DOCTORAL THESIS PROPOSAL

Design of Assured and Efficient Safety-critical Embedded Systems

Nesredin Mahmud
School of Innovation, Design and Engineering (IDT)
Mälardalen University
nesredin.mahmud@mdh.se
October 2018

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

Abstract

Safety-critical embedded systems should be analyzed rigorously in order to assure their functional correctness, moreover, they should be analyzed at the system design level, that is before implementation and testing for effectiveness and efficiency. Thus, safety-critical requirements specifications should be unambiguous, comprehensible, consistent, etc. Furthermore, the software design should conform to the specifications, hence satisfying the functional, and extra-functional requirements such as the timing constraints.

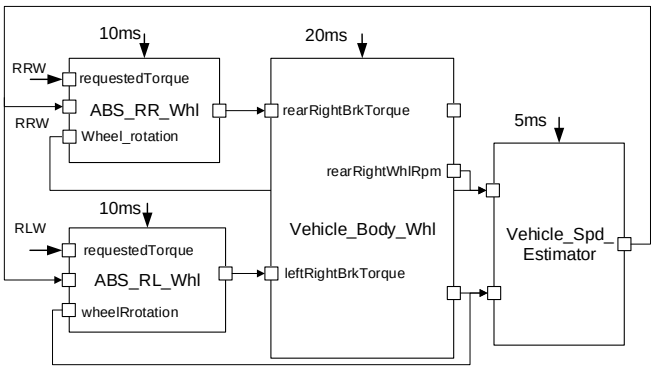
In distributed computing, e.g., electical/electronic vehicular execution platform of automotive systems, the software functionaliy is partitioned on multiple processors (or computing units), in which case, analyzing the timing constraints is not trivial due syncronization challenges, moreover, conserving critical system resources such as power and energy becomes crucial in order to ensure extensibility of the software. However, due to higher risks of system failure in distributing computing, maximizing reliability is desirable such as by applying fault tolerance which requires additional resources including higher power consumption.

In this thesis, we propose formal techniques and optimization methods respectively to assure improved quality of requirements specifications and software design, and efficient partitioning of software functionality. We propose a formal requiremets specification language ReSA, which is domain-specific and constrained natural language. The software design is modeled in Simulink, which is the de facto embedded software development environment in industry, and we propose a transformation of Simulink models into a network of stochastic timed automata via timed-automata patterns. The partitioning optimizes the power consumption of the distributed system while preserving the timing and reliability constraints of the safety-critical software, which is modeled in AUTOSAR, via the integer-linear programming and meta-heuristic methods.

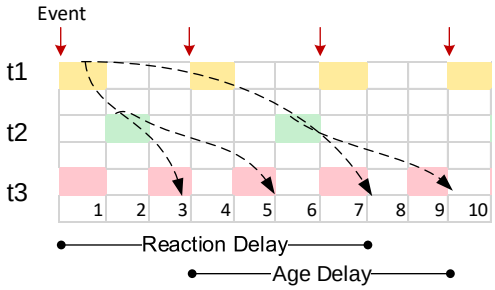
Our proposed domain-specific language and the statistical model checking of Simulink models are evaluated on the automotive use cases from Volvo Group Trucks Technology (VGTT), which consist of the adjustable-speed limiter (ASL) and the brake-by-wire (BBW) systems. The proposed optimization methods are evaluated on automotive benchmark from Bosch.

Introduction

An embedded system is some combination of computer hardware and software, either fixed in capability or programmable, placed within a larger system. Industrial machines, automobiles, medical equipment, airplanes, vending machines, as well as mobile devices are all possible locations for an embedded system [59]. In contrast to traditional software systems, e.g., web applications, desktop applications, embedded systems are usually safety critical, which means their failure to operate properly could cause fatalities and immense properties damage. Moreover, they operate in a rogue environment without interruption for a long time, therefore such systems should be dependable in many ways, for instance, they must be timely and reliable. The hardware platforms on which embedded software programs execute are resource constrained, therefore, efficient software partitioning and deployment is crucial especially in the midst of increasing software applications and their increasing functional complexity. In fact, the development of safety-critical embedded systems is driven by stringent functional safety requirements from respective industry standards, e.g., ISO 26262 in automotive systems [30], MIL-STD-2167/DO-178C in avionics [60], IEC 61511 in process engineering [7], etc, in order to ensure the quality of embedded systems, which is required to be enforced at different levels, including system development, maintenance, and operation. For instance, the ISO 26262 standard for automotive systems highly recommends rigorous methods to specify and analyze the systems, by applying *formal methods* at different levels of system development, such as requirements specification, system design and implementation. Formal methods [51] are mathematical techniques that can help the specification and analysis of system models, with precision and without ambiguity. In particular, in this thesis, we apply *lightweight formal methods* [2,31], which tackle the difficulty of applying formal methods by applying partial specification, modeling, and focused analysis of relevant system parts, thereby facilitating their applicability in practice.



(a) Brake-By-Wire Block Diagram.



(b) Timed Paths for the Chain: ABS_RR_Whl (t1)→ Vehile_Body_Whl (t2)→ Vehile_Spd_Estimator (t3).

Figure 1: Multirate System, Brake-By-Wire Example.

The difficulty of satisfying the timing requirements of embedded systems increases especially when embedded software applications are designed to run over different sampling times (or activation patterns), also known as *multirate*. A multriate design approach enables multitasking by allowing the tasks that realize the functionalities of software applications to execute independently (that is on their own sample time) through interleaving. The timing analysis of multirate systems is complex

due the various timed paths that are generated as signals propagate from the sourcing tasks to the sinking tasks. The timed paths are interpreted according to various delay semantics of which the most common are age delay and reaction delay. [5,19,47]. Multirate designs are widely observed in embedded systems, including in the automotive domain.

Assuming a Brake-by-Wire (BBW) system that controls the brakes of a car through electrical means, in Figure 1a we show a software application model for the specific part of the BBW system in which the rear-wheel rotation speed and the vehicle speed are used to estimate the target speed, by considering several state and environmental parameters. The *ABS_RR_Whl* and *ABS_RL_Whl* components are triggered every 10ms to compute the rotational speed of the wheels, which are fed to the *Vehicle_Body_Whl* component that runs every 20ms. The *Vehicle_Spd_Estimation* component runs every 5ms to compute the target vehicle speed by reading inputs from the previous component and other parameters not shown in the model. Figure 1a shows the timed paths of the age delay (the latest time the signal propagates from input to output) and reaction delay (the earliest time the signal propagates from input to output) of the model. In this thesis, we focus on the design and analysis of multirate embedded systems.

Embedded systems are usually developed over several development stages as illustrated by the V-model in Figure 2. In the particular case of top-down software development, the requirements specifications are employed in the design of high-level system architecture, as well as in designing the behavior of the software system according to the high-level architecture. In this model of the software development process, the detection and elimination of software errors at early stages is crucial in order to prevent their propagation to the implementation stage. By doing so, the maintenance cost and the time-to-market can be reduced [16, 26]. Therefore, in this thesis, propose scalable and usable ways of applying formal techniques at the requirements specification, system design and software design stages of embedded systems development, in order to assure and thereby improve the quality of embedded systems.

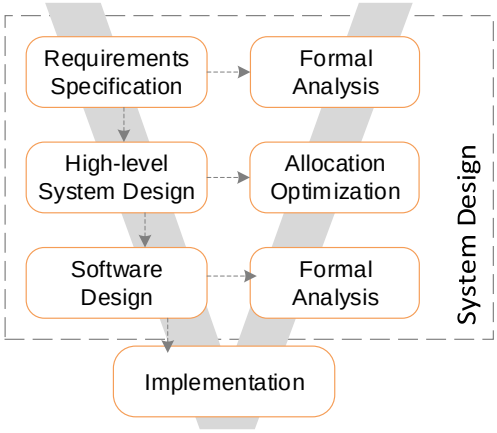


Figure 2: Top-down Development, in V-model.

Requirements specifications are structured collections of functional and extra-functional requirements, which are statements that describe the needs that the solution must satisfy [1]. The specifications are mainly textual but could also contain diagrams to elaborate the textual representations. The textual representations of requirements are mostly expressed in natural language mainly because natural language is intuitive. Furthermore, natural language is expressive and flexible, so the required system functionality can be sufficiently expressed and communicated easily. However, natural language can be deemed too flexible and unconstrained to the extent that engineers tend to specify requirements that are hard to understand, vague, ambiguous, etc. Template-based methods [29], controlled natural language [24,55] and property specification pattern systems with underlaying formal semantics [23,27,34,52] have been proposed to mitigate most of the natural language drawbacks and misuse. However, due to the lack of effective and scalable methods of specification, the above mentioned problems of requirements specifications persist in practice [57]. To address this deficit, we propose a flexible yet structured language, called ReSA [ref], for specifying embedded systems requirements, based on boilerplates and domain-specific concepts. To provide automated support, we also propose the ReSA tool [ref] that also implements a consistency checking method for system requirements, based on the satisfiability of the Boolean formulas that encode

the high-level requirements

At the system design level, the requirements specifications are realized by a system architecture of software and hardware parts that are constructed by functional components (or modules) that abstract the functionality of the system. In the platform-based development approach [54], the system design should take into account consumption of critical hardware resources, such as power consumption, for two main reasons: optimizing power consumption is beneficial in order i) to accommodate more applications as well as to support power-intensive applications, and ii) to increase battery-life by lowering the amount of heat released by electronic components in the system. In this thesis, we propose an exact software allocation approach, as well as a heuristic one, for multirate systems that need to meet both timing and reliability.

The system architecture is refined further by software design units, whose behavior is often specified in Simulink [32], which is the de-facto modeling language employed in industry. To provide assurance that Simulink models fulfill given functional and timing requirements, 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 with UPPAAL Statistical Model Checker [9]. Our method is scalable, and has been validated on industrial use cases [21]. The statistical model checking 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 [38].

The rest of the thesis proposal is organized as follows. Section 2 introduces the research goals and the scientific contributions to address the research goals. Section 3 presents the research methods applied to conduct the research especially in the context of academic-industry collaboration. Section 4 shows the proposed outline of the thesis, followed by the presentation of the research progress in Section 5, including the time plan until the doctoral defense. Section 6 presents the third-cycle outcomes, adapted from the individual study plan (ISP). Finally, Section 7 discusses the related work before concluding the proposal in Section 8.

Research Description And Thesis Contributions

The research is conducted within the VeriSpec project[1], in collaboration with the industrial partners of the project, that is, Volvo Group Trucks Technology (VGTT) and Scania. Consequently, the thesis deals with several issues that these companies have encountered in the development of automotive systems. The problems can be briefly summarized as follows: i) many software developers have expressed their dissatisfaction of using natural language to express requirements, mainly because it is too unconstrained, which may lead to vague, ambiguous and inconsistent specifications. Nevertheless, natural language is still appealing to many because it is intuitive, expressive and flexible; ii) architectural description languages such as AUTOSAR and EAST-ADL as well as the behavioral modeling environment Matlab/Simulink are widely used in companies to model and simulate automotive systems before code generation and deployment. However, there is little analysis support in the languages, and tools that are rigorous and scalable, e.g., for behavioral and timing analysis; iii) lack of (semi) automated bridging of specifications, designs, and implementations has lead to discrepancies (or inconsistencies) between the different views (levels of abstraction) of the system.

The above are the main industrial problems that have laid the grounds for the research in the

[1] http://www.es.mdh.se/projects/343-VeriSpec___Structured_Specification_and_Automated_Verification_for_Automotive_Fu

Ph.D. thesis. The full remedy to these problems remain desiderata not just in these companies but also in similar embedded systems industries [8, 26, 36, 37, 45]. In light of the problems discussed above, the over goal of the thesis is to:

Overall Goal: Provide assurance of functionality and quality of embedded software systems, at various levels of abstraction, via formal analysis and optimization of critical system resources.

Research Goals

The overall research goal is too broad to be addressed as such, therefore we refine it into five research goals. The research goals are stated after discussing their rationale. Subsequently, they are followed by discussing the challenges to address the goals. The first research goal deals with the issues of requirement specification.

Natural language is still the de-facto language for specifying requirements of embedded systems, albeit inherently ambiguous and too unconstrained, essentially enabling engineers to misuse it. Moreover, natural language specifications do not have formal semantics (or cannot be interpreted by computer programs), and therefore, cannot be rigorously analyzed. Due to the lack of a shared domain-specific knowledge-base, the problem of natural language specifications is further exacerbated by inconsistencies at the lexicon level or logical inconsistencies. Therefore, the first research goal is to:

Research Goal 1: Improve the comprehensibility and flexibility of natural language requirements specifications while providing analysis in the absence of system models.

One of the mechanisms to improve natural language specifications is by constraining the language, including its syntax, semantics and the lexicon [35]. The design of a constrained natural language for the specification of requirements is not trivial mainly because: i) natural language possesses conflicting properties (or metrics) [1], for instance by constraining natural language, its expressiveness and flexibility [48] can be impaired. Furthermore, it could lose its intuitiveness. Therefore, appropriate trade offs should be made in the design in order to have a robust and effective specification language; ii) in order to achieve a usable constrained natural language, the latter should be tailored to a specific domain of application, which requires expertise in the field; iii) usually requirements are specified before the system architecture is developed, so the constrained natural language should provide support to analyze requirements in the absence of architectural models.

In addition to assuring the quality of requirements specifications individually, the latter should be should be analyzed in ensemble in order to detect errors that span multiple specifications, e.g., logical inconsistencies within specifications. In essence, the specifications should possess rich semantics that relate individual specifications to each other. In this regard, the second research goal is to:

Research Goal 2: Facilitate rigorous analysis of embedded systems requirements specifications.

Natural language requirements 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, mainly due to the multitude of interpretation (or semantics) paradigms [11]. Model-theoretic semantics is

a widely-applied paradigm in computational semantics which computes truth values of sentences by inductively applying interpretation functions on the syntactic units (or structures) of the sentences in relation to mathematical systems, e.g., propositional, first-order systems. The depth of the interpretation greatly affects the use and applicability of the interpretation approach. For instance, with the assumption that a proposition equates to a clause in a sentence, propositional logic elegantly represents sentences and scales well to find truth values, albeit it provides a shallow interpretation of the sentences. In contrast, first-order logic representations provide rich semantics, and thus enable rigorous yet less tractable analysis. Therefore, the type of analysis and level of rigor needed drives the type of semantics.

Addressing research goals 1 & 2 assures that the quality of requirements specifications is improved before the latter's use in subsequent development phases of design and implementation, e.g., by removing syntactic and type errors, resolving logical contradictions and disambiguation, as well as by structuring the specifications such that they become comprehensible.

Afterwards, the specifications are used to create the high-level system design (or architecture) that realizes the required functionality. The system architecture consists of software and hardware logical components, as well as mappings from the software components to the hardware components (or computation nodes). The latter activity of the system design, also known as *software allocation*, should be handled effectively in order to preserve the functionality of the software architecture, including functional correctness and extra-functional attributes such as timing and reliability. Furthermore, the allocation should be efficient in order to conserve critical system resources while satisfying requirements as well as design and hardware constraints.

Effective and efficient allocation of multirate software applications is highly needed, as it is crucial to safety-critical systems such as automotive systems, aircrafts, etc. Many multirate applications are deployed on several computation nodes, which are possibly heterogeneous, over shared communication networks. Therefore, the third research goal is as follows:

Research Goal 3: Find an effective and efficient allocation of multirate software applications on a network of heterogeneous computing nodes, with respect to critical system resources.

Allocation of multirate software applications on a network of computation nodes is challenging mainly due to the complex timing analysis that needs to be considered during the allocation process [41,47]. The timed paths of multirate applications increasing the search space, and therefore finding an efficient allocation becomes exponential. Furthermore, the heterogeneity of the computation nodes forces the search method to consider every computation node for better result, thus increasing the optimization time, as opposed to considering homogeneous nodes. Since the software allocation could be intractable especially for large systems, methods based on heuristics and approximation should be provided instead of exact ones.

Addressing research goal 3 ensures that the software applications are deployed with minimal resource utilization at the same time assuring the satisfaction of extra-functional requirements such as timing and reliability. Furthermore, it ensures that the design and hardware constraints are met.

The system architecture, in particular the software architecture is further refined via software unit designs that capture the behavior of the software. We assume that the latter is captured by models in Simulink [32], which is one of the most popular and robust component-based software development environment employed in industry.

Simulink is used in the design, analysis, simulation and code-generation of embedded system. Since it is quite usual that code is generated from Simulink models, it is beneficial to analyze the latter before code generation in order to prevent potential high-level errors to propagate, hence

improve the quality of implementations. Traditional Simulink analysis techniques, e.g., by type checking, and simulation, are not sufficient to guarantee the full correctness of safety-critical design models. Rather, rigorous methods should be applied to find defects that are hard to find, such as division by zero, dead logic, buffer overflow, etc., which are supported by Simulink Design Verifier (SDV)[2]. Nevertheless, SDV is limited in its capability, especially when it comes to proving timed properties and it does not scale well for large models verification [39]. Therefore, the fourth research goal in this thesis is formulated below:

Research Goal 4: Enable scalable formal analysis of multirate software applications modeled in Simulink.

Simulink consists of connected and hierarchical Simulink blocks, which encode mathematical functions. 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 order to show the validity of our proposed solutions, a working prototype should be developed and should be applied on industrial uses cases. Our proposed specification and analysis methods and tools should be engineering friendly in order to facilitate their adoption in industry, for instance they should embody intuitiveness and seamless integration into industrial practices. Therefore, the last research goal is as follows:

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

Seamless integration of new development methods and tools require appropriate interfaces to plug into existing development methods and tools in order to facilitate adoption of formal techniques, by lowering the required effort and cost. In particular, formal methods should be accompanied by engineering-friendly interfaces, as the syntax of the formal notations is not familiar to most engineers, likewise, understanding the underlying semantics requires a substantial shift from traditional software development paradigms. In order to tackle these challenges, very close cooperation with engineers and know-how of domain-specific industrial tools and practices are paramount.

Papers Included In The Thesis

The papers to be included in the PhD thesis are listed as follows.

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

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 *10th IEEE International Symposium on Industrial Embedded Systems (SIES), 2015, IEEE*

My Contribution: I was the main driver of the paper. I developed the ReSA language including its syntax and semantics, and Cristina Seceleanu proposed a consistency analysis technique besides giving useful comments and ideas on the design of the language. Oscar Ljungkrantz provided useful materials from VGTT that were eventually analyzed for the language development, and gave feedback on the language design and implementation from an industrial viewpoint.

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 *Federated Conference on Computer Science and Information Systems (FedCSIS), 2016, IEEE*

My Contribution: I was the main driver of the paper. I developed the ReSA toolchain that consists of the editor and the consistency checker including the integration with the Z3 SAT solver in the backend. Cristina Seceleanu formulated the consistency checking and together with Oscar Ljungkrantz, they contributed to the paper with useful comments and ideas.

Paper C

Specification and Semantic Analysis of Embedded Systems Requirements: From Description Logic to Temporal Logic

Nesredin Mahmud, Cristina Seceleanu, Ljungkrantz Oscar

Abstract: *Due to the increasing complexity of embedded systems, early detection of software/hardware errors has become desirable. In this context, effective yet flexible specification methods that support rigorous analysis of embedded systems requirements are needed. Current specification methods such as pattern-based, boilerplates normally lack meta-models for extensibility and flexibility. In contrast, formal specification languages, like temporal logic, Z, etc., enable rigorous analysis, however, they usually are too mathematical and difficult to comprehend by average software engineers. In this paper, we propose a specification representation of requirements, which considers thematic roles and domain knowledge, enabling deep semantic analysis. The specification is complemented by our constrained natural language specification framework, ReSA, which acts as the interface to the representation. The representation that we propose is encoded in description logic, which is a decidable and computationally-tractable ontology language. By employing the ontology reasoner, Hermit, we check for consistency and completeness of requirements. Moreover, we propose an automatic transformation of the ontology-based specifications into Timed Computation Tree Logic formulas, to be used further in model checking embedded systems.*

Status: *Published in 15th International Conference Software Engineering and Formal Methods (SEFM), 2017, LNCS Springer.*

My Contribution: I was the main driver of the language. I developed the ReSA language semantics using event-base approach, which is encoded in description logic. Cristina Seceleanu and Ljungkrantz Oscar provided with useful ideas and comments.

Paper D

Scalable Allocation of Fault-tolerant Multi-rate AUTOSAR Applications
Nesredin Mahmud, Guillermo Rodriguez-Navas, Hamid Faragardi,Saad Mubeen, Cristina Seceleanu

Abstract: *Software-to-hardware allocation plays an important role in the development of resource-constrained automotive embedded systems that are required to meet timing, reliability and power requirements. This paper proposes an Integer Linear Programming optimization approach for the allocation of fault-tolerant embedded software applications that are developed using the AUTOSAR standard. The allocation takes into account the timing and reliability requirements of the multi-rate cause-effect chains in these applications and the heterogeneity of their execution platforms. The optimization objective is to minimize the total power consumption of the applications that are distributed over more than one computing unit. The proposed approach is evaluated using a range of different software applications from the automotive domain, which are generated using the real world automotive benchmark. The evaluation results indicate that the proposed allocation approach is effective and scalable while meeting the timing, reliability and power requirements in small- and medium-sized automotive software applications.*
Status: To be submitted to *Journal of Systems and Software (JSS), Elsevier.*
My Contribution: I am the main driver of the paper. I developed the system model (including the power consumption, timing, reliability models) and further refined by the co-authors. Hamid Faragardi and I developed the ILP model, and I implemented the ILP problem (including the system model) and collected experimental results. The co-authors gave useful ideas and comments on respected parts of the paper: Guillermo Rodriguez-Navas on reliability modeling, Hamid Faragardi on optimization and related work, Saad Mubeen on the timing analysis, and Cristina Seceleanu gave comments and ideas on the main contributions of the paper, including on the optimization objective and constraints.
In Case of Rejection: Power-aware Allocation of Fault-tolerant Multi-rate AUTOSAR Applications conference paper
Status: To appear in Proc. of the *25th Asia-Pacific Software Engineering Conference (APSEC), 2018, Japan*, IEEE CS.
Note: The JSS paper is an extension of the APSEC paper.

Paper E

SIMPPAAL - A Framework For Statistical Model Checking of Industrial Simulink Models

Predrag Filipovikj, Nesredin Mahmud, Raluca Marinescu, Cristina Seceleanu, Oscar Ljungkrantz, Henrik Lönn

Abstract: *The evolution of automotive systems has been rapid. Nowadays, electronic brains control dozens of functions in vehicles, like braking, cruising, etc. Model-based design approaches, in environments such as MATLAB Simulink, seem to help in addressing the ever-increasing need to enhance quality, and manage complexity, by supporting functional design from a set of block libraries, which can be simulated and analyzed for hidden errors, but also used for code generation. For this reason, providing assurance that Simulink models fulfill given functional and timing requirements is desirable. In this paper, we propose a pattern-based, execution-order preserving automatic transformation of atomic and composite Simulink blocks into stochastic timed automata that can then be formally analyzed with Uppaal Statistical Model Checker (Uppaal SMC). To enable this, we first define the formal syntax and semantics of Simulink blocks and their composition, and show that the transformation is provably correct for a certain class of Simulink models. Our method is supported by the SIMPPAAL tool, which we introduce and apply on two industrial Simulink models, a prototype called the Brake-by-Wire and an operational Adjustable Speed Limiter system. This work enables the formal analysis of industrial Simulink models, by automatically generating stochastic timed automata counterparts.*

Status: To be submitted (by 15th October 2018) to the ACM Transactions on Software Engineering and Methodology (TOSEM) Journal, ACM.

My Contribution: The 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 introduced a mechanism to enforce the execution order of the blocks using inter-arrival times. Predrag implemented the flattening algorithm and 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 defining the methodology, and with useful ideas and comments. Guillermo Rodriguez-Navas wrote the related work section. The industrial coauthors provided the use cases and commented on the final draft.

In Case of Rejection: It will be included as a technical report [22].

Paper F

SIMPPAAL meets ReSA: From Automated Requirements Specifications to Automated Formal Analysis of Simulink Models.
[authors part]

Abstract: *The main objective of this work is to extend the validation of our contributions, that is the SIMPPAAL method and tool [22] for the verification of Simulink models based on SMC, for which the properties will be automatically generated by employing the structured requirements specification framework called ReSA [40]. The verification will be applied on a set of Simulink models that are part of the Adjustable Speed Limiter system, which is an operational system installed in all Volvo Trucks.*
Status: Candidate venue for submission: *"NFM 2019: 11th Annual NASA Formal Methods Symposium"* Submission deadline: 2018-12-14. Notification Date: 2019-02-22.
In Case of Rejection: SEKE 2019: International Conference on Software Engineering and Knowledge Engineering. Submission Deadline: 2019-03-01. Notification date: 2019-04-20.

Thesis Contributions

The thesis contributions are research results that address the research goals presented in Sect2 and are included in the listed publications, or "in progress" articles. The research contributions are illustrated in Figure 3, which is aligned with the top-down software development approach. We propose a requirements specification language based on domain knowledge in embedded systems (Contr.#1), and we define its semantics in Boolean and description logic (Contr.#2), followed by consistency checking of requirements specifications (Contr.#3). Furthermore, we propose an allocation scheme for multirate software applications with the objective of optimizing power consumption while fulfilling the timing and reliability requirements (Contr.#5). Next, we also propose a method for transforming Simulink models into stochastic timed automata and the latter's statistical model checking (Contr.#6).

Last but not least we show how to transform structured requirements into temporal logics for exhaustive and statistical verification of models, and integrate the developed tool support for requirements specification and analysis, and model checking of Simulink models into the VeriSpec framework.

Contribution 1 - The ReSA Language

ReSA [42] is a constrained natural language. Therefore, it contains syntactic and semantic rules that limit requirements specifications, which are introduced to improve quality of specifications. It has almost similar syntax and semantics to the English language except for the parentheses and type-setting constructs. The language uses concepts from embedded systems, e.g., System, Parameter, Device, Mode, State, etc., to typeset nouns and noun phrases, and also axiomatic relations between the concept instances to limit semantically implausible specifications, thus effectively making the ReSA language domain specific. The valid strings of the ReSA language are sentential forms (or *boilerplates*) and comprise clauses and operators such as prepositions and conjunctives as shown in Table 1 and 2, respectively.

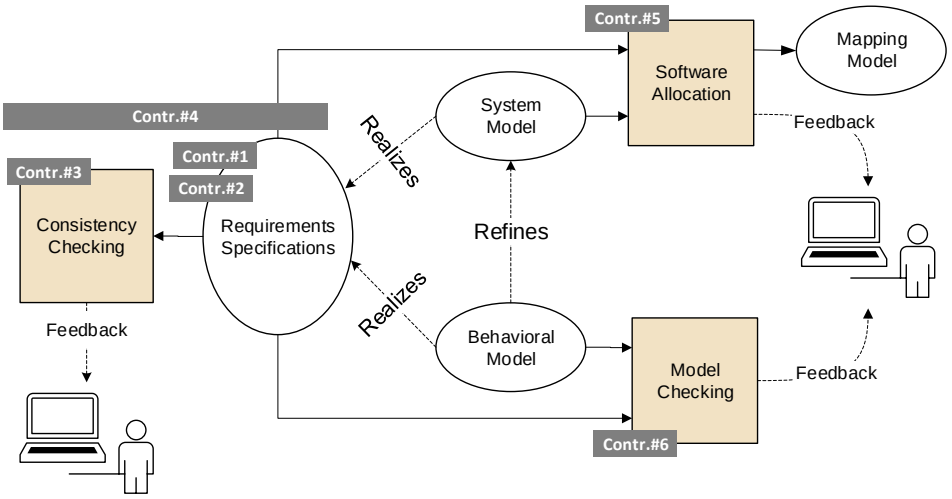


Figure 3: Research Contributions Overview.

Boilerplate Type	Example
Simple	<NP>V<NP>
Complex	if <Simple>, <Simple>
Compound	<Simple>and <Simple>

Table 1: Boilerplate Types

Operators	Examples
Coordinating Conjunctives	and, or
Subordinating Conjunctives	if, after, before, until, when, while
Prepositions	within, between, after,after,every

Table 2: Operators

A valid ReSA specification is shown below (left side). It specifies an ASL activation requirement using the boilerplates and operators. The text in right side is the specification after removing the tags (or in plain format).

Req =

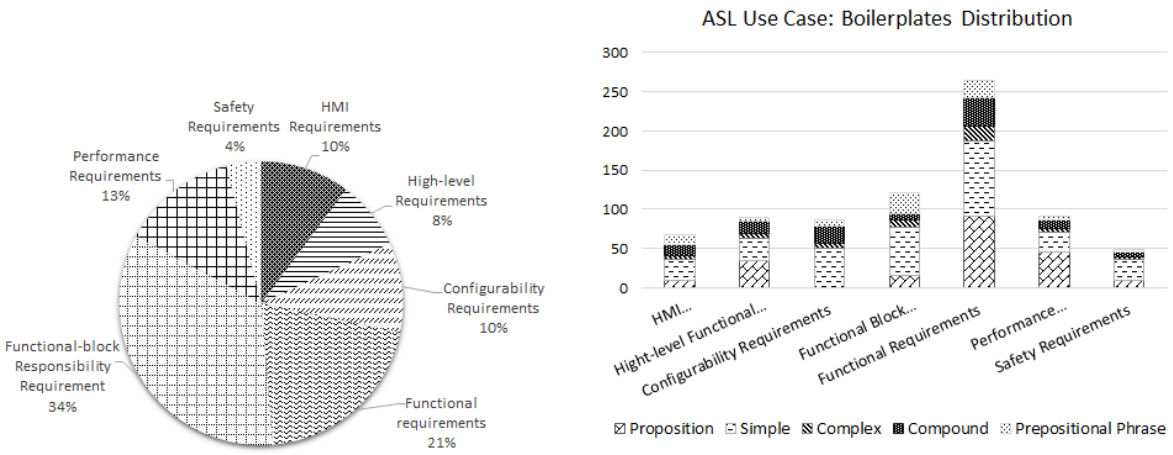
```
if "the vehicle":system is in running:mode and
    "the driver":user presses "the ASL button":indevice
then
   "the ASL":system shall be activated within 250ms;
endif
/*Original Format - with tags*/
```

Req = =

```
if the vehicle is in running mode and
    the driver presses the ASL button
then
    the ASL system shall be activated
endif
/*Plain Format - without tags*/
```

In a nutshell, the language's design and implementation has the following advantages: i) it improves readability by providing a simplified and structured English; ii) it facilitates reusability by allowing the user to instantiate templates (or requirements boilerplates), which can be saved as artifacts and be used later for specification; iii) disambiguates specifications by using parentheses, and iv) complex and compound specifications can be created inductively by combining clauses (or boilerplates) using the operators of the language.

We validate ReSA on the Adjustable Speed Limiter (ASL) use case. The ASL system is a vehicle speed control that limits the vehicle to not exceed a predefined speed limit, set by the driver. It is an automotive feature in Volvo trucks. The system comprises around 300 requirements, which consist simple and conditional statements that are categorized into different types of requirements as shown in Figure 4a. After rewriting the requirements in ReSA, the distribution of the clauses (or boilerplates) that are applied to specify the ASL requirements is summarized in Figure 4b. The validation result shows that about 90% of the textual ASL requirements can be expressed in ReSA.



(a) Requirements Types (or Categories). (b) Boilerplates Distribution.

Figure 4: Validation on Adjustable Speed Limiter Requirements.

However, 10% of the requirements contain quantifiers and timed properties, which could not be specified effectively in ReSA.

Contribution 2 - Formal Semantics Of The ReSA Language

In order to perform complex analysis such as consistency checking, specifications should have rich annotations that relate not only words within clauses themselves. To support such functionality, we propose a formal interpretation of the ReSA specifications in Boolean and description logics.

Semantics in Boolean logic: ReSA specifications are translated via three simple transformation axioms: i) normalizing rule Clause* $\leftrightarrow$ Clause - removes resolves negation, ii) affirmative rule Clause $\leftrightarrow p$, and iii) negative rule Clause $\leftrightarrow \neg p$. After translating ReSA specifications into Boolean expressions, they can be checked for satisfiability using SAT/SMT solvers in order to detect logical inconsistencies in the ReSA specifications [40,42]. Although SAT problems are known to be NP-hard, there exist known heuristics and approximation algorithms that handle problems with thousands of propositional variables. This enables checking large requirements specifications in reasonable time using existing solvers, such as the Z3 tool [13].

The following expression denotes the specification **Req** in Boolean formula.

$$\begin{aligned}\textbf{[Req]} = \quad & (p1 \wedge p2) \mapsto p3; \text{ where,} \\ & p1 = \text{“<vehicle> is in <"running"> mode;”,} \\ & p2 = \text{“<driver> <presses> <ASLButton>;”,} \\ & p3 = \text{“<ASL> shall be <activated>; within <250><ms>;”}\end{aligned}$$

Since propositional variables abstract a substantial amount of information by abstracting clauses, the analysis can be considered shallow, since advanced equivalence relations cannot be resolved between clauses. To circumvent this problem, we introduced a linguistic technique to represent the clauses in a richer format. The underlying formal representation is encoded in description logic [3] and uses the concept of *ontology* to capture the requirements specifications as a knowledge-base, enabling advanced analysis and querying (or inferences about specifications).

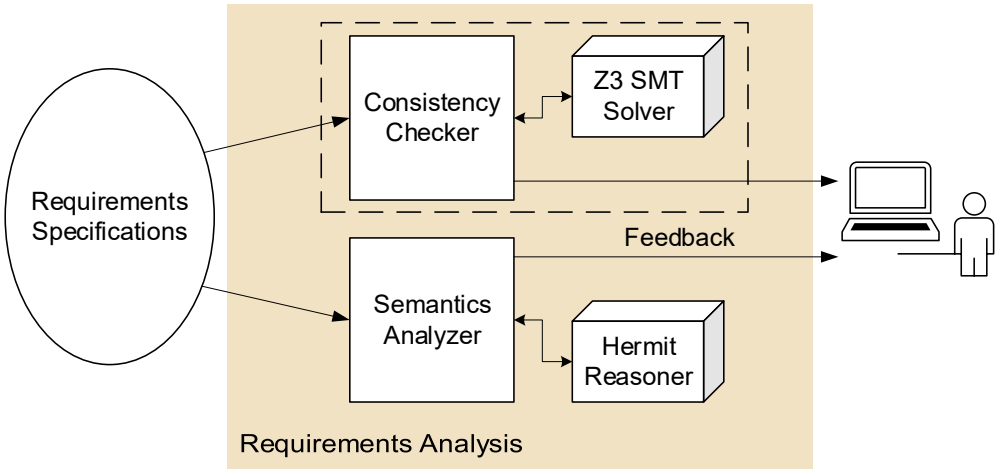


Figure 5: Architecture of ReSA Framework.

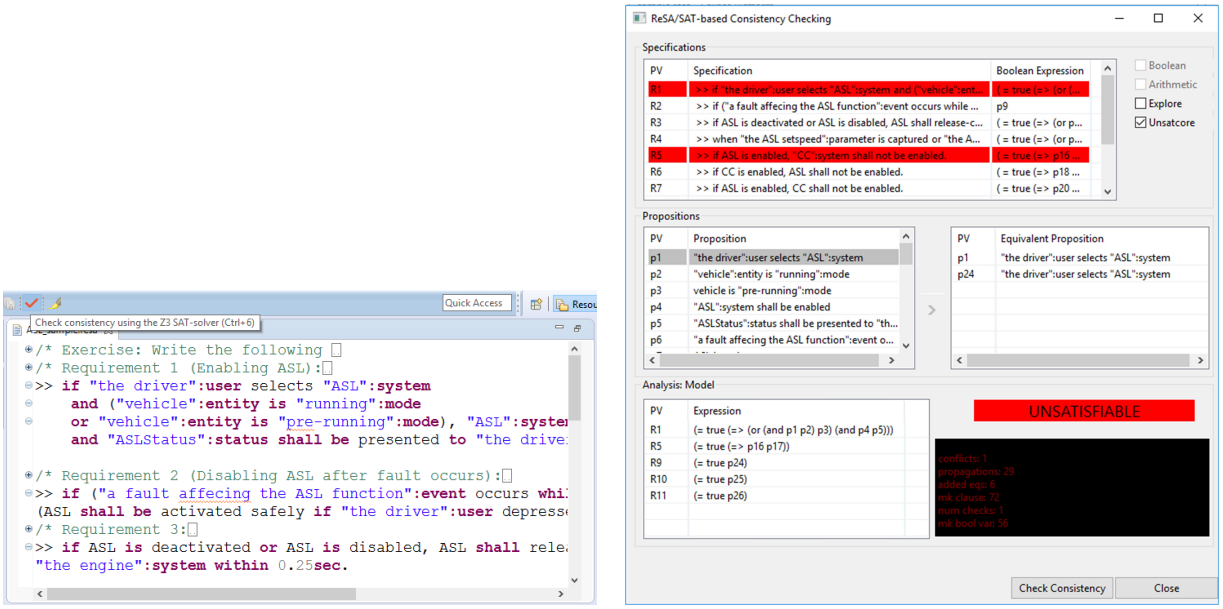
Semantics in description logic: Description logic [3] was initially proposed to describe and reason about semantic networks. It was designed to contain the decidable elements of first-order logic, as such it has gained popularity for practical use in several areas, including web semantics, biomedical informatics, and artificial intelligence. Description logic is also the underlying knowledge representation semantics of the popular web ontology language OWL [44].

Furthermore, description logic is used in linguistics for the purpose of Natural Language Processing (NLP) to represent sentences, not only of their syntactic structures but also their lexical contexts too, for instance, in order to disambiguate sentences. We develop this concept into a representation/description of requirements specifications following an event-based semantics approach [43]. The approach considers an event as a handler that represents a state or an action. The embedded system concept instances are annotated with thematic roles, which define the role of arguments (nouns and noun phrases) for specific predicates, e.g., the Agent role, the Instrument role etc [43]. The annotation enables reasoning about specifications at the lexical level, hence relating clauses semantically, discarding specifications that are semantically not sound, etc.

Contribution 3 - Consistency Checking Of Requirements Specifications

Once requirements specifications are checked individually, they should be analyzed together to detect errors, such as inconsistencies of specifications, incompleteness, etc. Depending on the rigor needed to analyze the specifications, we propose two methods of consistency checking as illustrated in Figure 5: SAT-based and ontology-based methods, respectively. In the former approach, the problem of checking consistency is formulated as a Boolean Satisfiability problem (SAT) [15], which is defined as follows: given a propositional formula $\phi = \mathrm{f}(x_1, \ldots, x_n)$, over a set of Boolean variables $x_1, \ldots, x_n$, decide whether or not there exists a truth assignment to the variables such that ϕ evaluates to true. SAT problem instances are usually expressed in a standard form called *conjunctive normal form* (CNF). A propositional logic formula is said to be in CNF if it is a conjunction (*and*) of disjunctions (*or*) of literals. A literal is either x, or its negation $\neg x$, for a Boolean variable x.

Definition 1 (Inconsistency of Requirements Specifications). *Let $\Psi = \{\psi_1, ..., \psi_n\}$ be a set of Boolean formulas that denotes a set of requirements specifications, where each formulas $(\psi_1, \ldots, \psi_n)$ encodes a requirement specification. We say that the set is inconsistent if $\psi_1 \wedge \psi_2 \wedge \ldots \psi_n \Rightarrow False$ holds, in other words if there do not exist any valuations (or assignments) of the Boolean variables*



(a) Graphical User Interface (GUI).

(b) SAT-based Consistency Checker.

Figure 6: ReSA Toolchain.

that satisfy Ψ.

In the ontology-based approach, the specifications are translated into a knowledge-base system according to the event-based semantics as encoded in description logic. The knowledge-base contains TBox and ABox parts, with the TBox concepts and axiomatic concepts asserted in TBox, and facts asserted in ABox. As an example, For example, the concepts *System*, *Device* are classified into Agents that are asserted in TBox, whereas *ASL:system*, *ASLButton:indevice*, *activate(e, ASL)*, etc., are facts that are in ABox. The consistency of the ontology is checked according to the following lemma:

Lemma 1. *The ontology is consistent if there exists an interpretation (or a model) that satisfies the TBox (T) and the ABox(A) assertions, i.e.,* $M \models ax$*, where:* $ax \in T \cup A$.

Contribution 4 - The ReSA Toolchain

The ReSA toolchain [40] is an integrated tool architecture for the specification and analysis of requirements specifications using the lightweight formal techniques discussed before. Figure 6 shows the Graphical User Interface (GUI) for the toolchain, which shows the editor and a toolbar button for triggering the consistency checker that is shown in Figure 6. The editor provides the user with rich-text editor, including content-completion, essentially guiding the user. The specifications are checked individually for syntax and type errors before analyzing them in ensemble to do complex analysis. The complex analysis is either shallow or deep analysis, respectively, and it is supported by SAT-based analysis and ontology-based analysis.

In the case of SAT-based analysis, the consistency checker translates the specifications into Boolean expressions, whose satisfiability is checked by the Z3 SMT solver. The consistency checker provides feedback to the user, which is: it returns “sat" if the expressions are *Satisfiable*, or “unsat" in the opposite case, in which case the inconsistent specifications (also known as unsat core) is also returned to the user. In case, the satisfiability of the expressions cannot be determined, the

solver returns "unknown". In the case of ontology-based analysis, the specifications are translated into a knowledge-base system according to the event-based semantics, encoded via the description logic. Afterwards, the ontology is reasoned about, consequently, its consistency, and completeness is proved via a reasoner program, in our case Hermit [56].

Unlike the ontology-based implementation, the SAT-based implementation is integrated into the EATOP integrated development environment (IDE). The IDE is an EAST-ADL[3] modeling environment including requirements modeling, where the ReSA toolchain has been integrated into. The ReSA editor is connected to the EAST-ADL model in order for the user to access the model elements and use them during specification. Such functionality is crucial especially for ensuring a consistent use of vocabularies in the development team.

Contribution 5 - Efficient Allocation Of Multirate Software Application Onto Network Of Computation Nodes

At the system design development stage, the software allocation process plays an important role in determining the quality of the software system. Effective and efficient allocation of safety-critical software applications must ensure that the functional and extra-functional requirements are satisfied after the allocation. Moreover, after the allocation, one should ensure that critical system resources are optimized. In this thesis, we consider power consumption as the critical system resource to be optimized, as it is often constrained in embedded systems. Moreover, by optimizing power consumption, future applications can be accommodated without additional power supply.

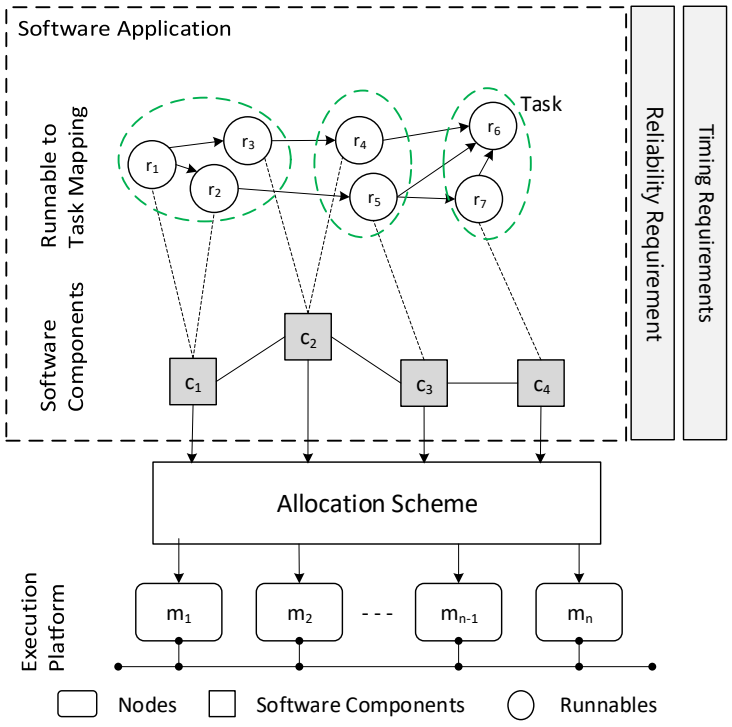


Figure 7: Allocation of AUTOSAR Software Application.

During the optimization, the timing and reliability requirements of the application should be guaranteed, as well as the design and hardware. To demonstrate our approach, we have used

[3]http://www.east-adl.info/

an AUTOSAR system, which is fault-tolerant and designed to run on a network of computation nodes [41]. The system model is described as follows:

System Model. The system model $\langle a, p, m \rangle$ is a tuple that includes a software application model a, a platform model p and a mapping scheme m as illustrated in Figure 7. The software application is modelled as a digraph of *Runnables*, which are AUTOSAR-schedulable entities (or programs) [49]. The runnables are grouped into tasks based on some heuristics, for instance runnables with the same period and located on the same host are grouped into a single task. The platform $\langle M, b \rangle$ consists of heterogeneous computation nodes M and a shared CAN bus b, whereas the mapping scheme $m : a \mapsto p$ is an allocation function that maps the application to the platform.

Allocation Problem. The allocation problem places optimization of power consumption as the main objective, and the timing and reliability requirements of the software applications as constraints. Depending on the nature of the application, the constraints can be hard or soft, and are user configurable. The cost function of the problem is shown in Equation 1.

$$\min_{x \in X} P(x) \tag{1}$$

Subjected to:

$$ResponseTime(x) \leq Deadline \tag{2}$$

$$Delay(x) \leq E2eReq \tag{3}$$

$$Reliability(x) \leq RelReq \tag{4}$$

where, the *ResponseTime()*, *Delay()*, and *Reliability()* functions compute the response time, end-to-end delay and reliability of the software application, respectively.

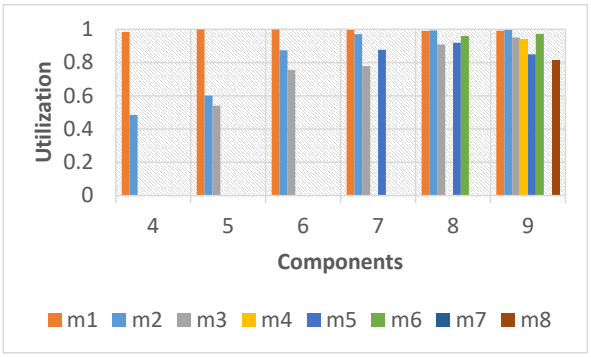
In this thesis, we propose two methods to solve the problem: an exact method using Integer Linear Programming (ILP) and a meta-heuristic method using an evolutionary algorithm. The methods have their own pros and cons in the context of solving the software allocation problem as discussed in their subsequent topics.

Exact Method. We have used an exact method to find the optimal solution, if there exists, by employing ILP, for relatively small problems, the solutions are returned almost instantaneously [41]. Figure 8a shows allocation of different sizes of software applications (indicated by the different number of components) on a network of computation nodes that vary from 4 to 9, and Figure 8b shows the optimal power consumption returned by the ILP solver for the different applications.

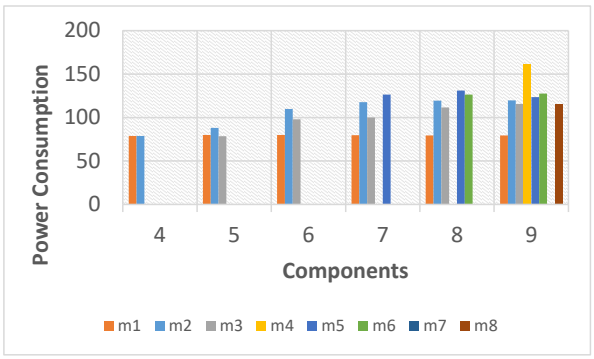
Meta-heuristics Method. The ILP approach is tractable to small and medium problems, that is, to a relatively few number of software components, cause-effect chains, and nodes [41]. Considering the same system model as previously, we apply a meta-heuristic algorithm to solve the problem. In contrast to ILP, the latter method scales well albiet returns near optimal solutions. Specifically, we use Particle Swarm Optimization (PSO), whose search is guided by the position of the particles as they progress towards the (near) optimal solution.

Contribution 6 - Formal Analysis Of Multirate Simulink Models

The system model, in particular the software architecture presented in Section 2.3.5 is refined by detailing its software behavior, modeled in Simulink [32], as shown in Figure 10. Simulink is one of the most widely-used graphical modeling and simulation environments of dynamical systems. A



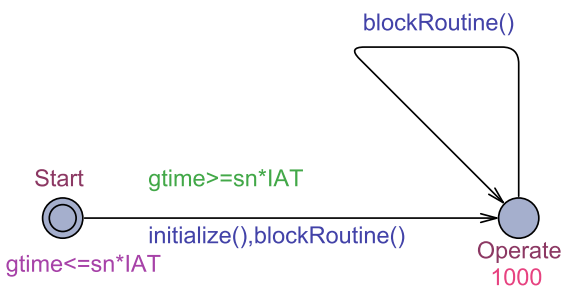
(a) Utilization of nodes.



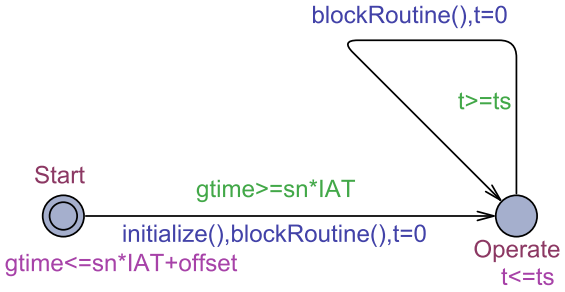
(b) Power consumption of nodes.

Figure 8: Allocation of applications on heterogeneous nodes.

Simulink model is represented via Simulink block diagrams, which are constructed from Simulink blocks. The Simulink blocks implement mathematical functions and communicate to one another via signals. The signals represent data or control messages, and are represented using different dimensions (e.g., scalar, vector, etc.) and data types. The Simulink block library hosts a large set of blocks types that can be used to model multi-domain, continuous, discrete and hybrid systems.



(a) Pattern for Continuous Blocks.



(b) Pattern for Discrete Blocks.

Figure 9: Our Proposed Timed Automata Patterns.

In this contribution, the framework presented in Figure 11 relies on two contributions: the SIMPPAL approach and tool for statistical model checking of large Simulink models [21], validated on two industrial use cases, and its integration with ReSA [the soon-to-be SIMPPAAL meets ReSA paper], which automates requirements specification and their transformation to WMTL properties.In the following, we first explain the SIMPPAAL approach, after which we complete the picture with a short overview of the SIMPPAAL-ReSA integration.

SIMPPAAL takes input the Simulink model and the sorted execution order of blocks, generated by Simulink, and it transforms the former into a network of stochastic timed automata on which we apply UPPAAL SMC for statistical model checking against properties expressed in WMTL. We use a pattern-based encoding, in which continuous-time Simulink blocks are modeled by instantiating the pattern of Figure 9a, and discrete-time ones by instantiating the pattern of Figure 9b. Our pattern-based approach is summarized by the following steps:

- Step 1: We identify the most frequently-used Simulink blocks by analyzing automotive uses cases from obtained from VGTT and Scania. In order to cover as many blocks as possible, we

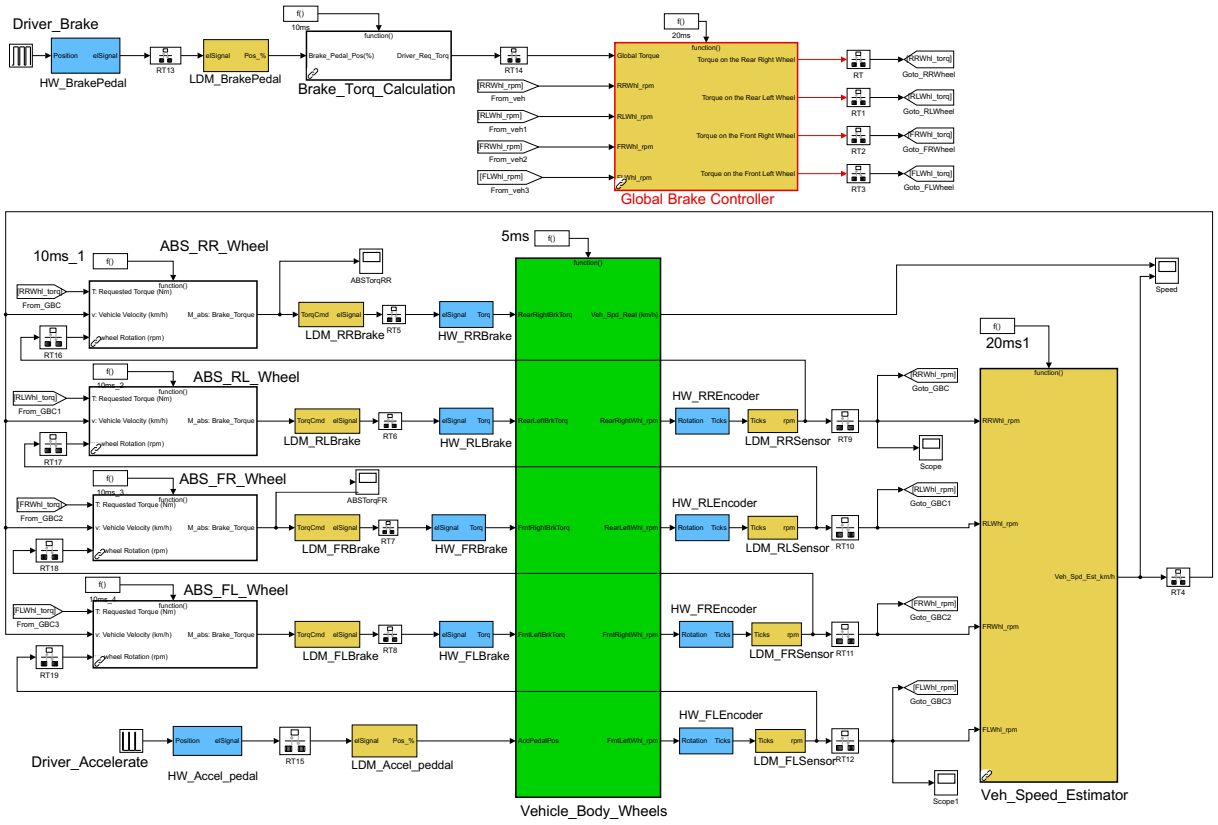


Figure 10: The Brake-By-Wire Simulink Model.

investigate several Simulink models that relate chassis control, engine control, and adaptive driver assistance systems.

- Step 2: We classify the blocks into continuous and discrete blocks based on their timing execution behavior. The continuous blocks execute over infinitely-small intervals of time and the discrete blocks execute periodically with the sample time t_s.

- Step 3: We generalize the blocks as tuple $B = \langle V_{in}, V_{out}, V_D, t_s, Init, blockRoutine \rangle$, where: V_{in}, V_{out}, V_D, are the sets of input, output, data variables, respectively, t_s the a sample time, $Init$ is the initialization function, and $blockRoutine$ is the function that maps input and data variables to output variables.

- Step 4: we propose a transformation that converts the multirate Simulink model into a network of stochastic timed automata via the proposed automata patterns.

- Step 5: Finally, we statistically model check the network of stochastic timed automata against properties specified in the WMTL$_{\leq}$ temporal logic. We check for probability estimation (in which we calculate the probability of some property to hold), as well as for hypothesis testing, in which we verify that a probability to meet some temporal property is higher than a threshold.

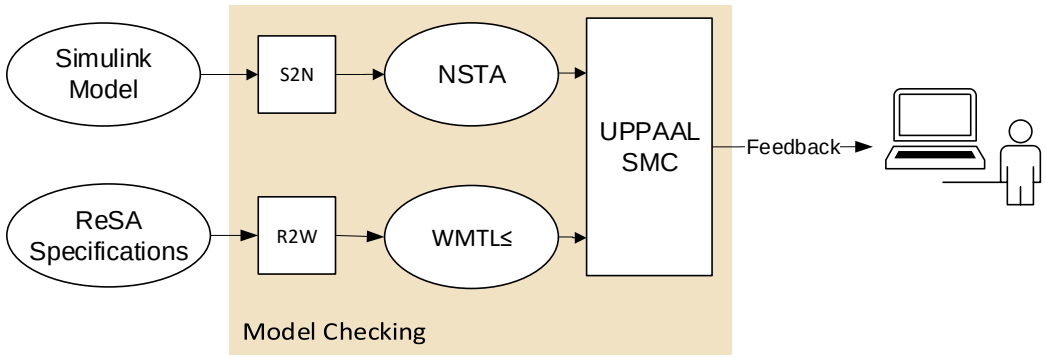


Figure 11: Our Integrated Framework for Model-checking Simulink Models.

The approach is automated in the SIMMPPAL tool, which is accessible from Bitbucket site [4]. Our approch as well as the tool is validated on the Brake-By-Wire industrial Simulink model, shown in Figure 10. The model contains 320 Simulink blocks of which 129 are discrete blocks and 26 are continuous blocks, and the rest are constant blocks. The model is checked against requirements that are translated into WMTL$\leq$ properties [22]. For each properties, the probability of satisfying the requirement, the confidence of the result and other statistics are returned from the model checker. In contrast to exact model checking, the statistical model checking has not experienced state-space explosion, as expected, for the particular case of the Brake-By-Wire model. In conclusion, the result indicates high confidence values in the range $[0.99 - 1]$ on average, which is good enough for many embedded systems applications. However, the confidence value can be higher by increasing the number of traces, which has great importance for safety-critical embedded systems applications.

Since the WMTL$\leq$ logic is difficult to use for most engineers, we provide an interface via the ReSA language, which requires extending the ReSA grammar with probabilistic constructs, as well as interpretation of a subset of the ReSA strings according to the WMTL$\leq$ semantics. This work is in progress and is found in Paper F.

We automate the transformation from the Simulink model to the NSTA model in the tool SIMPPAAL, which is accessible from Bitbucket site https://bitbucket.org/predragf/simppaal/src/master/. Furthermore, we automate the transformation from ReSA to WMTL$_{\leq}$ [Paper F] transformation from ReSA Our method of transformation as well as the tool is validated on the industrial Brake-By-Wire Simulink model, which is obtained from VGTT.

Research Methodology

Research methods, according to Jane et al. [6], are approaches, procedures and guidelines that are applied to conduct research, e.g., observation, interview, prototyping, experiment, etc. In this research, two main factors have driven the selection of research methods: i) the fact that the research is applied in industry (or involves industry-academia collaboration), which means the research results as well as the process should consider the interest and the nature of the industry, and ii) seamless integration of formal methods into existing engineering methods and practices with minimal cost, which requires careful consideration of existing engineering methods, guidelines, tools and capabilities. To summarize the main problems raised by the industry-academia collaboration:

[4] https://bitbucket.org/predragf/simppaal/src/master/

i) the research goals should target existing problems in the industry; ii) existing methods and tools should be leveraged; iii) proposed ideas, methods and tools should be agreed by both academia and industry teams before their design, implementation, validation, and integration to ensure usefulness; iv) furthermore, it is imperative that the proposed tools and methods are engineering-friendly, which means the research team is required to communicate with the actual users to capture the feel of the proposed methods and tools.

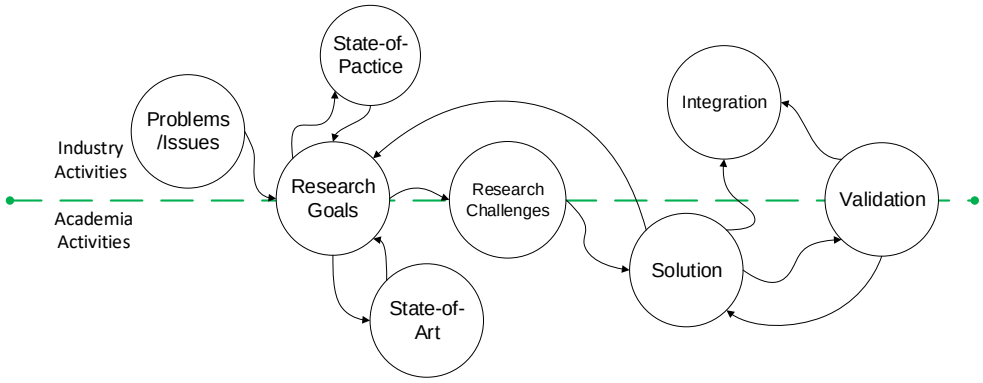


Figure 12: Research Process.

Figure 12 illustrates the research process employed in the thesis, which is an adaptation of the technology-transfer process proposed by Tony et al. [25]. Our main research activities has involved identification of research goals and research challenges, which are followed by solution proposals and their implementations. Subsequently, the solutions are validated on industrial use cases, and are also integrated seamlessly into industrial tool chains. In order to conduct these activities, several quantitative and qualitative research methods are blended in order to gather, analyze and interpret, respectively, quantitative and qualitative data via what is known as *hybrid (or mixed) research* [12]. The benefit of applying mixed research methods is mainly in the triangulation of research outcomes through various research methodical approaches. In fact, this can be better explained by the requirements specification research problem, which has accommodated empirical research via interview, observation to collect quantitative data that has allowed us to understand the current practices and needs of VGTT. Subsequently, we have proposed the ReSA framework, which have been validated through quantitative methods including statistical analysis and questionnaire on its effectiveness and usability[5].

During implementations of the proposed solutions, we have applied a prototyping method [10], which has enabled incremental development of the solutions, before introducing grand progress in subsequent development phases, via concept modeling, implementation, demonstration and revision. This has been the case during the implementation of the ReSA framework [40, 43] as well as the SIMPPAAL framework [21]. The prototyping has involved concept development, and experimentation with toy examples and use cases to internally validate the implementation solutions, before validation by practitioners. Of course, the implementation of the research process has not been straightforward; on the contrary, similar to other research activities, it has accommodated several iterative, cyclic activities to clarify research goals and update solutions based on knowledge gained via literature study and feedback from industry. Besides, the industrial flavor of the research has required persistent synchronization meetings through discussions and workshops in order to

[5]Work in progress - validation of the ReSA toolchain by practitioners, at VGTT.

update to the state-of-the-art and state-of-the-practice methods and tools.

Thesis Outline

The thesis is a collection of papers and its outline is presented as follows:

I Thesis

1 Introduction

2 Preliminaries

2.1 Requirements Specification Methods

A. Template-based Methods

B. Controlled Natural Language

C. Formal Specification Methods

2.2 Boolean Satisfiability Problem

2.3 Ontology

2.4 Software Allocation and Mathematical Optimization

2.5 Simulink

2.6 Stochastic Timed Automata and UPPAAL Statistical Model Checker (SMC)

3 Research Goals

4 Research Method

5 Thesis Contributions

5.1 The ReSA Language

5.2 Formal Semantics of the ReSA Language

5.3 ReSA Framework - Integrated Framework for Formal Analysis of Requirements Specifications

5.4 Optimal Allocation of Multi-rate Distributed Applications

5.5 Formal Analysis of Multi-rate Simulink Models

6 Related Work

7 Conclusions and future work

II Included Papers

1 Paper A

2 Paper B

3 Paper C

4 Paper D

5 Paper E

6 Paper F

Progress And Time Plan

Courses

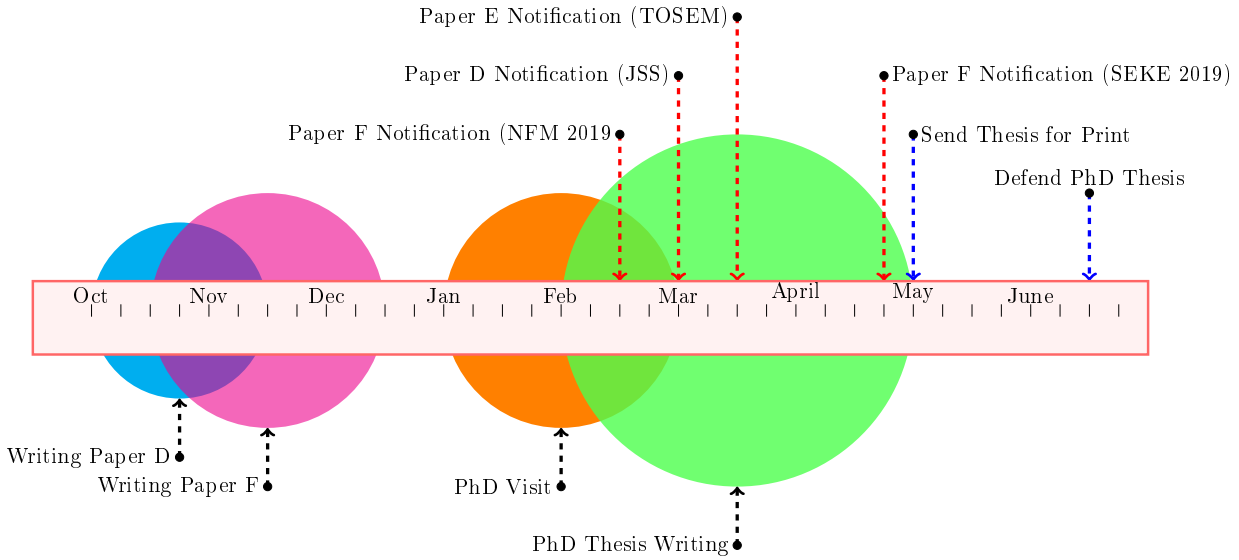
The PhD degree requires 75 ECTS from the courses credit which is met as shown in Table 3.

Courses	Credits	Status
Individual Project	7.5	Finished
VeriSpec Project: ReSA Toolchain Integration into AREATOP IDE, VGTT	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
Safety Critical Systems Engineering	7.5	Finished
Project Management and research commercialization	7.5	Finished
Industrial Systems Cloud Computing	7.5	Finished
International Summer School Marktoberdorf, July 29 to August 10, 2014	3.0	Finished
Total Credits	**75**	

Table 3: List of Courses Taken.

Time Plan

The time plan from the beginning of October to the time the PhD thesis is defended is shown as follows:



Thesis Opponent And Committee

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

1. Opponent: Prof. Joost-Pieter Katoen, RWTH Aachen University, Germany, and University of Twente, the Netherlands
2. Committee members:
   (a) Antonia Bertolino, Research Director at CNR, Italy
   (b) Associate Professor Patrizio Pellicione, Chalmers University of Technology, Sweden
   (c) Professor Peter Csaba Ölveczky, University of Oslo, Norway
3. Reserve: To be decided

Third Cycle Outcome

---

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

I have taken the following graduate courses:

- Advanced component-based software engineering, 7.5 credits
- Research methodology, 7.5 credits
- Industrial Systems Cloud Computing, 7.5 credits
- Research planning, 3 credits
- Dependable Software Systems Engineering, Marktoberdorf Summer School 2014, 3 credits
- Advanced Software Verification and Validation, 7.5 credits
- Design of Cyber-Physical Systems, 7.5 credits
- Safety Critical Systems Engineering, 7.5 credits
- Project Management and research commercialization, 7.5 credits

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, Siegen, Germany, 8-10 June 2015
- IEEE Emerging Technology and Factory Automation (ETFA 2014), Barcelona, Spain, 16-19 September 2014
- 21st International Symposium on Formal Methods, limassol, Cyprus, 9-11 November 2016
- Federated Conference on Computer Science and Information Systems, Trento, Italy, 4-8 September 2017

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, Software Center)

I have authored and co-authored the following papers:

- The Multi-Resource Server for Predictable Execution on Multi-core Platforms (Apr 2014) Rafia Inam, Nesredin Mahmud, Moris Behnam, Thomas Nolte, Mikael Sjödin
- Evaluating Industrial Applicability of Virtualization on a Distributed Multicore Platform (Sep 2014) Nesredin Mahmud, Kristian Sandström, Aneta Vulgarakis Feljan
- ReSA: An Ontology-based Requirement Specification Language Tailored to Automotive Systems (Jun 2015) Nesredin Mahmud, Cristina Seceleanu, Oscar Ljungkrantz
- ReSA Tool: Structured Requirements Specification and SAT-based Consistency-checking, Nesredin Mahmud, Cristina Seceleanu, Oscar Ljungkrantz
- Simulink to UPPAAL Statistical Model Checker: Analyzing Automotive Industrial Systems (Nov 2016) Predrag Filipovikj, Nesredin Mahmud, Raluca Marinescu, Cristina Seceleanu, Oscar Ljungkrantz , Henrik Lönn
- Semantic Analysis of Embedded System Requirements Specifications, Nesredin Mahmud, Cristina Seceleanu, Oscar Ljungkrantz,Technical Report
- Specification and Semantic Analysis of Embedded Systems Requirements: From Description Logic to Temporal Logic, Nesredin Mahmud, Cristina Seceleanu and Oscar Ljungkrantz
- Power-aware Allocation of Fault-tolerant Multi-rate AUTOSAR Applications, Nesredin Mahmud, Guillermo Rodriguez-Navas, Hamid Faragardiy, Saad Mubeen, Cristina Seceleanu
- Ontology-based Analysis and Scalable Model Checking of Embedded Systems Models, Nesredin Mahmud

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

I have taken the following graduate courses:

- Advanced Validation and Verification, 7.5
- Safety-critical Systems Engineering, 7.5

I have visited Volvo Groups Trucks Technology (VGTT), Gothenburg several times.

- I spent time investigating existing engineering tools, and development methods.
- I interviewed senior developers, group managers, and leaders from various departments.
- I have conducted empirical study on requirements specification at VGTT.
- I have used VGTT automotive use cases to validate our proposed solutions.
- I participated in the VGTT department meetings.

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:

- Research methodology, 7.5 credits

Deep knowledge of research methods in general area:

- research problem solving in industrial context
- embedded systems, cyber-physical research method

Deep knowledge of research methods in specific area:

- research solving for improving quality of embedded systems in automotive domain
- automotive system design and anlysis method
- software evaluation method in automotive domain

Remaining Activities	

- Writing Paper D
- Writing Paper F
- Writing PhD thesis

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 1.1a.
- I have written basic and advanced level thesis proposals:
  - Automated Transformation of Requirements from Natural language to Formal specification
  - Integration of ReSA into an Industrial IDE
- I have written my licentiate thesis proposal.
- I have written my PhD thesis proposal.
- I maintained lab assignments.

2.2b Contribute to the development of knowledge	YES

I contributed research results to research communities, shown in 1.1.a.
I assisted the following labs:

- Software Development for Real-Time Systems, DVA421 (7.5 credits) in 2017,18
- DVA218 vt 2017 Datakommunikation, DVA218 (7.5 hp) in 2015, 16, 17,18
- Datakommunikation för inbyggda system I, DVA235 (7.5 credits) in 2016
- Project in advanced embedded systems, DVA474, (7.5 credits) in 2015, 16, 17 times)

I supervised students:

- master level: *Object Detection For High Productivity, 2015*
- master level: *Automated Transformation of Requirements from Natural language to Formal specification, 2016* (Interrupted)

I contributed research results to industry.

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

- 19th IEEE International Conference on Emerging Technologies and Factory Automation, Barcelona
- The 36th IEEE Software Engineering Workshop (SEW-36), Gdansk
- Fuse, Joint Seminar on Functional Safety, Virtual Integration and Model Based Engineering, Göteborg
- I participated in VGTT, Scania södertälje workshop, and meetings
- Federated Conference on Computer Science and Information Systems, Trento, Italy, 4-8 September 2017

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
- I conducted research visits at VGTT, Scania södertälje
- I have developed research prototypes
- I have validated research results on industrial uses cases
- I have conducted empirical research at VGTT on requirements specification
- I initiated research cooperation on some research topics and have worked with it.

3. Judgment and approach	Completed
3.1 Demonstrate ability to make ethical assessments in own research	YES
I have participated in the planning of my own research I have taken courses that contributed knowledge to ethical assessments such as: • Research Methods Course, 7.5credit • Research Planning Course, 4.5credit I have worked together with industrial partners which I have learned individuals and organizational ethics.	
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 with VGTT, Scania • I participated in research projects, in general which had required ethical practices.	
3.3 Identify one's need of further knowledge and take responsibility for one's learning	YES
• I have authored and co-authored papers listed in 1.1a. • I have participated and presented research results in international venues. • I have participated in seminars and workshops to exchange knowledge. • I have developed software engineering tools, in particular relevant for industry. • I have visited VGTT, Gutenberg to better understand research problems.	

Related Work

In this section are discussed the related work on specification and analysis of requirements, allocation of software architecture, and formal analysis of behavioral models, designed in Simulink.

Requirements Specification And Analysis

Embedded system requirements are captured in different representations, e.g., textual, tabular, graphical, etc. The textual representation, which is the scope of the analysis, can be conveniently classified into two classes: i) controlled natural language (CNL), ii) template-based methods. The syntax and semantics of the CNLs are similar to natural language except that the lexicon and the syntax are restricted for different reasons, of which improving comprehensibility of the text and formal representations to support rigour analysis of the text are prominent. The ReSA language is designed to improve comprehensibility of requirements specifications as well as to be computer-processable. There are many computer-processable CNL in literature [35], e.g., Attempto Controlled English (ACE) [24], Processable English (PENG) [55], etc. Similar to most computer-processable languages, ReSA has limited syntactic constructions, and allows knowledge-representations, in contrast though, our language is catered for embedded systems and therefore it uses concepts as well as semantic rules that are domain-specific to embedded systems. Similar to PENG, its implementation supports look-ahead in order to enable predictive and guided specification.

The template-based methods, in particular requirements boilerplate uses templates (or boilerplates) which are reusable, recurrent patterns to specify requirements, e.g., CESAR boilerplates [18], RAT, etc. The main drawback of existing requirements boilerplates are: i) the

templates are usually too limited therefore not expressive enough, ii) it is not easy to find the appropriate boilerplate during specification. In this regard, ReSA extends boilerplates with a meta-model that guides for plausible instantiation of boilerplates.

Optimal Allocation Of Software Architectures

Different allocation schemes deliver different system performance and therefore efficient software allocation is crucial. Ernest Wozniak et al. [61] proposed a synthesis mechanism for an AUTOSAR software application that can executed over multiple nodes, with the objective fulfilling timing requirements. In contrast, we consider power consumption and reliability requirements besides timing. Similarly, Salah Saidi et al. [53] proposed an ILP based approach for allocation of an AUTOSAR application on a multi-core framework in order to reduce the overhead of inter-process communication while we consider a multi-nodes platform. Ivan Svogor et al. [58] proposed a generic approach of identifying resource constraints and a way of handling different measurement units with Analytic Hierarchy Process (AHP) in order to allocate a component-based software application on a heterogeneous platform. However, the resource constraints are trivialized, e.g., end-to-end delay calculations, which require timing specifications and activation patterns of tasks. As opposed to the previously mentioned related work, we considered a system model with a multi-rate software application, which basically impose complex timing analysis due to complex timed paths from the source to the sink of communication signals [47]. On a different work, there are research focusing on power and energy consumption in real-time distributed systems which are employ dynamic voltage scaling [4] and task consolidation by minimizing computational nodes [17] [14].

Formal Analysis Of Simulink Models

Several research endeavors have tackled the problem of formally analyzing Simulink models in order to gain better insight into the design of Simulink model, and they mainly differ in the aspects (or coverage) of the Simulink language that has been targeted for analysis and the formalisms applied. Besides to the scalability of the techniques, their robustness to formally analyze various types of Simulink models is also crucial especially for the proposed methods to be useful in industry. In this related work on formal analysis of Simulink models, existing formalisms and their applicability in industry are discussed, and also compared to our solutions.

Simulink already suppors formal analysis of Simulink models via its Simulink Design Verifier (SDV) product[6]. Though, there is limited resource to investigate about the pros and cons of the product, the study by Nellen et al. [50] indicates some limitations on its capability such as inconclusiveness on the verification results, lack of support to verify timed properties. The latter pros is also mentioned by Florian et al. [39] in the comparison against SPIN. For brevity, other approaches are classified into three categories, approaches that use Simulink traces, contract/theorem and model-to-model transformation.

The PlasmaLab proposed by Nikolaos et al. [33] transforms Simulink sample traces into statistical models, which are eventually analyzed by their statistical mode checker. Albeit the checker is assisted by an algorithm that determines sufficiency of the sample traces, it is unclear how it works. Unlike many approaches, PlasmaLab can analyze any Simulink models as long as the simulation results sample trances, which is the main advantage. Ferrante et al. [28] use contract-based theory in order to lift the block specification, and rely on a combination of SAT solvers and the NuSMV model checker for analysis. Hocking et al. [20] use the PVS specification language for writing the specification, and rely on the PVS theorem prover for analysis. A limitation of this strategy is that

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

both steps still require much user interaction, so it is error-prone and requires certain understanding of the formal analysis engines, which is not common among embedded systems engineers.

The model-to-model transformation approach basically transforms the Simulink model into a formal model that can be checked via model checking, e.g., for reachability properties. Bernat et al. [46] proposed transformation Simulink models that consist only discrete blocks, which are subsequently checked via the DiViNE model checker. The research endeavors proposed transformation only StateFlow/Simulink into timed and hybrid automata. The discussed model-to-model approaches are limited in scalability due the use of exact model checking. In contrast, our approach uses statistical model checking, and thus scales better albeit is not exhaustive. However, by collecting sufficient sample traces and consequently acceptable confidence value in the statistical analysis, the later challenge can be tackled. Furthermore, our approach supports transformation of any discrete and continuous blocks, and also blocks that are custom and StateFlow. The latter capability is achieved since our transformation abstracts the internal implementation of the blocks.

Conclusions And Future Work

Improving functional safety and quality of embedded systems is crucial due to the catastrophic consequences of their failure. Besides their applications in ragged environment, operating for a along time without interruption and the fact that such system are usually resource constrained, call for a systematic development approach. In this thesis, we have proposed several formal techniques for improving functional safety and quality of embedded systems, including requirements specifications, software system architecture, and software behavior, modelled in Simulink.

We proposed a domain-specific language for the specification of embedded systems requirements called ReSA. The language resembles the natural language English in syntax and semantics. However, its syntax is constrained in order to improve comprehensibility of the specifications. The language has formal semantics in Boolean and description logic, which has enabled for superficial and deep analysis of the requirements, e.g., consistency checking. The ReSA toolchain contains a ReSA editor that supports content-completion to guide requirements specification, and the specifications can be checked for consistency via the Z3 SMT solver. The language is validated for its expressively on a set of 300 industrial requirements, and has proven to express about 90% of the requirements.

We have also provided a mechanism to preserve timing and reliability requirements of multirate software applications during software allocation, while minimizing power consumption via Integer Linear Programming, ILP (exact) and Particle Optimization, PSO (meta-heuristic) methods. The ILP method is shown to be applicable to allocation of small and medium of software applications, whereas the PSO has shown to scale well for allocation of large software applications.

Furthermore, we have introduced an automated pattern-based, order-preserving method for transforming behavioral embedded systems models in Simulink into networks of stochastic timed automata analyzable by UPPPAAL SMC. The method is implemented in our tool SIMPPAAL that we have integrated with ReSA and validated on the BBW industrial prototype.

The ReSA language is a descriptive and intuitive language as it resembles natural language. By extending its constructs to encompass design constraints, it should be possible and in fact beneficial to synthesize a high-level architecture, using correct-by-construction. Furthermore, by raising detailed hardware platform specifications in to the system design, e.g., memory, CPU, power specification, effective software to hardware allocation can be achieved. The proposed formal analysis of Simulink models can be generalized to other language that comply to a data-flow programming paradigms, e.g., LabView.

References

[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] Sten Agerholm and Peter Gorm Larsen. A lightweight approach to formal methods. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1999.

[3] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. The Description Logic Handbook: Theory, Implementation and Applications. *Kybernetes*, 2010.

[4] Mario Bambagini, Mauro Marinoni, Hakan Aydin, and Giorgio Buttazzo. Energy-aware Scheduling for Real-time Systems: A survey. *ACM Transactions on Embedded Computing Systems (TECS)*, 15(1):7, 2016.

[5] Matthias Becker, Dakshina Dasari, Saad Mubeen, Moris Behnam, and Thomas Nolte. End-to-end timing analysis of cause-effect chains in automotive embedded systems. *Journal of Systems Architecture*, 2017.

[6] Bruce L (Bruce Lawrence) Berg and 1962-Author Lune Howard. *Qualitative research methods for the social sciences*. Boston Pearson, eighth edi edition, 2012.

[7] K Bond. IEC 61511-Functional Safety: Safety Instrumented Systems for the Process Industry Sector. In *Annual Symposium on Instrumentation for the Process Industries*, volume 57, pages 33–40. Instrument Society Of America, 2002.

[8] Peter Braun, Manfred Broy, Frank Houdek, Matthias Kirchmayr, Mark Müller, Birgit Penzenstadler, Klaus Pohl, and Thorsten Weyer. Guiding requirements engineering for software-intensive embedded systems in the automotive industry: The REMsES approach. *Computer Science - Research and Development*, 2014.

[9] 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.

[10] Mahil Carr and June Verner. Prototyping and Software Development Approaches. *Prototyping and Software Development Approaches*, 2004.

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

[12] J W Creswell. *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. 2014.

[13] Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient SMT Solver. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2008.

[14] Vinay Devadas and Hakan Aydin. On the Interplay of Voltage/Frequency Scaling and Device Power Management for Frame-based Real-time Embedded Applications. *IEEE Transactions on Computers*, 61(1):31–44, 2012.

[15] David Devlin and Barry OSullivan. Satisfiability as a classification problem. In *Proc. of the 19th Irish Conf. on Artificial Intelligence and Cognitive Science*, 2008.

[16] Christof Ebert and Capers Jones. Embedded Software: Facts, Figures, and Future. *Computer*, 42(4):42–52, 4 2009.

[17] Hamid Reza Faragardi, Aboozar Rajabi, Reza Shojaee, and Thomas Nolte. Towards Energy-aware Resource Scheduling to Maximize Reliability in Cloud Computing Systems. In *High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC), 2013 IEEE 10th International Conference on*, pages 1469–1479. IEEE, 2013.

[18] Stefan Farfeleder, Thomas Moser, Andreas Krall, Tor Stålhane, Herbert Zojer, and Christian Panis. DODT: Increasing requirements formalism using domain ontologies for improved embedded systems development. In *Proceedings of the 2011 IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems, DDECS 2011*, 2011.

[19] Nico Feiertag, Kai Richter, Johan Nordlander, and Jan Jonsson. A Compositional Framework for End-to-end Path Delay Calculation of Automotive Systems under Different Path Semantics. In *IEEE Real-Time Systems Symposium: 30/11/2009-03/12/2009*. IEEE Communications Society, 2009.

[20] Orlando Ferrante, Luca Benvenuti, Leonardo Mangeruca, Christos Sofronis, and Alberto Ferrari. Parallel NuSMV: A NuSMV extension for the verification of complex embedded systems. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2012.

[21] 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.

[22] Predrag Filipovikj, Nesredin Mahmud, Raluca Marinescu, Guillermo Rodriguez-Navas, Cristina Seceleanu, Oscar Ljungkrantz, and Henrik Lönn. Analyzing Industrial Simulink Models by Statistical Model Checking. Technical report, 3 2017.

[23] 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, 8 2014.

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

[25] Tony Gorschek, Per Garre, Stig Larsson, and Claes Wohlin. A model for technology transfer in practice. *IEEE Software*, 2006.

[26] K. Grimm. Software technology in an automotive company - major challenges. In *25th International Conference on Software Engineering, 2003. Proceedings.*, pages 498–503. IEEE, 2003.

[27] Volker Gruhn and Ralf Laue. Patterns for Timed Property Specifications. *Electronic Notes in Theoretical Computer Science*, 2006.

[28] Ashlie B. Hocking, M. Anthony Aiello, John C. Knight, and Nikos Aréchiga. Proving Critical Properties of Simulink Models. In *Proceedings of IEEE International Symposium on High Assurance Systems Engineering*, 2016.

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

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

[31] Daniel Jackson. Lightweight Formal Methods. In José Nuno Oliveira and Pamela Zave, editors, *FME 2001: Formal Methods for Increasing Software Productivity*, page 1, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

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

[33] Nikolaos Kekatos, Marcelo Forets, and Goran Frehse. Constructing verification models of nonlinear Simulink systems via syntactic hybridization. In *2017 IEEE 56th Annual Conference on Decision and Control, CDC 2017*, 2018.

[34] Sascha Konrad and Betty H. C. Cheng. Real-time specification patterns. In *Proceedings of the 27th international conference on Software engineering - ICSE '05*, page 372, New York, New York, USA, 2005. ACM Press.

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

[36] Phillip A. Laplante. *Requirements Engineering for Software and Systems, Third Edition*. Auerbach Publications, 10 2017.

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

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

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

[40] 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.

[41] Nesredin Mahmud, Guillermo Rodriguez-Navas, Hamid Reza Faragardi, Saad Mubeen, and Cristina Seceleanu. Power-aware Allocation of Fault-tolerant Multi-rate AUTOSAR Applications. In *25th Asia-Pacific Software Engineering Conference*, 12 2018.

[42] 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.

[43] Nesredin Mahmud, Cristina Seceleanu, and Oscar Ljungkrantz. Specification and Semantic Analysis of Embedded Systems Requirements: From Description Logic to Temporal Logic. pages 332–348. Springer, Cham, 9 2017.

[44] Serguei Mankovskii, Martin Gogolla, Susan D. Urban, Suzanne W. Dietrich, Susan D. Urban, Suzanne W. Dietrich, Ming-Hsuan Yang, Gillian Dobbie, Tok Wang Ling, Terry Halpin, Bettina Kemme, Nicole Schweikardt, Alberto Abelló, Oscar Romero, Ricardo Jimenez-Peris, Robert Stevens, Phillip Lord, Tom Gruber, Pieter De Leenheer, Avigdor Gal, Sean Bechhofer, Norman W. Paton, Changqing Li, Alejandro Buchmann, Nikos Hardavellas, Ippokratis Pandis, Bing Liu, Marc Shapiro, Ladjel Bellatreche, Peter M. D. Gray, W. M. P. Aalst, Nathaniel Palmer, Nathaniel Palmer, Tore Risch, Wojciech Galuba, Sarunas Girdzijauskas, and Sean Bechhofer. OWL: Web Ontology Language. In *Encyclopedia of Database Systems*, pages 2008–2009. Springer US, Boston, MA, 2009.

[45] Luiz Eduardo G. Martins and Tony Gorschek. Requirements Engineering for Safety-Critical Systems: Overview and Challenges. *IEEE Software*, 34(4):49–57, 2017.

[46] B. Meenakshi, Abhishek Bhatnagar, and Sudeepa Roy. Tool for Translating Simulink Models into Input Language of a Model Checker. pages 606–620. Springer, Berlin, Heidelberg, 2006.

[47] Saad Mubeen, Jukka Mäki-Turja, and Mikael Sjödin. Support for End-to-end Response-time and Delay Analysis in the Industrial Tool Suite: Issues, Experiences and A Case Study. *Computer Science and Information Systems*, 10(1):453–482, 2013.

[48] 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.

[49] Nico Naumann. AUTOSAR Runtime Environment and Virtual Function Bus. *Hasso-Plattner-Institut, Tech. Rep.*

[50] Johanna Nellen, Thomas Rambow, Md Tawhid Bin Waez, Erika Ábrahám, and Joost-Pieter Katoen. Formal Verification of Automotive Simulink Controller Models: Empirical Technical Challenges, Evaluation and Recommendations. pages 382–398. Springer, Cham, 7 2018.

[51] Gerard O'regan. *Undergraduate Topics in Computer Science*.

[52] Amalinda Post, Igor Menzel, Jochen Hoenicke, and Andreas Podelski. Automotive behavioral requirements expressed in a specification pattern system: a case study at BOSCH. *Requirements Engineering*, 17(1):19–33, 3 2012.

[53] Salah Eddine Saidi, Sylvain Cotard, Khaled Chaaban, and Kevin Marteil. An ILP Approach for Mapping AUTOSAR Runnables on Multi-core Architectures. In *Proceedings of the 2015 Workshop on Rapid Simulation and Performance Evaluation Methods and Tools - RAPIDO '15*, pages 1–8, New York, USA, 2015. ACM Press.

[54] Alberto Sangiovanni-Vincentelli, Luca Carloni, Fernando De Bernardinis, and Marco Sgroi. Benefits and Challenges for Platform-based Design. In *Proceedings of the 41st annual conference on Design automation - DAC '04*, page 409, New York, USA, 2004. ACM Press.

[55] R Schwitter. English as a formal specification language. In *Proceedings. 13th International Workshop on Database and Expert Systems Applications*, pages 228–232. IEEE Comput. Soc, 2002.

[56] Rob Shearer, Boris Motik, and Ian Horrocks. HermiT: A Highly-Efficient OWL Reasoner. In Catherine Dolbear, Alan Ruttenberg, and Ulrike Sattler, editors, *OWLED*, volume 432 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[57] Ernst Sikora, Bastian Tenbergen, and Klaus Pohl. Requirements engineering for embedded systems: An investigation of industry needs. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2011.

[58] Ivan Švogor, Ivica Crnkovic, and Neven Vrcek. An Extended Model for Multi-Criteria Software Component Allocation on a Heterogeneous Embedded Platform. *Journal of computing and information technology*, 21(4):211–222, 2014.

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

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

[61] Ernest Wozniak, Asma Mehiaoui, Chokri Mraidha, Sara Tucci-Piergiovanni, and Sébastien Gerard. An Optimization Approach for the Synthesis of AUTOSAR Architectures. In *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, 2013.