkers and provers. Manamcheri et al. [26] and Jiang et al. [20] propose transformation frameworks for Stateflow diagrams, into timed and hybrid automata, respectively, yet not considering other types of Simulink blocks. Compared to these frameworks, our approach covers both continuous- and discrete-time blocks, and we show how our transformation leads to the formal analysis of industrial automotive systems models, against a wide set of requirements. This is an endeavor not really carried out before. One other solution is the use of PLASMA Lab [24], a tool that is able to take as input different Simulink simulations and provide statistical model checking results. Compared to this approach, we generate a formal model that can be extended further (e.g., with extra-functional information) to provide additional verification results.

## 7 Conclusions and Future Work

In this thesis, we propose methods and tools for the specification of requirements, propose formal semantics to engineering artifacts, and show how to analyze industrial embedded systems. For improved readability and comprehensibility of requirements, we propose a domain-specific language for requirements specification based boilerplates and supported by an ontology instantiated for the EAST-ADL-based design of automotive systems. We provide transformation mechanism from the constrained natural language to propositional and temporal logic (TCTL) for formal analysis and property specification. Further, we propose a pattern-based transformation of Simulink blocks into networks of Stochastic Timed Automata (STA) that are then analyzed against functional and timing requirements coming form the automotive domain, which is followed by verification of functional and timing properties using the UPPAAL SMC model checker. We evaluated our approach on industrial use cases from Volvo VGTT, and the scalability results are encouraging. The statistical model checking method seems to be able to handle the functional and timing analysis of systems similar in complexity to the ones that we have exercised it on, at the expense of not delivering an exact guarantee of the property, but rather a probabilistic one with a

given accuracy. However, for some complex systems, one could apply fully exhaustive model checking against safety-critical requirements, and SMC for the less critical ones.

Currently we have two distinct methods, ReSA for requirements specification and consistency checking of ReSA specifications supported by the ReSA toolchain, and the Simulink2SMC method and tool, both validated to some extent on industrial use cases.

As future work, we intend to integrate these methods into one framework supported by a tool, which could assist engineers in developing predictable embedded systems, in an attempt to meet our ambition: improve the quality of the software of embedded systems. To achieve this, we also need to conduct extensive evaluations and apply our framework to more industrial systems of various degrees of complexity.definition of formal semantics to engineering artifacts (i.e., requirements, design models), and analysis and verification of embedded systems.

## References


[1] Simulation and model-based design. https://se.mathworks.com/products/simulink.html. Accessed: 2016-12-09.

[2] A. Agrawal, G. Simon, and G. Karsai. Semantic Translation of Simulink/Stateflow Models to Hybrid Automata using Graph Transformations. *ENTCS Journal*, 109:43–56, 2004.

[3] Reyadh Alluhaibi. Simple interval temporal logic for natural language assertion descriptions. *IWCS 2015*, page 283, 2015.

[4] KM Annervaz, Vikrant Kaulgud, Shubhashis Sengupta, and Milind Savagaonkar. Natural language requirements quality analysis based on business domain models. In *Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on*, pages 676–681. IEEE, 2013.

[5] J. Barnat, J. Beran, L. Brim, T. Kratochvíla, and P. Ročkai. Tool chain to Support Automated Formal Verification of Avionics Simulink Designs. In *FMICS*, pages 78–92. Springer, 2012.

[6] Barry W Boehm. Software and it's impact: A quantitative assessment. *Datamation*, page 41, 1971.

[7] Peter Bulychev, Alexandre David, Kim Guldstrand Larsen, Axel Legay, Guangyuan Li, Danny Bøgsted Poulsen, and Amelie Stainer. *Monitor-Based Statistical Model Checking for Weighted Metric Temporal Logic*, pages 168–182. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[8] Alexandre David, Dehui Du, Kim G Larsen, Axel Legay, Marius Mikučionis, Danny Bøgsted Poulsen, and Sean Sedwards. Statistical model checking for stochastic hybrid systems. *arXiv preprint arXiv:1208.3856*, 2012.

[9] Alexandre David, Kim G Larsen, Axel Legay, Marius Mikučionis, and Danny Bøgsted Poulsen. Uppaal smc tutorial. *International Journal on Software Tools for Technology Transfer*, 17(4):397–415, 2015.

[10] Matthew B Dwyer, George S Avrunin, and James C Corbett. Property specification patterns for finite-state verification. In *Proceedings of the second workshop on Formal methods in software practice*, pages 7–15. ACM, 1998.

[11] Matthew B Dwyer, George S Avrunin, and James C Corbett. Patterns in Property Specifications for Finite-state Verification. In *Software Engineering, 1999. Proceedings of the 1999 International Conference on*, pages 411–420. IEEE, 1999.

[12] Alessandro Fantechi, Stefania Gnesi, Gioia Ristori, Michele Carenini, Massimo Vanocchi, and Paolo Moreschini. Assisting requirement formalization by means of natural language translation. *Formal Methods in System Design*, 4(3):243–263, 1994.

[13] Predrag Filipovikj, Mattias Nyberg, and Guillermo Rodriguez-Navas. Reassessing the pattern-based approach for formalizing requirements in the automotive domain. In *2014 IEEE 22nd International Requirements Engineering Conference (RE)*, pages 444–450. IEEE, 2014.

[14] Radu Grosu and Scott A. Smolka. *Monte Carlo Model Checking*, pages 271–286. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[15] Lars Grunske. Specification patterns for probabilistic quality properties. In *2008 ACM/IEEE 30th International Conference on Software Engineering*, pages 31–40. IEEE, 2008.

[16] Constance Heitmeyer, Bruce Labaw, and Daniel Kiskis. Consistency checking of scr-style requirements specifications. In *Requirements Engineering, 1995., Proceedings of the Second IEEE International Symposium on*, pages 56–63. IEEE, 1995.

[17] Constance L Heitmeyer. Software cost reduction. *Encyclopedia of software engineering*, 2002.

[18] Constance L Heitmeyer, Ralph D Jeffords, and Bruce G Labaw. Automated consistency checking of requirements specifications. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 5(3):231–261, 1996.

[19] 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? In *ACM SIGCSE Bulletin*, volume 38, pages 96–114. ACM, 2006.

[20] Y. Jiang, Y. Yang, H. Liu, H. Kong, M. Gu, J. Sun, and L. Sha. From Stateflow Simulation to Verified Implementation: A Verification Approach and A Real-Time Train Controller Design. In *RTAS'16*, pages 1–11, April 2016.

[21] Vegard Johannessen. CESAR-text vs. Boilerplates: What is More Effcient-requirements? Written as Free Text or Using Boilerplates (templates)? 2012.

[22] Sascha Konrad and Betty HC Cheng. Real-time Specification Patterns. In *Proceedings of the 27th international conference on Software engineering*, pages 372–381. ACM, 2005.

[23] Sascha Konrad and Betty HC Cheng. Real-time specification patterns. In *Proceedings of the 27th international conference on Software engineering*, pages 372–381. ACM, 2005.

[24] A. Legay and L.M. Traonouez. Statistical Model Checking of Simulink Models with Plasma Lab. In *FTSCS'15*, pages 259–264. Springer, 2015.

[25] Florian Leitner. Technical report evaluation of the matlab simulink design verifier versus the model checker spin. 2008.

[26] K. Manamcheri, S. Mitra, S. Bak, and M Caccamo. A Step Towards Verification and Synthesis From Simulink/Stateflow Models. In *HSCC'11*, pages 317–318. ACM, 2011.

[27] Alistair Mavin and Philip Wilkinson. Big Ears (The Return of "Easy Approach to Requirements Engineering"). In *2010 18th IEEE International Requirements Engineering Conference*, pages 277–282. IEEE, sep 2010.

[28] Alistair Mavin, Philip Wilkinson, Adrian Harwood, and Mark Novak. Easy Approach to Requirements Syntax (EARS). In *2009 17th IEEE International Requirements Engineering Conference*, pages 317–322. IEEE, aug 2009.

[29] B Meenakshi, A. Bhatnagar, and S. Roy. Tool for Translating Simulink Models into Input Language of a Model Checker. In *ICFEM*, pages 606–620. Springer, 2006.

[30] Steven P. Miller. Bridging the Gap Between Model-Based Development and Model Checking. In *TACAS*, pages 443–453. Springer, 2009.

[31] Amalinda Post and Jochen Hoenicke. Formalization and analysis of real-time requirements: a feasibility study at bosch. In *Verified Software: Theories, Tools, Experiments*, pages 225–240. Springer, 2012.

[32] Amalinda Post, Jochen Hoenicke, and Andreas Podelski. rt-inconsistency: a new property for real-time requirements. In *International Conference on Fundamental Approaches to Software Engineering*, pages 34–49. Springer, 2011.

[33] Amalinda Post, Igor Menzel, and Andreas Podelski. Applying Restricted English Grammar on Automotive Requirements: Does It Work? A Case Study. In *Requirements Engineering: Foundation for Software Quality*, pages 166–180. Springer, 2011.

[34] Adwait Ratnaparkhi. *Maximum entropy models for natural language ambiguity resolution*. PhD thesis, University of Pennsylvania, 1998.

[35] Marc O Rauw. A simulink environment for flight dynamics and control analysis-application to the dhc-2'beaver'. *Delft University of Technology*, pages 121–138, 1993.

[36] Koushik Sen, Mahesh Viswanathan, and Gul Agha. *On Statistical Model Checking of Stochastic Systems*, pages 266–280. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[37] Thomas Wasow, Amy Perfors, and David Beaver. The puzzle of ambiguity. *Morphology and the web of grammar: Essays in memory of Steven G. Lapointe*, pages 265–282, 2005.

[38] Jonas Westman, Guillermo Rodriguez-Navas, and Mattias Nyberg. Verification of requirements in simulink design verifier and uppaal-an industrial case study, 2016.

[39] Oliver Wyman. Car innovation 2015: A comprehensive study on innovation in the automotive industry. *Oliver Wyman Report*, 2007.