Contents

1. Introduction

REAL-TIME systems are usually characterized by timely computations, which are bounded by *deadline*, besides correct results of the computations [5]. They are applied in many *safety-critical embedded* systems, which are specialized computer systems designed for safety-critical applications [32], e.g., the braking system inside vehicles applies proportional force on the wheel to the pressing of a brake pedal in order to slow down (or halt) the vehicle, and must act within sometime otherwise the system fails, consequently, accident can happen. Therefore, safety-critical real-time systems should be analyzed rigorously for functional and timing correctness, which is also specified in the functional safety standards, such as the ISO 26262 "Road vehicles-Functional safety" [15]. The latter standard also suggests the use of *formal methods*, which are mathematical techniques and tools that enable unambiguous specification, modeling and rigorous analysis [31], to develop safety-critical automotive systems.

In distributed computing [17], the safety-critical software is mapped on multiple hardware systems to capitalize on the computational power provided by the distributed architecture, e.g., the braking software can be executed on multiple electronic control units (ECU). Since the distributed software is normally exposed to a greater degree of permanent and transient faults, reliability of the safety-critical software should be maximized to improve dependability of the system which requires additional critical systems resource such as power and energy besides computational resources. However, the embedded hardware is usually resource constrained, therefore, the software should be efficiently mapped to the hardware to conserve critical system resources, thereby accommodate current and future growth of the software functionality.

In this thesis, we apply formal methods to improve the requirements specifications of safety-critical systems, and to analyze the functional and timing behavior of the safety-critical software against the specifications. The safety-critical specifications should be unambiguous, comprehensible, etc [1]. According to the ISO 26262 standard, semi-formal or formal languages are recommended to specify safety-critical requirements. However, natural language is the de facto method to specify embedded systems requirements in industry because it is intuitive and expressive, though inherently ambiguous [1]. In the context of natural language, template-based specification and controlled natural language can be considered semi-formal

and formal specification methods. The template-based specification methods, e.g., requirements boilerplates [14], property-specification systems [8], etc., lack meta-model to effectively create templates, and is usually cumbersome to select the templates. The controlled natural languages, e.g., Attempto [13][12], etc., renders the syntax and semantics of the natural language and have formal semantics, however lacks support for embedded systems, hence are less effective. In this thesis, we propose a constrained natural language which is domain-specific and uses the notion of boilerplates to facilitate reuse. The specifications have semantics in Boolean logic and description logic to enable rigorous analysis via Boolean satisfiability [26] and ontology [3], respectively.

The specifications are employed in subsequent system development including software design to verify the latter for correct functionality. The software design is usually modeled, simulated and analyzed before implementation. In this regard, Simulink is one of the most widely used development environment for multi-domain, multi-rate, discrete and continuous safety-critical systems in industry [16]. For this main reason, there is increasing interest in formal analysis of Simulink models [27]. Simulink Design Verifier[1], which is based on the Prover[2] model checker, is the de facto tool in the Simulink environment to formally verify Simulink design models. However, it has limited functionality, e.g., it supports only discrete models, has issues with scalability due to state-space explosion, and lacks verification of timed properties [20]. In contrast, we propose a scalable, timed analysis via a statistical model checking [19], which uses traces of executions and statistical analysis techniques, e.g., monte-carlo simulation, etc., first by transforming Simulink models into a network of stochastic timed automata using timed-automata patterns [11].

The software design should be mapped to hardware effectively, that is satisfying the timing and reliability requirements of the distributed safety-critical software, but also efficiently to minimize the power consumption of the distributed system to facilitate extensibility of the software, and also accommodates the demand from the increasing software functionality. We consider the software is scheduled using a fixed-priority preemptive policy, which is quite common in industry, and posses end-to-end timing requirements, e.g., the time duration between the brake-pedal press and the slow-down (or halt) action. Furthermore, we consider the fault tolerance as a means to maximize reliability of the distribute safety-critical software by mapping redundant software functionality on different computing units. We propose *exact* and *heuristic* optimization methods, which deliver optimal and near-optimal solutions, respectively, to efficiently map the distributed safety-critical software to a network of computing units. Specifically, we propose a formulation of integer-linear programming

[1]Simulink Design Verifier - https://se.mathworks.com/products/sldesignverifier.html
[2]Prover - https://www.prover.com/software-solutions-rail-control/formal-verification/

(ILP) [23], which is solved using branch and bound. Furthermore, we propose a hybrid-particle optimization [28], which is a meta-heuristic algorithm, to solve the shortcomings of the exact method for large-scale problems [24] with trade-off over non-optimality.

1.1 Research Contributions Overview

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

- **Formal Analysis of natural language requirements:** we propose a fairly expressive, flexible yet structured and domain-specific constrained natural language, called *ReSA* [21][25]. The language has semantics in Boolean and description logic to support for shallow and rigorous analysis, respectively. The Boolean specifications are checked for consistency using the satisfiability-modulo theory via the Z3 SMT solver. Whereas, the description logic is used to encode the specification as ontology, where we check consistency of the specifications at the lexical level using Reasoner (Inference engine) such Hermit. The ReSA tool, which consists of an editor and implements consistency-checking functionality, is integrated seamlessly into EATOP, which is an open source EAST-ADL IDE, to complement the requirements modeling.

- **Scalable analysis of Simulink models:** we propose a pattern-based, execution-order preserving automatic transformation of atomic and composite Simulink blocks into stochastic timed automata that can be formally analyzed using UPPAAL Statistical Model Checker [4]. Our method is scalable, and has been validated on industrial use cases [10]. The statistical model checker analyzes a state-transition system by conducting statistical analysis on the collected traces of the system executions, effectively mitigating the state-space explosion of (exact) model checking [19].

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

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

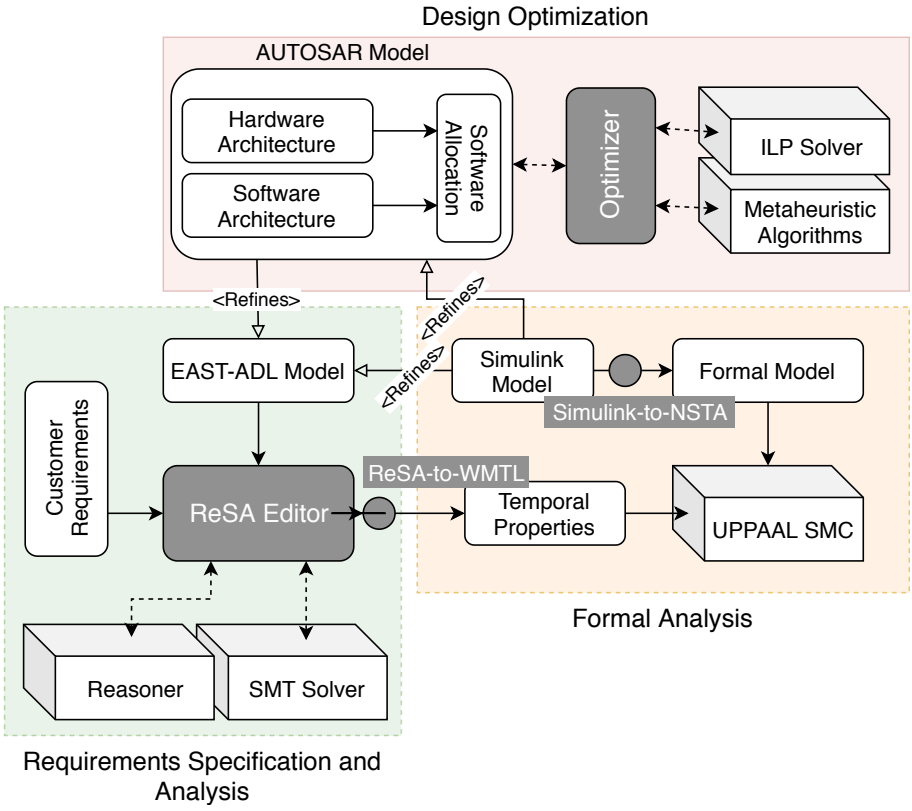


Figure 1.1: Thesis Contributions Workflow.

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

1.2 Thesis Outline Overview

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

2. Preliminary

3. Problem Formulation

The automotive electrical/electronic system executes complex safety-critical software, e.g., x-by-wire software, engine control, traction control, etc. Over the last decades, the complexity of the safety-critical software has been on the rise which is evident on the modern cars, which implement many and complex automotive functions, and also on the emergence of the electrical and autonomous vehicles. Thus, the thesis is motivated by the need for advanced (or rigorous) methods to the requirements specification, modeling and analysis of the complex safety-critical automotive software, and their seamless integration into the existing methods and tools of the automotive systems development at VGTT and Scania. Furthermore, the thesis is motivated by the need for efficient mapping of safety-critical software to hardware in the distributed computing to facilitate software extensibiliy and support the increasing functionality of the automotive systems.

Thus, the *overall goal* of the thesis is to:

> **Overall Goal** — Provide assurance of safety-critical functionality, at the various levels of abstraction, via formal analysis, and optimization of critical system resources.

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

3.1 Research Goals

Many safety-critical automotive systems are developed according to the ISO 26262 standard, which recommends highly the use of semi-formal languages to specify safety-critical requirements to improve quality of the specifications, e.g., by reducing ambiguity and improving comprehensibility. In the context of textual representations, the semi-formal specification methods are constrained natural languages, such as templates (e.g., requirements boilerplates [9][25]), controlled natural languages [18](e.g., Attempto [13]).

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

Thus, the first research goal is to:

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

One of the mechanisms to improve natural language specifications is by constraining the language, including its syntax, semantics and the lexicon [18]. The design of a constrained natural language for the specification of requirements is not trivial. By constraining the language, its expressiveness and intuitiveness can be impaired [1][30], therefore, appropriate trade-offs should be made during the design in order to have a robust and effective specification language.

Besides improving quality of individual requirements, the latter should be analyzed in ensemble in order to detect errors that span multiple specifications, e.g., logical contradictions. However, natural language lacks formal (or precise and unambiguous) semantics, therefore is difficult to rigorously analyze (or reason) natural-language requirements specifications. There are several methods to natural language semantics, of which the use of *logic* is common [6].

Thus, the second research goal is to:

> **RG 2:** — Facilitate formal analysis of the requirements specifications through transformation to Boolean and description logics.

Natural language specifications are constructed from syntactic units, such as words, phrases, clauses, statements, etc. Consequently, rigorous analysis of the specifications involve parsing and interpreting the syntactic units, which is a complex problem in computational linguistics [6]. The depth of the interpretation (or semantics) greatly affects the applicability of the methods, e.g., the propositional logic representation of the specifications is simple and the analysis scales well, however, it is shallow as it abstracts away the details. On the other hand, the first-order-logic representations are more rigor, thus enable thorough analysis but are less tractable. Therefore, appropriate interpretation of the natural language specifications is crucial.

The software designs and software-design units (or behavioral models) should conform to the requirements specifications. We consider the software-design units are modeled in Simulink, which is the most widely used model-based development environment in industry to model and simulate the behavior of multi-domain, discrete, continuous embedded systems. Simulink also enables the generation of code from discrete Simulink models which directly execute on specific platforms, thus is crucial to

conduct rigorous analysis of the Simulink models to reduce errors introduced in the generated code.

The de facto Simulink analysis techniques, e.g., by type checking, simulation, and formal verification via the Simulink Design Verifier (SDV[1]) are not sufficient to address the full correctness of safety-critical real-time Simulink models. SDV lacks support for checking temporal correctness as specified in timed properties, e.g., in TCTL, and also lacks support for verifying continuous models and suffers from scalability due to its reliance on the exact model-checking [20]. In contrast to the exact model checking, the statistical model-checking verifies properties over sufficiently collected traces of system simulations via statistical methods. It scales better over the trade-off for exhaustiveness.

Thus, the third research goal of the thesis is to:

> **RG 3:** — Enable formal analysis of large-scale, multi-rate and hybrid Simulink models using statistical model-checking.

Simulink consists of connected and hierarchical Simulink blocks, which encode mathematical functions [16]. 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 the model checking challenging.

In the distributed computing, the automotive software is allocated on multiple computing units (or ECU), consequently is exposed to higher permanent and transient faults, hence necessitates to maximize the reliability of the safety-critical software system. Fault tolerance using redundancy is the most widely approach to improve reliability such as by replicating software functionality on multiple ECU, however, it requires additional critical system resources such as power sources. In this regard, the software-to-hardware allocation plays a crucial role to minimize the power consumption of fault-tolerant distributed safety-critical software while satisfying the timing and reliability constraints of the safety-critical software.

Thus, the fourth research goal is to

> **RG 4:** — Minimize the power consumption of distributed safety-critical software while satisfying the timing and reliability constraints during the software allocation.

The software allocation model is not trivial as we consider exact method of schedulability analysis and reliability calculations, which makes the

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

optimization complex. We assume a sufficient and necessary scheduling test based on the worst-case response time analysis [2][7], moreover end-to-end delay analysis based on the age delay semantics [29], which is practical but computationally expensive. For the reliability analysis, we apply an exact method of calculation based on the state enumeration, in contrast to the series-parallel method, which is trivial and computationally less expensive. As a result, we consider an exact optimization method using ILP for relatively small and metaheuristics for relatively large software allocation problems.

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

Thus, the last research goal is to:

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

Seamless integration of our proposed methods and tools into the existing development process require close cooperation between the domain experts and the practitioners. The role of the domain experts should be to simplify usage of the tools, e.g., by rendering their interface to existing once, etc., and the practitioners should cooperate with materials that assist the validation of the proposed solutions. The cooperation is not trivial considering the challenge of formal methods, and companies culture for being restrictive.

4. Contributions

The general contribution of the thesis is a safety-critical design approach that employs formal methods at various stages of software development such as requirements specification and software design, and a power-efficient and integrated mapping of the software to hardware while satisfying timing and reliability of the software in a distributed environment. It satisfies the research goals explained in Section x via the specific contributions: a requirements specification editor [25][21], a formal analysis of the specifications [21][22], a formal analysis of Simulink models [11] and optimization of software mappings in the context of distributed architecture [23][24]. The contributions are contextualized in a development framework that make use of EAST-ADL, AUTOSAR and Simulink, as illustrated by the workflow diagram shown in Figure 4.1.

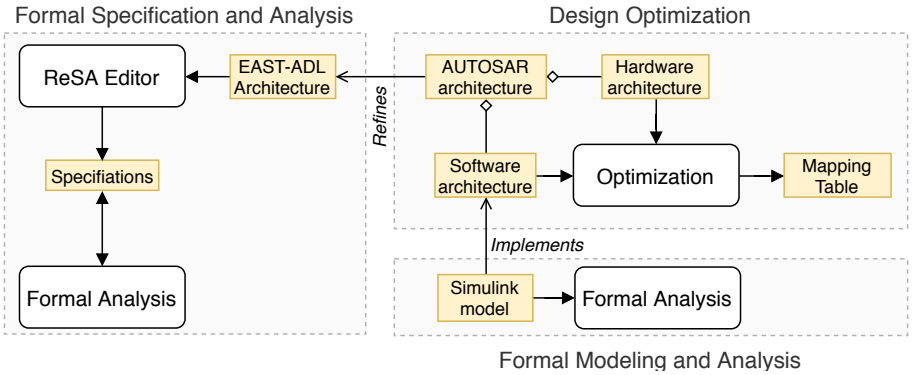


Figure 4.1: Thesis contributions workflow.

Initially requiremens are specified in our domain-specific language ReSA, which is a constrained natural language, by accessing model elements from an EAST-ADL architecture. The specification, which supports rudimentrary syntax and type checking, is followed by formal analysis of the specifications such as to detect and remove logicalinconsistencies. We assume the software design is implemented (or modeled) in Simulink, subsequently, Simulink models are formally analyzed against the specifications. Finally, the software functionality is mapped to the underlying hardware architecture which considers optimizing power consumption while meeting timing and reliability constraints of a distributed safety-critical software, modelin in AUTOSAR.

Concept	Description
System	refers to any physical or logical entity that can process an input and return an output, e.g., the Adjustable Speed Limiter system, Cruise-control system
Parameter	refers to the input and output parameters of a system, e.g., vehicle speed, vehicle speed set by the driver, torque force, etc.
Device	the input and output devices of a system, e.g., sensors and actuators, human-machine interfaces, etc.
State	refers to the operational and dynamic state of a system and its components, e.g., ASL activated, ASL deactivated, ASL enabled, etc.
Mode	refers to the collection of operational capabilities and activities to achieve system objective, e.g.,,ASL mode,Cruise-control mode, etc.
Entity	refers to internal or external occurrences that need to be handled by a system

4.1 ReSA - a Requirements Specification Language

The language is designed to improve comprehensibility and reduce ambiguity of embedded systems specifications. It is a constrained natural language on both syntax and semantic levels. There are fixed syntactic rules to construct valid statements, and embedded system concepts that dectate the semantic construction of statements as shown in Table x, and relations between instances of the concepts.

Boilerplates ....

Examples ...

Boilerplate	Description
Simple	instantiates a simple statement, e,g., `Simple := System Verb within Quantity TimeUnit.`
Compound	instantiates a compound statement, e.g., `Compound := Simple and/or Simple+`
Complex	instantiates complex statement, and is constructed from Simple and a subordinating conjunction, such as when, while, until, e.g., `Complex := if Simple, Simple`
Nested-Complex	instantiates nested-complex statements, and is constructed from independent clause and complex boilerplate(s), e.g., `Ncomplex := after Simple, Complex.`
Conditional	instantiates conditional statements including if, if-else, if-else-if

R1 ASL:system shall send "the driver":user notification:status every 200ms.
if (ASL:system is activated and
(the driver:user presses PAUSEBtn:inDevice or
vehicle:user is not in running:mode))
R2 then
ASL:system shall stop to limit vehicle speed:parameter and
ASL status:status shall be set to enabled
endif
R3 After ASL:system is enabled, if IncButton:inDevice is pressed, ASL:system shall be activated.

4.2 Formal Analysis of ReSA Specifications

The ReSA specifications are transformed into Boolean and propositional logics to conduct rogorous analysis besides syntax and type checking such as consistency checking. The Boolean expressions are check for consistency via a SAT solver, and the description-logic specifications via an ontology enference engine.

4.2.1 SAT-based Analysis

. specifications are translated into Boolean expressions, subsequently, an SMT solver is employed to check the consistency of the specifications. The analysis is scalable as the SAT solving scalable to thousands of propositional variables, though shallow since the details of each clause in the specifications are abstracted by a propositional variable. Thus, advanced analysis is not possible or suitable using the Boolean approach, e.g., the clauses “ASL is activated” and "ASL is deactivated" are represented by two variables and does not cosider the lexical relation between the ‘activated” and “deactivated” words, which in fact are antonyms. Moreover, temporal analysis is not suitable or intractable.

Consider the conditional requirements R2, its translation to a Boolean expression is $(a \wedge (b \vee c)) \rightarrow (d \wedge e)$, where a is “ASL:system is activated”, b the driver:user presses PAUSEBtn:inDevice, c “vehicle:user is not in running:mode”, d “ASL:system shall stop to limit vehicle speed:parameter”, e “ASL status:status shall be set to enabled”.

4.2.2 Ontology-based Analysis

The antonym relation and other lexical relations, e.g., similarity, generality, specialization, etc, can be leveraged to detect complex logical inconsistencies. So, the translation of ReSA specifications to description logic considers the lexical relations of words in statements to realizes the advanced anlalysis.

Essentially, the translated specifications in the description logic constitute a requirments specification ontology, subsequently checked for consistency via an enference engine (o reasoner).

Figure x shows architecture of the requirement specification analysis using the SMT solving and ontology.

4.3 Formal and Scalable Analysis of Simulink Models

A Simulink model consists of functional blocks, which are modeling elements to realize some functionlity, e.g., generate signal, delay a signal, mathematical operations over signals, etc., and lines connecting the blocks. The Simulink block can be either discrete or continuous depending on the block's execution behavior. If the block is executed periodically with a sample time t_s, it is descrete, otherwise, it is continuous if it has no sampling time, rather executes over finitely small (or minor) sampling times. The blocks can also be classified into atomic and composite blocks, where the atomic blocks, e.g., a delay block, is atomic schedulable entity that cannot be decomposed further. Whereas, the composite blocks, e.g., the Subsystem block, contains atomic blocks, and is used to create hierarchy, hence improving the visualization or to impose collective execution behavior over a set of atomic blocks.

Syntax of a Simulink block

Semantics of Simulink block

The Simulink model is transformed into a network of stochastic automata, which is a formal model, using *stochastic timed-automata transformation* patterns, before anlyzing the latter model via a statistical model checking.

4.3.1 Stochastic Timed Automata Transformation Patterns

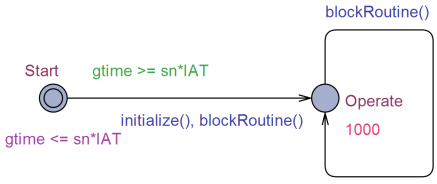
The stochastic timed-automata are continuous and descrete timed automata patters which model the generic execution behavior of the descrete and continuous blocks, respectively, thus providing a generic formalization template of Simuink models.

4.3.2 Simulink Models Transformation

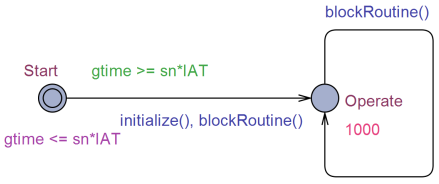
4.4 Fault-tolerant AUTOSAR Allocation

4.5 Scalable Fault-tolerant AUTOSAR Allocation

4.6 Validation On Industrial Use Cases



(a) Continuous.



(b) Discrete.

Figure 4.2: STA transformation pattersn.

5. Research Method

6. Related Work

7. Conclusions and Future Work

Bibliography

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

[2] S. K. Baruah, A. Burns, and R. I. Davis. Response-time analysis for mixed criticality systems. In *Proceedings - Real-Time Systems Symposium*, 2011.

[3] Willem Nico Borst and W N Borst. *Construction of Engineering Ontologies for Knowledge Sharing and Reuse*. PhD thesis, University of Twente, Netherlands, 1997.

[4] 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.

[5] Giorgio C Buttazzo. Hard real-time computing systems: Predictable scheduling algorithms and applications. *Computers & Mathematics with Applications*, 2003.

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

[7] Robert I. Davis, Alan Burns, Reinder J. Bril, and Johan J. Lukkien. Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised, 2007.

[8] Matthew B. Dwyer, George S. Avrunin, and James C. Corbett. Patterns in property specifications for finite-state verification. In *Proceedings of the 21st international conference on Software engineering - ICSE '99*, 1999.

[9] 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.

[10] 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.

[11] Predrag Filipovikj, Nesredin Mahmud, Raluca Marinescu, Guillermo Rodriguez-Navas, Cristina Seceleanu, Oscar Ljungkrantz, and Henrik Lönn. Simppaal - A Framework For Statistical Model Checking of Industrial Simulink Models. *Submitted to Software Engineering and Methodology (TOSEM) Journal*, 2018.

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

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

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

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

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

[17] Hermann Kopetz. Real-time systems: Design principles for distributed embedded applications. *Computers & Mathematics with Applications*, 2003.

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

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

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

[21] 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.

[22] N Mahmud, C Seceleanu, and O Ljungkrantz. Specification and semantic analysis of embedded systems requirements: From description logic to temporal logic. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2017.

[23] 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.

[24] Nesredin Mahmud, Guillermo Rodriguez-Navas, Hamid Reza Faragardi, Saad Mubeen, and Cristina Seceleanu. Power-aware Allocation of Fault-tolerant AUTOSARSystems with End-to-end Timing Constraints viaHybrid Particle Swarm Optimization. Technical report, 2019.

[25] 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.

[26] Sharad Malik and Lintao Zhang. Boolean satisfiability from theoretical hardness to practical success. *Communications of the ACM*, 52(8):76, 2009.

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

[28] Seyedali Mirjalili. Particle swarm optimisation. In *Studies in Computational Intelligence*. 2019.

[29] 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.

[30] 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.

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

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