Design of Assured and Efficient Safety-critical Embedded Systems

Abstract page

Abstract Safety-critical systems should be analyzed rigorously to remove software/specifications errors, i.e., their requirements specifications should be unambiguous, comprehensible and consistent, and the software design should conform to the specifications, hence avoiding undesirable system failures. In particular, analyzing the timing behavior of safety-critical software that is refined by multi-rate periodic tasks, and has data age constraints, i.e., freshness of data, across the input-output path of end-to-end software functionality, is not trivial due to the undersampling and oversampling effects, which results as data propagates from higher to lower rates and vice versa, respectively. Furthermore, when the safety-critical software is deployed on a distributed architecture, e.g., electrical/electronic vehicular system, besides assuring the timeliness, the reliability of the distributed software should be maximized to counter the higher risk of failures in the distributed computing, hence improving the overall predictability of the safety-critical system. However, reliability design usually requires additional critical system resources such as power and energy. Thus, to accommodate the growing complexity of software functionality, the design of the safety-critical systems should consider the efficient use of critical system resources such as power source while meeting the timing and reliability requirements.

In this thesis, we propose formal methods and optimization techniques to assure improved quality of requirements specifications and software design, and to efficiently map software functionality to hardware. Contributions of the thesis are: (i) a constrained requirements specifications language of embedded systems, *ReSA*; (ii) a formal analysis of ReSA specifications via SMT and Ontology; (iii) a statistical model checking of a software design, modeled in Simulink, via transformation to a network of stochastic timed automata; (iv) a resource efficient allocation of fault-tolerant software with end-to-end timing and reliability constraints via integer integer-linear programming and hybrid particle-swarm optimization. Our proposed solutions are evaluated on automotive use cases such as the adjustable-speed limiter (ASL) and the brake-by-wire (BBW) systems from Volvo Group Trucks Technology (VGTT), and on an engine management system benchmark from Bosch.

To My Parents

List of Papers Included in the Thesis

This thesis is based on the following papers:

Paper A ReSA: An ontology-based requirement specification language tailored to automotive systems. Nesredin Mahmud, Cristina Seceleanu and Ljungkrantz Oscar.*In the 10th IEEE International Symposium on Industrial Embedded Systems (SIES)(pp. 1-10). IEEE.*

Paper B ReSA tool: Structured requirements specification and SAT-based consistency-checking. Nesredin Mahmud, Cristina Seceleanu and Ljungkrantz Oscar. *In the 2016 Federated Conference on Computer Science and Information Systems (FedCSIS)(pp. 1737-1746). IEEE.*

Paper C Specification and semantic analysis of embedded systems requirements: From description logic to temporal logic. Nesredin Mahmud, Cristina Seceleanu and Ljungkrantz Oscar. *In the International Conference on Software Engineering and Formal Methods (pp. 332-348). Springer, Cham.* Acceptance rate: 26%.

Paper D SIMPPAAL - A Framework For Statistical Model Checking of Industrial Simulink Models. Predrag Filipovikj, Nesredin Mahmud, Raluca Marinescu, Cristina Seceleanu, Oscar Ljungkrantz and Henrik Lönn. *Submitted to ACM Transaction on Software Engineering and Methodology (TOSEM). ACM Journals.*

Paper E Power-aware Allocation of Fault-tolerant Multirate AUTOSAR Applications. Nesredin Mahmud, Guillermo Rodriguez-Navas, Hamid Faragardi, Saad Mubeen and Cristina Seceleanu. *In the 25th Asia-Pacific Software Engineering Conference (APSEC'18). IEEE. .*

Paper F Optimized Allocation of Fault-tolerant Embedded Software with End-to-end Timing Constraints

Nesredin Mahmud, Guillermo Rodriguez-Navas, Hamid Faragardi, Saad Mubeen and Cristina Seceleanu. Optimized Allocation of Fault-tolerant EmbeddedSoftware with End-to-end Timing Constraints. *Mälardalen Real-time Research Center Technical Report (MRTC).* Submitted to Elsevier JSA Journal.

List of Papers Not Included in the Thesis

- Evaluating industrial applicability of virtualization on a distributed multicore platform. Nesredin Mahmud, Kristian Sandström, and Aneta Vulgarakis. *In Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA), pp. 1-8. IEEE, 2014.*

- The multi-resource server for predictable execution on multi-core platforms. Rafia Inam, Nesredin Mahmud, Moris Behnam, Thomas Nolte, and Mikael Sjödin. *In 2014 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS), pp. 1-12. IEEE, 2014.*

- Simulink to UPPAAL statistical model checker: Analyzing automotive industrial systems. Predrag Filipovikj, Nesredin Mahmud, Raluca Marinescu, Cristina Seceleanu, Oscar Ljungkrantz, and Henrik Lönn. *In International Symposium on Formal Methods, pp. 748-756. Springer, Cham, 2016.*

Reprints were made with permission from the publishers.

Contents

List of Tables

List of Figures

1. Introduction

REAL-TIME systems are usually characterized by timely computations, which are bounded by *deadlines* besides correct results of the computations [14]. They are applied in many *safety-critical embedded* systems, which are specialized computer systems designed for safety-critical applications [72], e.g., brake-by-wire, flight control. For instance, the brake-by-wire system [**?**] applies torque proportional to the pressing of the brake pedal in order to slow down (or halt) the vehicle. Besides computing the correct torque, the system must act timely, i.e., not too soon or not too late so that a traffic accident is avoided. Therefore, safety-critical real-time systems should be analyzed for functional as well as timing correctness, and in fact, they should be analyzed rigorously according to functiona safety standards, e.g., ISO 26262 "Road vehicles-Functional safety" [38] recommends the use of *formal methods*, which are mathematical techniques and tools that enable unambiguous specification, modeling and rigorous analysis [60], to develop safety-critical automotive systems. In this thesis, we apply various formal methods early, i.e., at the requirements specification and software design levels, to improve the quality assurance of developing safety-critical embedded systems.

Over the last decades, a lot of functionality that have been provided by mechanical and electical systems have moved to software functionality, e.g., the x-by-wire technology [**?**], and several software functionality, which are driven by innovation and safety standards, are integrated into safety-critical systems to provide advanced, safe and reliable services, e.g., the advanced driver-assistance systems, vision system for autonomous and self-driving vehicles. In response, several computing architectures have moved from a federated architecture (i.e., dedicated hardware systems executing safety-critial software systems or applications) into consolidated (i.e., multiple safety-critial software applications co-hosted on the same hardware) and distributed architectures. In distributed computing [41], the safety-critical software is mapped on multiple hardware systems to capitalize on the computational power provided by the distributed architecture, e.g., the brake-by-wire software executing on multiple electronic control units (ECU) that are connected through a network bus, e.g., CAN []. The distributed software, unlike in the case of federated computing, it is exposed to a greater degree of permanent and transient faults, therefore, reliability of the software should be maximized to improve the overall dependability [] of the system.

However, such measures requires additional critical system resources besides computational resources, e.g., power and energy, which are constrained in battery-driven embedded systems. Thus, the distributed safety-critical software should be deployed efficiently on the distributed architecture while meeting the timing and reliability requirements of the software.

In this thesis, we propose a design approach which considers rigoruous analysis and efficiency of safety-critical embedded systems by applying formal methods and optimization techniques at the early stages of software development. Thus, the safety-critical specifications should be unambiguous, comprehensible, etc [2], and the software design should conform to the specifications. Since natural languag is inherently ambigous [2], many functional safety standards, e.g., ISO 26262, recommends semi-formal and formal specification methods to specify safety-critical embedded systems requirements, e.g., template-based languages, controlled natural language, temporal logic. The template-based specification methods, e.g., requirements boilerplates [37], property-specification systems [24], etc., lack meta-model to effectively create templates, and is usually cumbersome to select the templates. The controlled natural languages, e.g., Attempto [32][31], etc., renders the syntax and semantics of the natural language and have formal semantics, however lacks support for embedded systems, hence are less effective. Rather, 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 [51] and ontology [11], respectively.

The requirements specifications are used 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 [39]. For this main reason, there is increasing interest in formal analysis of Simulink models [52]. 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 [45]. In contrast, we propose a scalable, timed analysis via a statistical model checking [44], which uses traces of executions and statistical analysis techniques, e.g., monte-carlo simulation.

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

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

consumption of the distributed system to facilitate accommodating complex software functionality. We assume 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) of the vehicle. Furthermore, we consider fault tolerance as a means to maximize reliability of the distributed 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 nodes. Specifically, we propose a formulation of integer-linear programming (ILP) [48], which is solved using branch and bound. Furthermore, we propose a hybrid-particle optimization [56], which is a meta-heuristic algorithm, to solve the shortcomings of the exact method for large-scale problems [49] with trade-off over optimality.

The main contributions of the thesis are: (i) formal analysis of natural language requirements - we propose a fairly expressive, flexible yet structured and domain-specific constrained natural language called *ReSA* [46][50]. The language has semantics in Boolean and description logic to support shallow and rigorous analysis, respectively. The Boolean specifications are checked for consistency using the Boolean satisfiability via the Z3 SMT solver. The ReSA tool is integrated seamlessly into EATOP to complement the requirements modeling of EAST-ADL. The language and its tool support are validated on the adjustable speed limiter (ASL) use case, which is a vehicle speed control system provided by Volvo Group Trucks Technology (VGTT); (ii) scalable and formal analysis of Simulink models - we propose a pattern-based, execution-order preserving automatic transformation of a Simulink model into a network stochastic timed automata that can be formally analyzed using the UPPAAL statistical model checker (SMC) [29]. The statistical model checker analyzes a state-transition system by conducting statistical analysis on the collected traces of the system executions [13], effectively mitigating the state-space explosion of (exact) model checking [44]. Our proposed technique is validated on the brake-by-wire (BBW) use case, which is an industrial prototype provide for academic uses from VGTT; (iii) efficient allocation of distributed safety-critical software applications - we propose an integer-linear programming (ILP) model and hybrid particle swarm optimization algorithms to allocate AUTOSAR software applications on a network of heterogenuous computing nodes with respect to processor speed, failure rate and power consumption specifications. The ILP problem is implemented using ILOG CPLEX, which is a toolset for modeling and solving optimization problems. The proposed allocation methods are validated on an automotive benchmark developed according to AUTOSAR.

1.1 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 on the logic-based reasoning using Boolean and description logic, Simulink, UPPAL statistical model checking, and the optimization techniques based on ILP and PSO. In Chapter 3, we explain the research problem and outline the research goals. The thesis contributions are discussed in Chapter 4, which states the peer-review papers mappings to the contributions. In Section 5, we provide the related work on requirements specification, formal analysis of Simulink models, and software allocation. In Chapter 6, we describe the research method applied to conduct the research. Finally, in Chapter 7, we conclude the thesis, and outline the limitation and discuss possible directions for future work.

2. Background

In this section, we give background on the different concepts and technologies used in this thesis, which include logic-based reasoning Simulink, stochastic timed automata, statistical model checking, integer-linear programming and population-based metaheuristics.

2.1 Logic-based Reasoning

In this thesis, we express requirements specifications in Boolean logic and apply *Boolean satisfiability* (SAT) to detect inconsistencies within the specifications. Furthermore, we use *ontology* to represent the specifications, and conduct rigorous analysis via a linguistic approach.

Boolean Satisfiability

The study of a Boolean formula is generally concerned with the set of truth assignments (assignments of 0 or 1 to each of the variables) that make the formula true. Finding such assignments by answering the simpler question: "does there exist a truth assignment that satisfies the Boolean formula?" is called the Boolean satisfiability problem (SAT), which is proven to be NP complete [23][10]. However, efficient heuristics and approximation algorithms do exist to solve such problems.

SAT problems are usually checked using decision procedures known as SAT solvers. Z3 integrates several background theories (or Satisfiability Module Theories, SMT) and decision procedures based on SAT solvers [21]. It is a well known SMT solver and a theorem prover developed by Microsoft Research which has been used in the formal verification and reasoning of software and hardware systems, mainly due to its strong support for integration to verification tools via its well known APIs written in Java, Phyton, C++, C and .Net.

Z3's input language is an extension of the SMT-LIB2 standard, which are commands used to assert logical statements and instruct the solver to do some action, e.g., to check for satisfiability, the command *(check-sat)* is envoked. Z3 returns *sat* if the formulas are satisfiable. If the formulas are not satisfiable, Z3 returns *unsat*, or if it cannot decide the result, it returns *unknown*. In case of unsat, if the unsat-core option is enabled, Z3 can return the subset of unsatisfiable assertions by envoking the *(get-unsat-core)* command.

Ontology and Description Logic

Ontology is a knowledge representation technique frequently used in the artificial intelligence and other domains to facilitate automated decision making. It is defined as "a formal specification of a shared conceptualization" according to Borst [12]. It employs different modeling entities, e.g., concepts, instances, axioms, which are compared to classes, objects, inheritance in Object-Oriented Progreamming. The ontology is consistent if every axioms and relations (or assertions) hold [53].

Description logic (DL) [5] is a knowledge presentation language, and is the most widely language used to construct ontologies. It is a fragment of first-order logic with many of the core reasoning problems decidable. For this reason, it has several application, e.g., in web semantics (e.g., OWL) [69], artificial intelligence [9], bioinformatics [61], software engineering, natural language processing. Thus, it is usually backed by solid reasoning services that terminate and deliver results in reasonable time.

2.2 Simulink

Simulink [39] is a graphical development environment for the modeling, simulation and analysis of embedded systems which is widely used in industry to model and inspect the dynamics of systems before implementation. It is robust as it supports multi-domain, continuous, discrete, hybrid systems, also discrete systems that execute with different sampling times (or multi-rate). Figure 2.1 shows a multi-rate subsystem of the brake-by-wire Simulink model that models the brake pedal (i.e., continuous behavior) and global brake functionality (i.e., discrete behavior), which executes every 10ms and 20ms.

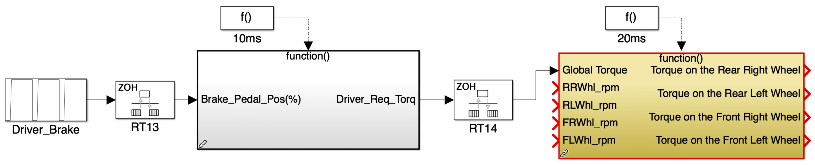


Figure 2.1: Brake pedal and global brake controller of the brake-by-wire model.

Simulink is frequently used in the development of safety-critical systems, e.g., automotive, avionics, furthermore, it is used to generate code automatically from the models. Therefore, it is crucial that such models are analyzed rigorously. The Simulink Design Verifier (SDV) provides a formal verification technique that is based on the exact model checking to exhaustively verify functional properties [54], however, its funtionality is limited as it lacks support for timed anlaysis, which is vital to ensure predictability of safety-critical embedded systems.

Simulink Blocks

A Simulink model is constructed from communicating function blocks (or Simulink blocks). The blocks implement simple to complex functionlity and consist of Input and Output ports, which enable communication via connectors. The latter modeling elements support basic and user-defined datatypes, e.g., integer, floating-point; and simple and complex data structures, e.g., scalar, vector, matrix. Simulink blocks are classified into *virtual* (non-computational, e.g., Mux, Demux blocks) and *non-virtual* (computational, e.g., Gain, Integrator) blocks based on their computability. The non-virtual blocks improve the visualization of the model, but unlike the non-virtual blocks, they are not executed, hence do not affect the execution semantics of the model. The blocks can be composed into groups, e.g., using Subsystem, Model blocks, to enable hierarchical modeling, which is used to improve the visualization, and to enforce execution order on a group of blocks. The fundamental constructs of the composite blocks are *atomic* blocks, e.g., Gain, Sum blocks.

The non-virtual (computational) atomic blocks can be categorized into *continuous* and *discrete* blocks based on the execution semantics of the blocks. A discrete block executes periodically with sample time t_s, whereas a continuous block executes over infinitesimal sample times. Since the Simulink blocks libraries are not usually sufficient to model practical embedded system systems, Simulink supports mechanisms to extend functionality that engineers can exploit to develop complex systems. The mechanisms include S-function, Custom Block and Masking. S-function is a computer language of Simulink blocks which allows advanced implementations of block routines, written in MATLAB, C, C++, or Fortran.

Execution of Simulink Blocks

During the initial phase of the simulation in the Simulink environment, the model is compiled, thus the order in which the blocks are executed is established in the *sorted order* list. Figure 2.2 shows the execution order via the labels in red color, annotated as $s : b$, where s denotes the system/subsystem index and b the block index[1]. Basically, the list is determined according to the data dependency of blocks' outputs on the blocks' input ports, i.e., if the output depends on the current value of the

[1]https://se.mathworks.com/help/simulink/ug/controlling-and-displaying-the-sorted-order.html

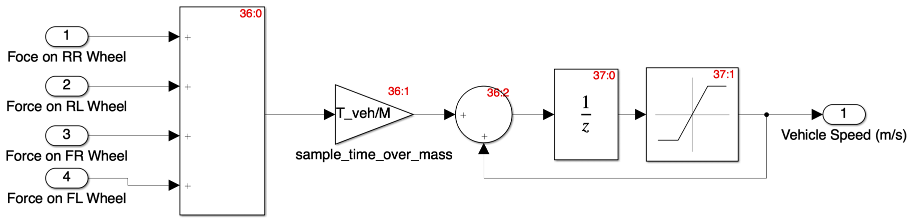


Figure 2.2: A Subsystem block from the brake-by-wire model, computes vehicle speed. The labes in red color are indexed identifiers and denote of the execution order.

input, the input port is identified as *direct-feedthrough* port. Thus, to preserve the data dependency in the model, the sort-order rules require that the blocks that derive other blocks with direct-feedthrough ports must come first in the list, e.g., blocks that derive Gain block. However, the blocks with non-direct-feedthrough ports, e.g., Delay block, can execute in any order, considering the previous rule appies. The execution order is also affected by the user defined priorities, nevertheless, the priorities do not violate the rules. The sorted order list can be fetched by simulating the model in the debugging mode.

2.3 Stochastic Timed Automata

The theory of stochastic timed automata builds upon the timed automata theory via probability to support stochastic behavior in modeling and anlaysis of real-time systems. It can be used to model continuous-time behaviors, e.g., pressing the brake pedal, which occurs at continuous time points, and others, e.g., waiting time in network communication, uncertainly in input values, soft real-time deadlines.

A timed automaton $\mathcal{A}$ is a finite-state automaton proposed by Alur et al. [4] to model and anlayze real-time systems, i.e., by extending the automaton with real-valued clock variables X. It is widely used in the model checking to verify functional and timing requirements, e.g.,liveness, safety properties. The stochastic version automaton $\langle \mathcal{A}, \mu, \gamma \rangle$ basically extends the timed automaton with probability, i.e., on the delays between states via the delay density functions μ and discrete transitions between locations via the output probability functions γ, where $\mathcal{A}$ is represented as a tuple $\langle L, l_0, X, \Sigma, E, I \rangle$, and its elements are described as follows: L is a set of locations of which l_0 is the initial location, E is the set of edges between locations, Σ is a set of action labels, and I is a set of invariants, which are Boolean constraints over X assinged to locations, i.e., the system (or automaton) can only stay in the location as long as its invariants hold. An edge $(l, g, a, r, l') \in E$ denotes a transition relation from the location l to l', with conjuctions of guards g,

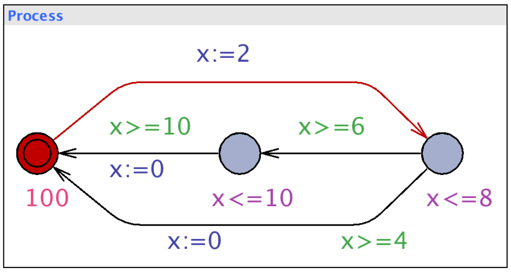


Figure 2.3: STA example.

action a and clock resets r. The gaurd is also a Boolean constraint over X, and when it holds, the transition on the edge is enabled.

Example 1 (STA Example)**.** Figure 2.3 is a dummy STA example with three locations A,B and C, where A is the initial location. The guards of the automaton are labeled in Green color, the clock resets in Blue and the constraints in Red.

Assume that, $d(l,v)$ is the infimum delay which satisfies the disjunction of guards on the edge that start from l, and assume that, $D(l,v)$ is the superimum delay which satisfies the invariant $I(l)$. If the delay is bounded, i.e., there exists $D(l,v)$, in location l, each delay density function in l, μ_s, is assumed to be uniformly distributed on the interval $[d(l,v),D(l,v)]$. However, if the location is time unbounded, the delays are assumed to be exponentially distributed with the rate $R(l)$. To illustate this point, consider the same STA example, the delays at the location A are exponentially distributed, i.e., with rate 100, since the the location is time unbounded, and delays at B and C are uniformly distributed on the interval $[4,8]$ and $[10,10]$, respectively. The output probability of locations A and C is 1 and of location B is 1/2.

Network of Stochastic Timed Automata

Under the assumption of input-enabledness, disjointness of clock sets and output action sets, a network of STA (denoted by NSTA) is parallel composition of STA_i, i.e., $(STA||STA2||,...,||STAn)$, where n is the number of STA in the netowrk. The state of a networked TA is a tuple $\langle s_1,s_2,...,s_n\rangle$, where $s_i = (l,v) \in L_i \times X_i$. In NSTA, the automata communicate through broadcast channels and globally shared variables. Moreover, the semantics of the NSTA relies on the independent computation of the individual STA (or components), i.e., each component automaton, based on the delay density functions and the output probability functions decides repeatedly on which output to generate and at which point in time. In this race, the output will be determined by the component automaton that has chosen to produce the output after the minimum delay [19].

2.4 Statistical Model Checking

Statistical model checking uses a finite set of randomly selected execution traces (or runs) of the automata, and subsequently checks if the traces satisfies a propery φ, i.e., $S \in Runs(A) \models \varphi$. It applies statistical techniques such as hypothesis testing and mote carlo simulation the satisfaction of properties.

UPPAAL SMC is a statistical model checker found in the UPPAAL toolset. It takes in systems that are modeled as a network of stochastic priced timed automata. In this thesis, we use the real-valued clocks X_j that evolve with implicit rate 1, thus the priced automata have only stochastic semantics. In this thesis, the notion of stochastic priced timed automata is used only to model monitor automata (composed in parallel with the actual system model), whic implement the *stop-watch* mechanism.

Properties Specification

UPPAAL SMC employs the probabilistic extension of *weighted metric temporal logic* (WMTL) [?], i.e., PWMTL, to describe quanitative and qualitative statistical properties, which can be inductively constructed from the following grammar:

$$\varphi ::= P(\Diamond_{c \le t} \phi) \bowtie p | P(\Box c \le t) \bowtie p \tag{2.1}$$

where c is the observer clock of the automaton under analysis, ϕ is a state property, $\bowtie \in \{<, \le, =, \ge, >\}$, and $p \in [0, 1]$.

The following types of properties can be checked in UPPAAL SMC:

- *Hypothesis testing* - it is a qualitative method to check if the probability of satisfying a property φ is less than or equal to a certain probabilistic threshold p over a network of STA.

- *Probability evaluation* - it is quantitative method to determine the probability of satisfying a certain property φ over a network of STA.

- *Probability comparison* - it is qualitative method to compare the probabilities of satisfying two properties φ_1 and φ_2 over a network of STA.

2.5 Integer-linear Programming

ILP is an optimization problem where the decision variables are integer, and the objective function and the constraints are linear models. They are problems that can be represented as

$$Maximize \sum_{j=1}^{n} c_j x_j \tag{2.2}$$

$$\text{Subjec to:} \tag{2.3}$$

$$\sum_{j=1}^{n} a_i j x_j \leq b_i \qquad \text{for all } i = 1, ..., m \tag{2.4}$$

$$x_j \geq 0 \wedge x_i \in \mathbb{I} \qquad \text{for all } i = 1, ..., m \tag{2.5}$$

It has many mathematical, and engineering applications, e.g., transportation, scheduling. The binary (0-1) problem is a special case of ILP where the decision variables takes on 0 or 1. It applied in several decision making problems, e.g., resource allocation in the scheduling/ assignment problem. The software mapping is an instance the assignment problem where each binary variable denote if a software component is mapped to a processor (or computing unit) or not, i.e., 1 if assigned otherwise 0.

ILP problems are frequently solved via exact algorithms, e.g., branch and bound, but can also be delt via heuristics, e.g., using simulated annealing, hill climbing. In this work, we use the CPLEX solver form IBM to the software-to-hardware mapping optimization.

2.6 Population-based Metaheuristics

For complex optimization problems, the ILP formulation is a difficult task and the scalability to solve large problems instances is usually prohibitively expensive. Metaheuristics [1][34] is a type of heuristics which uses search strategies that are sometimes less dependent on the problems and more efficient computation-wise but less optimal. Population-based metaheuristics is a class of meta-heuristic methods which uses a set of individuals (or population) in the search strategy to determine the global optima of the problem, e.g.,differential evolution [70][18], particle swarm optimization [3][66].

In this work, we apply differential evolution, particle-swarm optimization to search for the best software-to-hardware mapping solution. The differential evolution employees a set of agents, which represent candidate solutions, to find the best software-to-hardware mapping solution. Each agent applies a particular types of evolutionary operators such as mutation, crossover and selection to reach a target position in the search space. A mutant agent (or offspring) is generated for every agent and in every iteration from three other angents. If the offspring is more fit, i.e., better than the parent agent, it replaces the latter, otherwise is discarded.

$$\mathbf{v} \leftarrow \mathbf{a} + F \circ (\mathbf{b} - \mathbf{c}) \tag{2.6}$$

$$\mathbf{u} \leftarrow crossOver(\mathbf{v}, \mathbf{x}, CF, F) \tag{2.7}$$

$$\mathbf{x} \leftarrow \begin{cases} \mathbf{u} & \text{if } f(\mathbf{u}) < f(\mathbf{x}) \text{ functions} \\ \mathbf{x} & \text{otherwise} \end{cases} \tag{2.8}$$

where $F \in [0,2]$ is the differential weight, $CF \in [0,1]$ is the crossover probability.

Likewise, in the particle-swarm optimization, the method employs agents (in this case, particles), which are member of the population and represents candidate solutions. It records the best position of each particle so far $\mathbf{p}_{bst}$ and best position of the population $\mathbf{z}$. Thus, the motion of each particle is guided by its velocity, attraction towards its best position $\mathbf{p}_{bst} - \mathbf{p}$ and attraction towards the best position of the swarm $\mathbf{z} - \mathbf{p}$, where $\mathbf{p}$ is a n-dimensional matrix which represents the current position of a particle in the search space.

$$\mathbf{v} \leftarrow \omega \mathbf{v} + c_1 Rand() \circ (\mathbf{p}_{bst} - \mathbf{p}) + c_2 Rand() \circ (\mathbf{z} - \mathbf{p}) \tag{2.9}$$

$$\mathbf{p} \leftarrow \mathbf{p} + \mathbf{v}, \tag{2.10}$$

where ω is the weight of the velocity, also known as *inertia coefficient* and controls the convergence of the algorithm. The c_1, c_2 constants are acceleration coefficients and control the weight of attraction towards the cognitive and social components, respectively. $Rand() \in U(0,1)$ is a random function along the acceleration coefficients, which is element-wise multiplied with the components to improve diversity of the search by introducing stochastic behavior

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 [26][50]), controlled natural languages [43](e.g., Attempto [32]).

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 [43]. 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 [2][58], 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 [16].

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 [16]. 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 [45]. 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 [39]. 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 [7][20], moreover end-to-end delay analysis based on the age delay semantics [57], 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. Thesis 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 allocation of software to hardware while satisfying timing and reliability of the software in a distributed environment. It satisfies the research goals explained in Subsection 3.1 via the following contributions: a requirements specification language tailored to embedded systems [50][46], formal analysis of the specifications [46][47] via formal and knowledge-based methods, formal analysis of large-scale Simulink models [30] and efficient mapping of software to hardware in the context of distributed architecture [48][49] via integer-linear programming and metaheuristics.

4.1 Design Workflow: Contributions Overview

Figure 4.1 illustrates the workflow of an embedded software development that make of use our contributions. The workflow is contextualized in the automotive software development where Simulink, EAST-ADL and AUTOSAR architectural language are used.

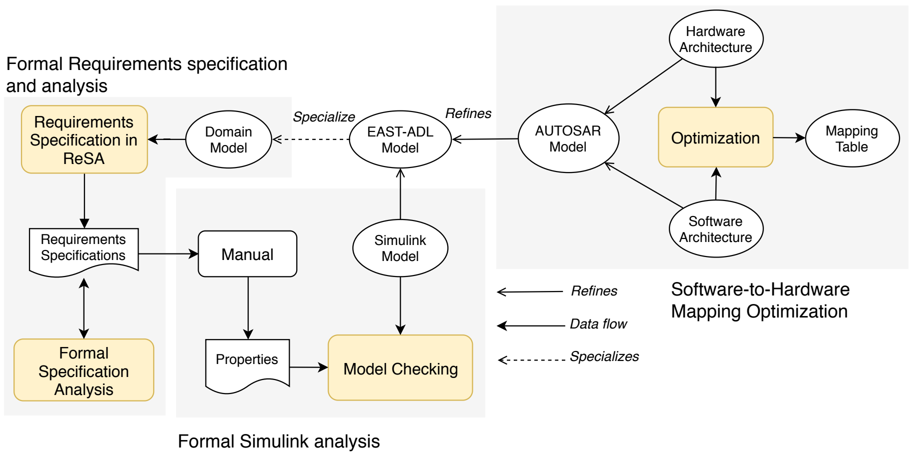


Figure 4.1: Thesis contributions workflow.

Initially the textual requirements are specified in our domain-specific language ReSA, which is a constrained natural language tailored to embedded systems. The ReSA editor supports content completion as it is connected to a system model, which contains instantiation of words/phrases. If the ReSA editor is required to be domain-specific, e.g., to the automotive domain that employs EAST-ADL, the system model is specialized to the later model, thus enabling requirements specifications by accessing ESAST-ADL model elements from the editor. To check the consistency, the ReSA specifications are translated into Boolean and description logic, subsequently, analyzed via SAT solver and inference engine (or reasoner), respectively.

After the quality of the specifications are improved, i.e., become unambiguous, comprehensible, consistent, they can be used in the verification of behavioral software designs. In this thesis, we consider the software design is made via Simulink, which is one of the most widely integerated environment for modeling, simulation and analysis of embedded system models. In order to rigorously analyze a Simulink model, we transform it automatically into a formal model, i.e., a network of stochastic timed automata. Subsequently, the latter is analyzed via statistical model checking against properties initially expressed in ReSA, but manually translated into the query (properties) language of the model checker, i.e., probabilistic weighed-metric temporal logic (PWMTL).

The Simulink model, at the architectural level, is represented by an EAST-ADL software architecture at the implementation level, i.e., via an AUTOSAR software architecture, which consists of software components that communicate through virtual function bus (VBS). Note that, the refinement from the AUTOSAR model to the Simulink model is manual, which is the normal practice, however, the Simulink model composition (or structure) is automatically generated. The software architecture complements the Simiulink model with computational resource specifications, i.e., via runnables, which are schedulable piece of code (or objects) by the AUTOSAR OS. Moreover, it complements with other resource specifications, e.g., power specification, interface specification of the underlaying hardwares. In this regard, we propose (near) optimal software to hardware optimization techniques that incorporate the timing and reliability of software applications as constraints, and optimizes the power consumption of the software.

The thesis contributions are summarized as follows:

4.2 Contr. 1: The ReSA Language

We propose a domain-specific and constrained natural language, both at the syntax and semantics level, which is designed to improve comprehensibility and reduce ambiguity of embedded systems requirement specifications. The language employs concepts from the embedded systems domain, e.g., *System*, *Parameter*, *Device*, *State*, *Mode*, *Entity*, to typeset domain-specific words/phrases. Besides grammatical rules (or syntax), semantic relations between the concepts limit on the possible construction of the specifications, e.g., grammatically the "The ASL shall limit the driver." specification is correct, but semantically, it makes no sense. To solve this problem, the ReSA type system imposes on the a type constraint on the relation between instances "The ASL"*:system* and "the driver"*:user*.

Syntax

The language design is modular, i.e., it uses sentence catagories, e.g., *Simple*, *Compound*, to structure the specifications, thus improve comprehensibility and reduce ambiguity. It also allow complex specifications by inductively applying the grammer rules of the language. The following context-free grammar is a snippet of the ReSA language.

$$\begin{array}{rcl}
\langle\text{specification}\rangle & \models & \langle\text{simple}\rangle \mid \langle\text{complex}\rangle \mid \langle\text{nested-complex}\rangle... \\
\langle\text{simple}\rangle & \models & \langle\text{mainClause}\rangle. \\
\langle\text{complex}\rangle & \models & \langle\text{subClause}\rangle, \langle\text{mainClause}\rangle. \\
\langle\text{mainClause}\rangle & \models & \langle\text{primitiveClause}\rangle \mid ... \\
\langle\text{primitiveClause}\rangle & \models & \langle\text{sub}\rangle\langle\text{verb}\rangle\langle\text{obj}\rangle \mid ... \\
\vdots & &
\end{array}$$

Note: please checkout the ReSA documentation for the complete grammar and its tool support from its webpage located at the URL https://bitbucket.org/nasmdh/resa/src/master/.

Example 2 (Adjustable Speed Limiter (ASL) Requirements)**.** ASL is a speed control system found in Volvo trucks. It controls the vehicle speed to not exceed a certain speed, which is set by the driver or legal authorities. It consists around 300 fuctional and extra-functional requirements. Just to demo the usage of the language, we specify only four of ASL equirements.

R1 ASL:system shall send "the driver"*:user* notification:status every 200ms.

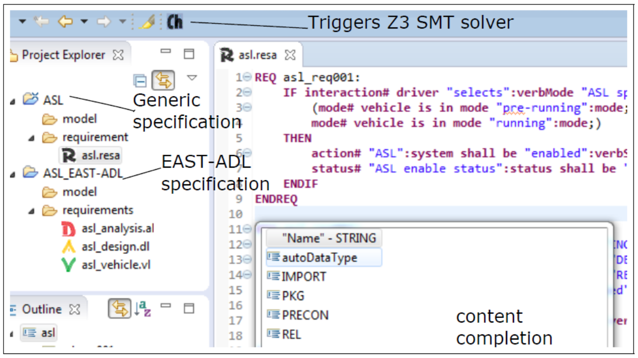
R2 if "driver:user" selects "ASL speed control"*:mode* and
    (vehicle is in "pre-running" mode or vehicle is in "running" mode)
  then
    ASL:system shall be enabled within 200ms and
    "ASL enabled"*:status* shall be presented to "the driver"*:user*
  endif

R3 After ASL:system is enabled, if IncButton*:inDevice* is pressed, ASL:system shall be activated.

R4 if driver selects "ASL speed control"*:mode* then ASL*:system* shall be disabled.

Tool Support and Validation on ASL

The language is implemented in the Xtext framework[1], which is an Eclibse-based integrated development environment for the development of domain-specific and general-purpose languages. The ReSA editor, which is shown in Figure 4.2, supports content completion, errors/warring messages and boilerplate management. The editor is integrated to EATOP, i.e., the editor can be trigged from within the IDE, moreover, the content-completion feature dispalys contents of the EAST-ADL model elements, hence enabling consistent use of vocabularies during specification. The ReSA tool as a standalone, and as a plugin for EATOP, can be downloaded from the URL `https://bitbucket.org/dashboard/overview`.



```
REQ asl_req001:
    IF interaction# driver "selects":verbMode "ASL sp
        (mode# vehicle is in mode "pre-running":mode;
        mode# vehicle is in mode "running":mode;)
    THEN
        action# "ASL":system shall be "enabled":verb
        status# "ASL enable status":status shall be '
    ENDIF
ENDREQ
```

```
"Name" - STRING
autoDataType
IMPORT
PKG
PRECON
REL
```

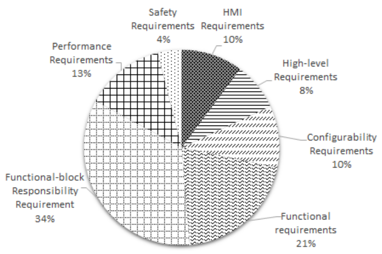
Figure 4.2: The ReSA graphical user interface (GUI).

The language as well as its tool support is validated on the 300 ASL requirements, which are initially expressed in natural lanugage. The requirements distribution is as shown in Figure 4.3a, i.e., in terms of the types of requirments. The requirements are rewritten in ReSA, and Figure 4.3bshows the sentence catagories (or boilerplates) employed to specify the requirments. In general, the language is expressive, though is constrained., and over a period of time, more and more constructs are added to improve the expressiveness.

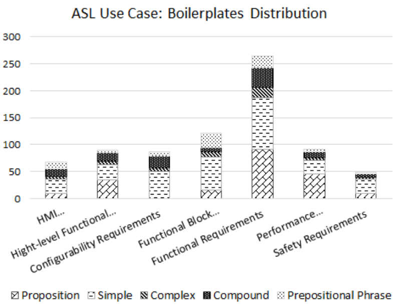
4.3 Contr. 2: Formal Analysis of ReSA Specifications

The editor shows syntax and typing warning and error messages. However, such static analysis are rudimentary since the analysis per specification. In this contribution, we propose two methods of consistency analysis: *SAT-based* anaysis and *ontology-based* analysis.

[1]`Xtext:https://www.eclipse.org/Xtext/`



(a) Types of ASL Requirements.



(b) Distribution of ASL boilerplate types.

Figure 4.3: Requirements specification ontology (screenshot from Protégé tool).

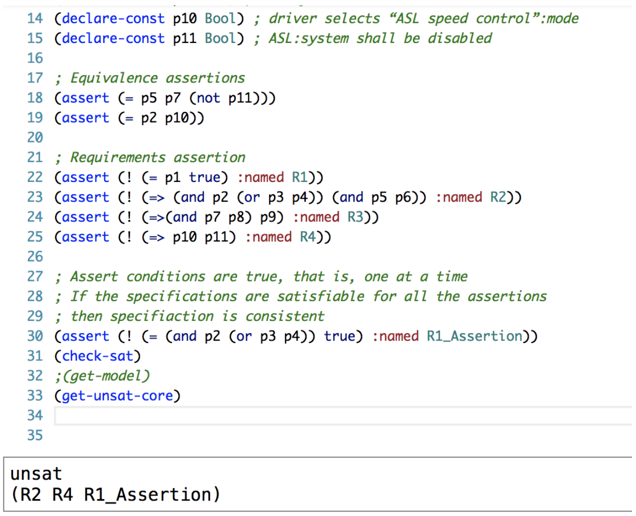
SAT-based Analysis

Boolean satisfiability (also known as SAT) can informally defined as the problem of finding truth-values (or assignments of boolean variables) such that a Boolean formula holds. It is NP-complete, however, there exist efficient heuristic algorithm which solve SAT problems of large-size, i.e., that consist of thousands of Boolean variables. Thus, first we transform the ReSA specifications into Boolean expressions [46], then we formulate the problem of finding inconsistency the Boolean expressions in terms of SAT as follows:

Definition 4.3.1 (Inconsistency of Specifications)**.** Let $\Phi = \{\Phi_1, \Phi_2, \dots, \Phi_n\}$ is the ReSA specifications after translating ReSA specifications into Boolean expressions, where $\Phi \in N$ is a Boolean formula and denotes a requirement specification. Thus, the set Boolean expressions Φ is inconsistent if the expression $\Phi_1 \wedge \Phi_2 \wedge \cdots \wedge \Phi_n \Rightarrow false$ holds, that is, there exists at least one expression that cannot be satisfied, i.e., $\Phi_i \models false$.

Example 3. Assume that we want to check the consistency of the requirements that are specified in Example 2. Figure 4.4 shows the translation of the specifications into Boolean expressions (i.e. in Z3 SMT-LIB format). Each expression is asserted before the Z3 solver is called using the command $(check - sat)$. Some clauses are resolved for negation and opposite words, e.g., the fact that “enabled” is opposite of “disabled” implies $p_5 = p_7 \neq p_{11}$. The solver returned “unsat (R2 R4 R1_Assertion)”, which means, the expressions are inconsistent and indicates the specifications that are responsible for the inconsistency (or unsat-core).

The SAT-based analysis, although, easy and scalable, it does not allow rigorously analysis. This is due to the fact that the Boolean (or propositional) variables abstract away details of the clauses in the specifications. The ontology-based approach solves this problem with trade off over scalability.



```
(declare-const p10 Bool) ; driver selects “ASL speed control”:mode
(declare-const p11 Bool) ; ASL:system shall be disabled

; Equivalence assertions
(assert (= p5 p7 (not p11)))
(assert (= p2 p10))

; Requirements assertion
(assert (! (= p1 true) :named R1))
(assert (! (=> (and p2 (or p3 p4)) (and p5 p6)) :named R2))
(assert (! (=>(and p7 p8) p9) :named R3))
(assert (! (=> p10 p11) :named R4))

; Assert conditions are true, that is, one at a time
; If the specifications are satisfiable for all the assertions
; then specifiaction is consistent
(assert (! (= (and p2 (or p3 p4)) true) :named R1_Assertion))
(check-sat)
;(get-model)
(get-unsat-core)


```

```
unsat
(R2 R4 R1_Assertion)
```

Figure 4.4: The ReSA specifications R1- R4 encoded as Z3 format, and the unsat-core feedback from the Z3 solver, to localize source of the inconsistency.

Ontology-based Analysis

Ontology as a knowledge representation technique can be used to capture the intricate relations of words/phrases of specifications, e.g., synonyms, antonyms, generalization. Morover, it can be used to capture the syntax of the specifications as well, essentially modeling the the ReSA specification as a concrete requirements specification ontology.

Besides the challenge of constructing the ontology, accurate representation of the specifiactions is crucial to a certain degree for the ontology to be useful, i.e., the specifications should be interpreted linguistically. In this research, we propose the *event-based* semantics approach [47] to construct the meaning of each specification compositionally fromts its constituents (or grammar units). The event-based semantics, which is based on the first-order logic, uses an existentially quantified events to relate words/phrases (or arguments of predicate) , clauses, adjuncts of a sentence [47], e.g., the clause “ASL:system shall limit “vehicle speed”:parameter” is represented as $\exists e$[limiting(e) & Agent(ASL) & Recipient(vehicle speed)], where Agent and Recipient are known as *thematic* roles, which define the semantic roles of the arguments in the clause. Since the requirement specifications entail technical words/phrase, the thematic roles of these arguments are not obvious, so we extended the thematic roles into the ReSA concepts, e.g., any instance of *System* or *User* is *Agent*, similarly instances of *Parameter* is *Recipient*. In this way, the interpretation of the clauses become domain-specific.

Following the event-based semantics, the ReSA specifications are translated into description logic as ontology. Once the ontology is constructed considering the event-based semantic, thematic roles, lexical relation, we check the consistency of it via an inference engine (or reasoner), which is a software program that applies logical rules on a logical system, such as the ontology, to obtain new information.

Definition 4.3.2 (Inconsistency of Specifications)**.** The requirements specification ontology is inconsistent if there does not exists an interpretation (or a model) M that satisfies the terminological assertions T and the concrete assertions A, that is, $M \not\models ax$, where: $ax \in T \cup A$, and T is a set of terminological assertions and A a set of facts (or concrete assertions).

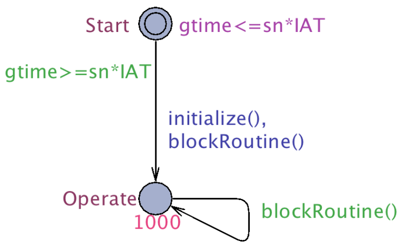
4.4 Contr. 3: Scalable Formal Analysis of Simulink Models

Several research exist on formal anlaysis of Simulink models, e.g., using theorem proving, exact model checking, statistical model checking. Many of the proposed solutions are limited either due to frequent user involvement or scalability issues. In this thesis, we propse a scalable formal analysis of larege-scale Simulink models via statistical model checking. The Simulink models can be discrete, continuous or hybrid, moreover, they can consist of atomic and composite Simulink blocks that are scattered on multiple files, which is a practical scenario. Essentially, we consider typical industrial Simulink models, e.g., the brake-by-wire and adjustable speed limiter Simulink models from VGTT [30].

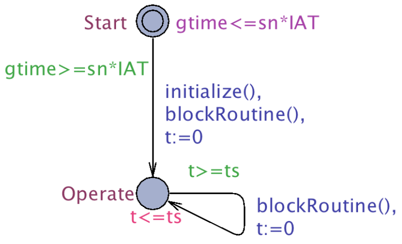
Our approach has the following important characterstics: (i) the transformation employs continous and discrete transformation patterns, which are generic and reusable, thus can be applied to any Simulink blocks; (ii) the transformation preserves the execution orders of the Simulink blocks; (iii) it is robust, i.e., it handles various models such as continuos, discrete, hybrid, and models which contain blocks that execute with different sampling rates, also known as multi-rate.

Transformation Patterns

The continuous-time and discrete-time stochastic timed automata (STA), which are shown in Figure 4.5, are used to transform the continuous-time and discrete-time Simulink blocks into their respective STAs. The automata in both cases makes transition from the "Start" location to the "Operate" location at the global time $sn * IAT$, which is determined according to each block's order of execution sn, i.e., the lower sn, the sooner the automaton makes transition to the "Operate" location, and IAT is an infinitesimal inter-arrival time between the release of the automata. In the case of the continuous-time STA pattern, the automaton makes a loop transition at the "Operate" location every infinitesimal sample time, which is distributed exponentially according to the rate $\lambda = 1000$. Whereas in the case of discrete-time STA pattern, it makes a loop transition every ts sample time with probability of 1, as there is only one edge that goes out of the location and the delay transition is distributed uniformly in the interval $[ts, ts]$.



(a) Continuous-time.



(b) Discrete-time.

Figure 4.5: STA transformation patterns.

Simulink to NSTA Transformation Process

Besides proposing the transformation patterns, we provide an automatic transformation of a Simulink model to its equivalent network of STA by applying the following tasks: (i) the Simulink model is simulated, subsequently, the sorted-order list, which contains the execution order of the Simulink blocks, is extracted. (ii) in case the model is hierarchical, the execution order is flattened first, thus generating a flattened sorted-order list; (iii) the Simulink model is parsed, as a result, each block is translated into a stochastic timed automata via the corresponding transformation pattern. The sorted-order number *sn* is instantiated from the sorted-order list on the fly. (iv) any non-computational blocks is accounted in the transformation but are not trasformed into automata, e.g., the Mux, SubSystem blocks; (v) the connections between the blocks are translated into global variables, which acts as communication between the automata. Please checkout Section 3 of [30] for the detailed discussion on the transformation as well as its soundness proof.

Validation on the Brake-by-wire Use Case

Our approach is applied on an industrial brake-by-wire Simulink model, which is a prototype developed for academic use by VGTT. The model consists of 320 blocks, of which 19 are discrete blocks, 26 constant blocks and the rest are continuous-blocks. The model design consists of of modules that collects sensory data, e.g., vehicle speed, pedal position, and computes the proportional torque force that is applied on the wheels to brake the vehicle. Besides, it has the anti-braking functionality (i.e., the ABS) to avoid (or minimize) skidding, which occurs when the slip rate of a wheel is greater than its friction coefficient.

The BBW requirements are initially specified in ReSA, subsequently are translated into PWMTL properties manually by using monitors (or observers), which are stopwatch stochastic priced timed automata. The monitors are modeled for particular requirements, and basically observe the progressionf the execution of the automata.

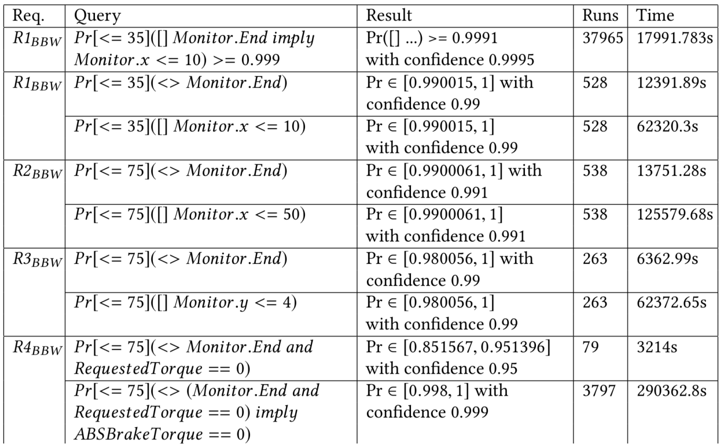
Example 4 (Statistical model checking of BBW)**.** The BBW model is automatically transformed into a network of STAs, and we modeled the monitor (Appendix A) to represent 5 functional and timing requirements, of which 3 requirements are shown in the box below for illustration.

R1	The brake request shall be computed within 10 ms.
R3	The brake request shall be be propagated to two different wheel actuators within 4 ms.
R4	If the brake request is 0, then the ABS shall set the torque to 0.

Figure 4.6 shows the PWMTL propeties and their model checking results. R1 and R3 are safety requirements which are expressed using the $\Box$ operator as indicated by R1$_{BBW}$ and R3$_{BBW}$ PWMTL properties, respectively. R4 is a conditional requirement which is expressed using the $\Diamond$ operator as indicated by R4$_{BBW}$ PWMTL property. Each result shows the probability interval of satisfying the property (if qualitative) or the probability of satisfying a property (if quantitatiy), confidence level, the number of runs and the time needed by the model checker to return the result. For all properties, the probability interval (or confidence interval) approximates $[0.99, 1]$ with confidence level ≈ 0.999, which can be interpreted as "the properties are satisfied" if the probability requirements lower-bound are in the degree of 0.99. However, in many safety-critical applications, the requirement on the probability is normally high, e.g., 0.999999 (essentially approximates to "no failure should happen"). In this case, the interval should be more precise, which can be achieved by lowering α and β of the hypothesis testing, which are the probabiliy of accepting false positive (i.e., accepting H_0 when H_1 holds) and false negative (i.e., accepting H_1 when H_0 holds), respectively [19].

4.5 Contr. 4: ILP-based Software Allocation

Safety-critical systems are usually resource constrained, that is, the execution hardware platform has limited computational resources and power/energy sources. Therefore, the system should be designed efficiently especially



Req.	Query	Result	Runs	Time
$R1_{BBW}$	$Pr[<= 35]([]\ Monitor.End\ imply\ Monitor.x <= 10) >= 0.999$	Pr([] ...) >= 0.9991 with confidence 0.9995	37965	17991.783s
$R1_{BBW}$	$Pr[<= 35](<>\ Monitor.End)$	Pr ∈ [0.990015, 1] with confidence 0.99	528	12391.89s
	$Pr[<= 35]([]\ Monitor.x <= 10)$	Pr ∈ [0.990015, 1] with confidence 0.99	528	62320.3s
$R2_{BBW}$	$Pr[<= 75](<>\ Monitor.End)$	Pr ∈ [0.9900061, 1] with confidence 0.991	538	13751.28s
	$Pr[<= 75]([]\ Monitor.x <= 50)$	Pr ∈ [0.9900061, 1] with confidence 0.991	538	125579.68s
$R3_{BBW}$	$Pr[<= 75](<>\ Monitor.End)$	Pr ∈ [0.980056, 1] with confidence 0.99	263	6362.99s
	$Pr[<= 75]([]\ Monitor.y <= 4)$	Pr ∈ [0.980056, 1] with confidence 0.99	263	62372.65s
$R4_{BBW}$	$Pr[<= 75](<>\ Monitor.End\ and\ RequestedTorque == 0)$	Pr ∈ [0.851567, 0.951396] with confidence 0.95	79	3214s
	$Pr[<= 75](<>\ (Monitor.End\ and\ RequestedTorque == 0)\ imply\ ABSBrakeTorque == 0)$	Pr ∈ [0.998, 1] with confidence 0.999	3797	290362.8s

Figure 4.6: BBW SMC result snippet from UPPAAL SMC.

critical system resources such as power in order to accommodate complex safety-critical software functionality. In this section, we propose exact method of mapping software to hardware in order to optimize the total power consumption of a distributed safety-critical software via ILP, which is needed when highly-efficient design is needed especially small-scale sofware applications, e.g., software components not exceeding 10.

System Model

We consider an AUTOSAR distributed safety-critical software that can be mapped to multiple computing nodes (or units), which are heterogeneous computing systems with respect to power specification, failure rate and processing speed. The software has end-to-end timing and reliability requirements, therefore, the mapping must satisfy the requirements. In order to meet the reliability requirement, the mapping applies fault tolerance by replicating software components subsequently mapping the replicas on different computing units. Moreover, the task and messages that realize the safety-critical software functionality are scheduled using a fixed-priority preemptive scheduling policy and fixed-priority non-preemptive scheduling policy (over a CAN bus), respectively.

The AUTOSAR software functionality is implemented by Runnables that communicate over the virtual function bus (VFB), which abstracts the details of the run-time environment. We assume the runnables are triggered periodically and have multiple worst-case execution times as they can executed on processors that have different processing speed.

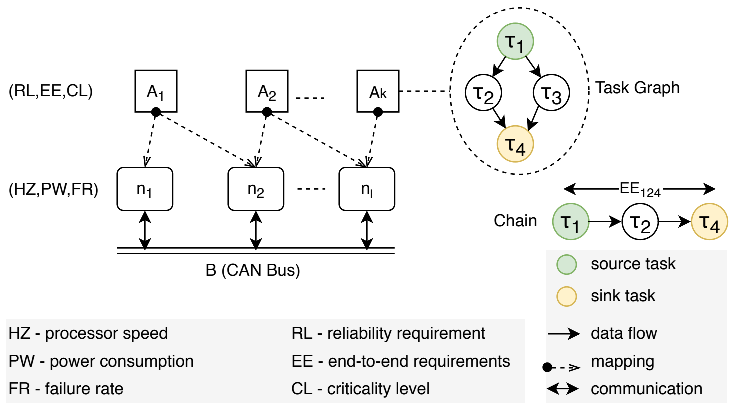


Figure 4.7: Computation time of the various algorithms for solving different instances of the software allocation problem.

ILP Model

Consider that the mapping solution is represented by a vector of binary matrices $\mathbf{x} = \{\mathbf{x}_1 : 1,...,K\}$, where x^k_{ij}represents the mapping of the software component $q^{A_k}_{ij}$ to the computing node $n_j \in \mathcal{N}$.

$$\mathbf{x}^k = \begin{bmatrix} x^k_{11} & x^k_{12} & \dots & x^k_{1K} \\ x^k_{21} & x^k_{22} & \dots & x^k_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ x^k_{N1} & x^k_{N2} & \dots & x^k_{NK} \end{bmatrix} \tag{4.1}$$

The power consumption of a software application $\{c_i : i = 1,...,n_C\}$ that is mapped to computing units $N' \in N$ is calculated as the sum of the power consumption of each node $n_h \in N'$, thus $\mathcal{P}_{total}(x) = \sum_{n \in N'} \mathcal{P}(u_n(x))$, where $\mathcal{P}(u)$ is the power consumption of a computing node which linearly proportional to its utilization [48]. The utilization of a computing node is the sum of the utilization of its constituent tasks that realize corresponding software components.

$$(u_1,...,u_K)(\mathbf{x}) = \sum_{k=1}^{K} \sum_{\tau \in T_{c_i}} x^k_{ij} * \frac{WCET_\tau}{P_\tau}, \tag{4.2}$$

where T_{c_i} is the set of tasks realizing the software component c_i, $WCET_\tau$ and P_τ are the worst-case execution time and period of the task τ.

4.5.0.1 Tasks Deadline Constraint

The mapping solution must satisfy the timing and reliability requirements imposed on the safety-critiacal application. We assume the taskset in each computing node is scheduled using a fixed-priority preemptive scheduler [68] and we check the schedulability via the classical worst-case response time analysis [7]. Since the time analysis model involves recursion which is difficult to model as a linear model. Therefore, we construct linear logical constraints ξ that corresponds to the valid tasksets that can potentially partitioned to computing nodes, and these tasksets are schedulable.

The logical system is constructed as follows: first we identify the possible partitions of the software components, $P = 2^C$, then we check the schedulability of each partition and catagorize them per node, i.e., $Y_j = \{p \in P | isSched(p, n_j)\}$, where *sched* is a boolean function that returns true if p is schedulable on n_j. A partition is identified by a sequence of 0s and 1s based on its costituent components, e.g., the id of the partition $\{c_1, c_4, c_5\} \subseteq \{c_1, c_2, c_3, c_4, c_5\}$ is $10011 = 19$. So, given the mapping $\mathbf{x}$, the partition id that is mapped to the computing node n_j can be computed as

$$g_j(\mathbf{x}) = \sum_{i=1}^{N} 10^{N-i} \max_{1 \leq k < K} x_{ij}^k \tag{4.3}$$

where the $\max_{1 \leq k < K}$ function returns 1 if at least one component replica is mapped n_j, otherwise returns 0, indicating no replica is mapped to that node. Thus, each node must have a valid partition that is schedulable as indicated in the logical constraint,

$$\bigvee_{p \in Y_j} g_j(\mathbf{x}) = id(p), \tag{4.4}$$

where $id(p)$ is a predicate that returns the id of the partition p.

4.5.0.2 End-to-end Timing Constraint

The age delay of a chain is computed according to the semantics discussed in [27][57]. The response-time logical assertions ensure that every task in the distributed system are schedulable for a mapping $\mathbf{x}$. Therefore, we can safely check if the age delay of each chain satisfy its end-to-end timing requirement without the need to check the chain's constituent tasks are schedulable or not. Similar to the previous approach, we construct logical constraints η that correspond to the valid set of mappings of chains on the network of computing nodes. Given $\Gamma = \{\Gamma_i : i = 1, ..., N\}$ of chains, the set of possible mappings of each chain Γ_i on a set of $\mathcal{N}^\Gamma$. Thus the set of valid mappings of each chain satisfy the end-to-end timing requirement is $Z_i = \{\gamma \in \mathcal{N}^\Gamma | ageDelay(\gamma) < EE_\gamma$.

4.5.0.3 Software Application Reliability Constraint

The reliability constraint refers to the probability that the application functions by the time t, i.e $[0, t]$ [33]. We assume that the reliability requirement of a software application is 0.99999999 over a 2-year operation time, which is to say that the safety-critical system almost should not fail within the stated period. Furthermore, we assume that the mean-time to failure of the computing nodes is given as 10^6 hours (0.0000001 per hour), which is the usual practice in many functional safety design.

Unless the the software application is replicated on one or more computing nodes, the reliability requirement is usually unattainable. So, the allocation strategy is basically to replicate sufficient software components to meet the requirement at the same time it should not exceed the limit so that the application consumes less power. However, the reliability calculation is not trivial as the replication results in functional inter-dependency of the computing nodes, in this case, the series-parallel method does not apply. For this reason, we propose an exact method based on state enumeration [**?**], which basically enumerates all the possible configurations of the nodes PS. Subsequently, the reliability of the application becomes the total probability of the configurations that enable the functioning of the application [48].

Definition 4.5.1 (Software Application Failure)**.** A software application fails in the configuration $s \in \xi$ if there exists a component type c_i where all of its replicas Q_i *fail*, otherwise, it functions, as shown in Equation (4.5). The component replica $q_i, j \in Q_i$ of type c_i fails if n_h fails, that is $s_h = 0$.

$$f_s(x) = \left\lfloor \frac{\sum_i f_{c_i}(x,s)}{N} \right\rfloor = \begin{cases} 1 & \text{if } application \text{ functions} \\ 0 & \text{if } application \text{ fails} \end{cases} \tag{4.5}$$

Thus, the reliability of the software allocation is

$$Reliability(\mathbf{x}) = \sum_{s \in PS} f_s(\mathbf{x}) * p_s, \tag{4.6}$$

where p_s is the probability that the computing nodes attain a configuration s which is computed as $p_s = \prod_{m \in M} \left[(z_m * \lambda_m) + (1 - z_m) * (1 - \lambda_m)\right]$, where $z_m \in \{0,1\}$ indicates if the node fails (i.e., 0) or functions (i.e., 1), λ_m is the failure rate of the node m and $1 - \lambda_m$ is the opposite of failure rate, i.e., that is the rate that it does not fail.

4.5.0.4 ILP Problem

The goal of the ILP optimization is to minimize the total power consumption, which is the objective function, and the constraints Equation (4.8), (4.9) and (4.10)ensure that the optimization satisfies the tasks deadline, end-to-end timing and reliability constraints, respectively.

$$\min_{\mathbf{x}\in X} \mathcal{P}_{total}(\mathbf{x}) \qquad \text{Subjected to:} \tag{4.7}$$

$$\bigvee_{p\in Y_j} g_j(\mathbf{x}) = id(p) \qquad \text{for all } j = 1,..,|\mathcal{N}| \tag{4.8}$$

$$\bigvee_{\gamma\in Z_j} h_j(\mathbf{x}) = id(\gamma) \qquad \text{for all } j = 1,..,|\Gamma| \tag{4.9}$$

$$\sum_{s\in PS} f_s(\mathbf{x}) * p_s \leq \mathsf{RL} \qquad , \tag{4.10}$$

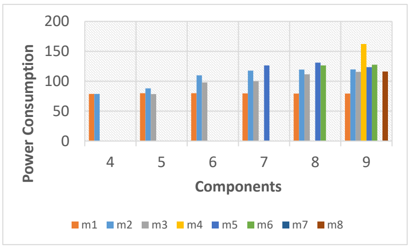
Tool Support and Validation on Automotive Benchmark

Our proposed ILP approach is validated on different size of software applications, i.e., applications with variable number of software components, cause-effect chains and degree of replication. The applications are synthesized according to the automotive benchmark proposed by Kramer et al. [42], which discusses a typical complexity of an AUTOSAR engine management system in terms of, e.g., the number and execution time of runnables, and the number and activation patterns of cause-effect chains.

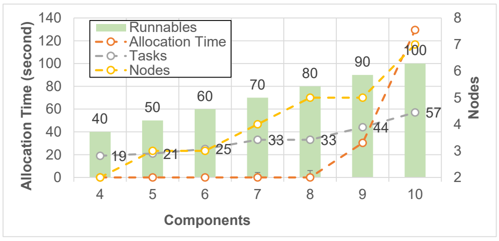
We prepared 6 software applications and an execution platform that consists 8 nodes. The smallest application consist of 4 components and the largest 10 components. Furthermore, the cause-effect chains varies from 10 to 60 and activation patters ranges from 2 to 4. Figure 4.8 shows the result of the optimization from the CPLEX solver, that is, the power consumption and the computation time, respectively. In general, the optimization maps components to nodes with lower power specification and higher processor speed, however, this is not always the case since the end-to-end timing and reliability constraints can dictate the selection of different nodes, which consume more power as a result. Figure 4.8b shows a slow increase of computation time from until the application with 8 number of components, which took 6.06 sec, and rapidly increased to 30.3 sec and 129.4 sec allocating application with 9 and 10 components, respectively. The optimization took extremely large amount of time for applications that consist of more than 10 components, and more than 15 component the waiting time was intractable, thus the optimization was interrupted manually.

4.6 Contr. 5: Metaheuristics-based Software Allocation

In the previous contribution, we observe that the ILP approach does not scale well to the large software allocation problems. To address this problem, in this section, we propose approximation algorithms based on metahuristics, which is “high-level problem-independent algorithmic framework that



(a) Power consumption of nodes.



(b) Allocation times software applications.

Figure 4.8: ILP optimization of different sizes of software applications based on the number of software components, cause-effect chains, computing nodes.

provides a set of guidelines or strategies to develop heuristic optimization algorithms". Meta-heuristic algorithms does not guarantee optimality of solutions, nevertheless, the solutions can be deemed satisfactory. In the case of minimizing power consumption, what is more appealing to system designers is usually the benefit attained as a result of the power consumption reduction, e.g., accommodating more and more software applications, improving battery-life, rather than optimality. In this spirit, metaheuristics can be useful to solve complex and large optimization problems, yet in reasonable time, as compared to exact methods, e.g., branch-and-bound, dynamic programming.

Hybrid Particle-swarm Optimization

The canonical PSO technique uses the constriction factors to balance exploitation and exploration of the search space to get closer to the global optima, hence improving solution quality. Nevertheless, it still suffers from premature convergence or local minima especially when applied on complex and large problems [62]. Its hybridization is proven to perform better in many cases [67]. In particular, it is shown to perform better in the tasks assignment problem, that is when hybridized with, e.g., the genetic algorithm [64], the hill-climbing [74], simulated annealing [75], differential evolution [70]. As compared to the hybridization with genetic, the hybridization with hill-climbing HCPSO is shown to perform better by Yin et al. [74] for the tasks allocation problem to maximize reliability of distributed systems.

In this work, we apply HCPSO to the problem at hand, and to tackle its stagnation when applied to large problems. Moreover, we hybridize PSO with the differential evolution technique, DEPSO, to improve diversification by applying the mutation and cross-over operators of the differential evolution. Algorithm 1 show the pseudocode of the hybrid PSO. Line 3 and 4 compute the personal best and the swarm best solutions, respectively. For each particle in the swarm, the velocity and position is computed in Lines 5-8. Lines 9-13 apply the hybridization based on the choice of the algorithm, i.e., DE, HCPSO and SHPSO intermittently, i.e., whenever the interval criterion condition is met.

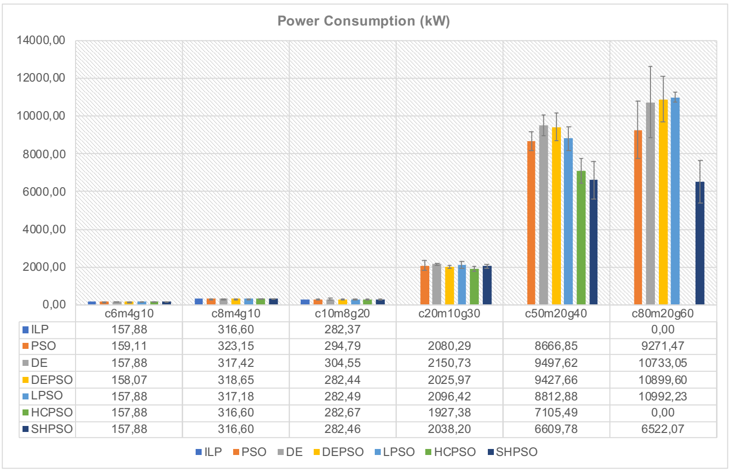
```
    input  : PSOparameters, DEparameters
    output: Software allocation solution sBest.x
 1  Particles $P \leftarrow$ initPSO();
 2  while termination criteria do
 3      $\mathbf{p}_{bst} \leftarrow$ComputePersonalBest($P$);
 4      $\mathbf{z} \leftarrow$ComputeSwarmBest($P$);
 5      foreach $p \in P$ do
 6          computeParticleVelocity($p$) according to
             Equation (2.9);
 7          computeParticlePosition($p$) according to
             Equation (2.10);
 8      end
 9      if interval criteria then
10          $P \leftarrow$ optimizeUsingDE($P$);
11          // $P \leftarrow$ optimizeUsingHC($P$)
12          // $P \leftarrow$ optimizeUsingSHC($P$)
13      end
14  end
```

Algorithm 1: Hybrid PSO Pseudocode.

Validation on the Bosch EMS Benchmark

In this section, we evaluate our proposed hybrid PSO algorithms for the allocation of software applications to heterogeneous computing units. The algorithms are evaluated against different specifications of automotive software applications and execution platforms with regard to effectiveness, stability and scalability. The software-application specifications consist of the number of software components c, runnables r, tasks t and cause-effect chains g. The specifications are synthesized from the automotive benchmark proposed by Kramel et al. [42]. The benchmark indicates a strong correlation between runnables and cause-effect chains in terms of timing and activation patterns. It shows the timing specifications of runnables and their shares in an engine management system. Moreover, it shows the activation patterns of cause-effect chains, the runnables per activation and their shares in the system. The engine management system is one of the most complex automotive systems in the vehicular electrical/electronic execution platform.

Result: We conducted two experiments: i) the first experiment compares the algorithms in terms of quality of solutions, computational time, stability; ii)



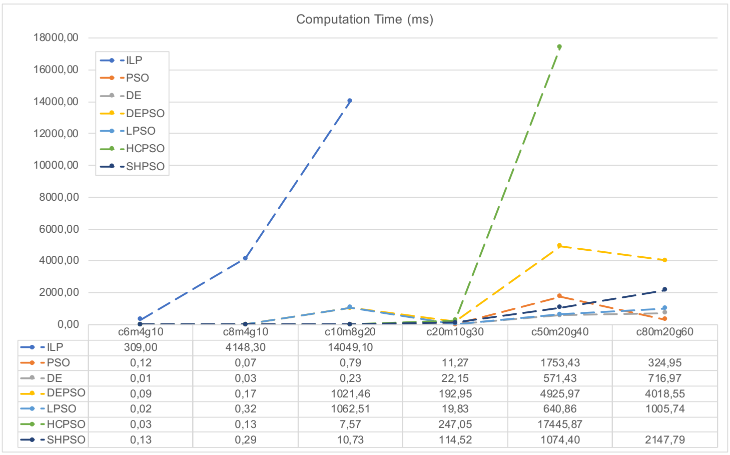
	c6m4g10	c8m4g10	c10m8g20	c20m10g30	c50m20g40	c80m20g60
ILP	157,88	316,60	282,37			0,00
PSO	159,11	323,15	294,79	2080,29	8666,85	9271,47
DE	157,88	317,42	304,55	2150,73	9497,62	10733,05
DEPSO	158,07	318,65	282,44	2025,97	9427,66	10899,60
LPSO	157,88	317,18	282,49	2096,42	8812,88	10992,23
HCPSO	157,88	316,60	282,67	1927,38	7105,49	0,00
SHPSO	157,88	316,60	282,46	2038,20	6609,78	6522,07

Figure 4.9: (Near) Optimal Power Consumption of the Different Software Allocation Problems.

the second experiment evaluates the effect of applying the approximation algorithm on the overhead of the end-to-end delays calculations due to the replication overhead.

In the first experiment, as illustrated in Figure 4.9, we show that ILPreturned optimal solutions in the first three problems, likewise all except PSO and DEPSO. However, as the problem size increases, the hybrid PSO with the local search algorithms delivered better solutions as compared to DEPSO. Futher, we showed that SHPSO scalled better than HCPSO and its results computation time became exponential as shown in Figure 4.10 and did not return solutions. In contrast, the hill-climbing-PSO hybrid performed the best in all last three problems except HCPSO failed to return near optimal solutions to the largest problem $c_{80}g_{20}n_{60}$. The result also shows that the stochastic version of the hill-climbing-PSO hybrid SHPSO scales better the over quality of the solution can deemed acceptable as compared to the ILP in the first three problems and HCPSO in the next two problems.

Due to the replication of components to meet the reliability goals, the overhead of computing the delays is exponential. Figure 4.11 compares the effect of with and without applying the approximation algorithm. It shows that $61\% - 81\%$ computational time improvement over the exact method while facing quality degradation only for samples $g_{30}d_3$ and $g_{60}d_2$. The improvements are in seconds, which means for a single usage (or run) of the meta-heuristic optimization algorithms, it is not significant. However, considering practical engineering process, which requires several iterations, the commutative effect of appliying the approaximation algorithm can beneficial in terms of responsiveness to engineers.



	c6m4g10	c8m4g10	c10m8g20	c20m10g30	c50m20g40	c80m20g60
ILP	309,00	4148,30	14049,10			
PSO	0,12	0,07	0,79	11,27	1753,43	324,95
DE	0,01	0,03	0,23	22,15	571,43	716,97
DEPSO	0,09	0,17	1021,46	192,95	4925,97	4018,55
LPSO	0,02	0,32	1062,51	19,83	640,86	1005,74
HCPSO	0,03	0,13	7,57	247,05	17445,87	
SHPSO	0,13	0,29	10,73	114,52	1074,40	2147,79

Figure 4.10: Computation time of the various algorithms for solving different instances of the software allocation problem.

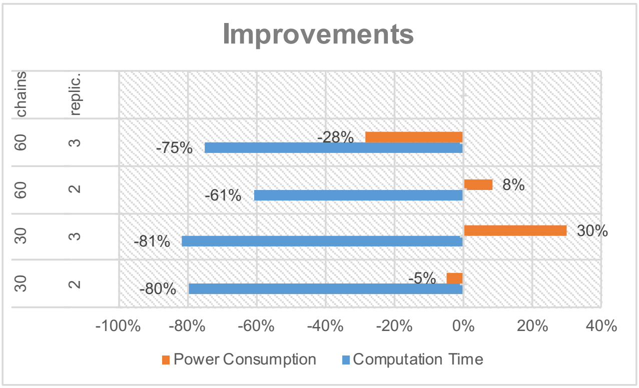


Figure 4.11: Effect of approximate algorithm over delay calculations with replication.

4.7 Paper Contributions

Tables 4.1 shows the peer-reviewed papers and their contribution to the thesis.

Paper A

ReSA: An ontology-based requirement specification language tailored to automotive systems. Nesredin Mahmud, Cristina Seceleanu and Ljungkrantz Oscar.*In the 10th IEEE International Symposium on Industrial Embedded Systems (SIES)(pp. 1-10). IEEE.*

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*

Paper	Sec4.2	Sec4.3	Sec4.4	Sec4.5	Sec4.6
A	×				
B	×	×			
C		×			
D			×		
E				×	
F				×	×

Table 4.1: *List of the peer-reviewed papers and their contribution to the thesis.*

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.

Personal Contributions: 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.
Status: Published

Paper B

ReSA tool: Structured requirements specification and SAT-based consistency-checking. Nesredin Mahmud, Cristina Seceleanu and Ljungkrantz Oscar. *In the 2016 Federated Conference on Computer Science and Information Systems (FedCSIS)(pp. 1737-1746). IEEE.*

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.*

Personal Contributions: 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.
Status: Published

Paper C

Specification and semantic analysis of embedded systems requirements: From description logic to temporal logic. Nesredin Mahmud, Cristina Seceleanu and Ljungkrantz Oscar. *In the International Conference on Software Engineering and Formal Methods (pp. 332-348). Springer, Cham.* Acceptance rate: 26%.

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.

Personal Contributions: 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.
Status: Published

Paper D

SIMPPAAL - A Framework For Statistical Model Checking of Industrial Simulink Models. Predrag Filipovikj, Nesredin Mahmud, Raluca Marinescu, Cristina Seceleanu, Oscar Ljungkrantz and Henrik Lönn. *Submitted to ACM Transaction on Software Engineering and Methodology (TOSEM). ACM Journals.*

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.

Personal Contributions: 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.
Status: Under Review

Paper E

Power-aware Allocation of Fault-tolerant Multirate AUTOSAR Applications. Nesredin Mahmud, Guillermo Rodriguez-Navas, Hamid Faragardi,Saad Mubeen and Cristina Seceleanu. *In the 25th Asia-Pacific Software Engineering Conference (APSEC'18). IEEE. .*

Abstract: *The growing complexity of automotive functionality has attracted revolutionary computing architectures such as mixed-criticality design, which enables effective consolidation of software applications with different criticality on a shared execution platform. Mixed-critical design that is required to satisfy end-to-end timing and reliability specifications should consider power-efficient software design in order to accommodate more and more functionality. Due to the recursive and exhaustive nature of the real-time and reliability analysis, exact methods, e.g., branch and bound, dynamic programming, are prohibitively expensive. We propose hybrid particle-swarm optimization algorithms based on differential evolution and hill-climbing algorithms to minimize power consumption of the safety-critical software, which have end-to-end timing and reliability requirements, on a network of heterogeneous computing units. The optimization approach employs fault tolerance to maximize reliability of the software applications subsequently meet the reliability requirements. Our proposed integrated software-allocation approach is evaluated using a range of synthetic software applications based a real-world automotive benchmark. The evaluation makes comparative analysis of the differential evolution, particle-swarm optimization, integer-linear programming and hybrid particle-swarm optimization algorithms. The results show that the hybrid algorithms based* *on the hill-climbing algorithms outperform the rest of the meta-heuristic algorithms, in particular, the stochastic version of the hill-climbing algorithm scales well in large software allocation optimization problems while its overall optimality performance can be deemed acceptable.*

My Contributions: 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.
Status: Published

Paper F

Optimized Allocation of Fault-tolerant Embedded Software with End-to-end Timing Constraints

Nesredin Mahmud, Guillermo Rodriguez-Navas, Hamid Faragardi,Saad Mubeen and Cristina Seceleanu. Optimized Allocation of Fault-tolerant EmbeddedSoftware with End-to-end Timing Constraints. *Mälardalen Real-time Research Center Technical Report (MRTC)*. Submitted to Elsevier JSA Journal.

Abstract: *The growing complexity of automotive functionality has attracted revolutionary computing architectures such as mixed-criticality design, which enables effective consolidation of software applications with different criticality on a shared execution platform. Mixed-critical design that is required to satisfy end-to-end timing and reliability specifications should consider power-efficient software design in order to accommodate more and more functionality. Due to the recursive and exhaustive nature of the real-time and reliability analysis, exact methods, e.g., branch and bound, dynamic programming, are prohibitively expensive. We propose hybrid particle-swarm optimization algorithms based on differential evolution and hill-climbing algorithms to minimize power consumption of the safety-critical software, which have end-to-end timing and reliability requirements, on a network of heterogeneous computing units. The optimization approach employs fault tolerance to maximize reliability of the software applications subsequently meet the reliability requirements. Our proposed integrated software-allocation approach is evaluated using a range of synthetic software applications based a real-world automotive benchmark. The evaluation*

makes comparative analysis of the differential evolution, particle-swarm optimization, integer-linear programming and hybrid particle-swarm optimization algorithms. The results show that the hybrid algorithms based on the hill-climbing algorithms outperform the rest of the meta-heuristic algorithms, in particular, the stochastic version of the hill-climbing algorithm scales well in large software allocation optimization problems while its overall optimality performance can be deemed acceptable.

Personal Contributions: I am the main driver of the paper. I developed the system model, the metaheuristic formulation, implemented the problem in Java, and collected and analyzed the experimental results. The co-authors gave writing update, useful ideas and comments on respected parts of the paper: Guillermo Rodriguez-Navas on reliability modeling, Hamid Faragardi on optimization, Saad Mubeen on the timing analysis, and Cristina Seceleanu gave idea on the main objective and constraints.
Status: Submitted to Journal of System Architecture (JSA), Elsevier Journals.

5. Research Method

Research methods, according to Jane et al. [8], are approaches, procedures and guidelines that are applied to conduct research, e.g., observation, interview, prototyping, experiment. 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: 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.

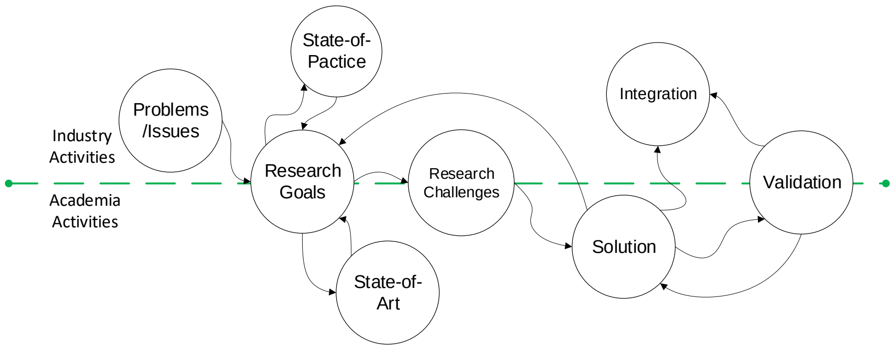


Figure 5.1: Research Process.

Figure 5.1 illustrates the research process employed in the thesis, which is an adaptation of the technology-transfer process proposed by Tony et al. [35]. 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* [17]. 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[1].

During implementations of the proposed solutions, we have applied a prototyping method [15], 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 [46, 47] as well as the SIMPPAAL framework [29]. 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 update to the state-of-the-art and state-of-the-practice methods and tools.

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

6. 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.

6.1 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 [43], e.g., Attempto Controlled English (ACE) [32], Processable English (PENG) [65], 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 [26], 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.

6.2 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[1]. Though, there is limited resource to investigate about the pros and cons of the product, the study by Nellen et al. [59] 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. [45] 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. [40] 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. [36] 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. [28] 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 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. [55] 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

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

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.

6.3 Software-to-Hardware Allocation

Different allocation schemes deliver different system performance and therefore efficient software allocation is crucial. Ernest Wozniak et al. [73] 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. [63] 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. [71] 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 [57]. On a different work, there are research focusing on power and energy consumption in real-time distributed systems which are employ dynamic voltage scaling [6] and task consolidation by minimizing computational nodes [25] [22].

7. 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. The rest 10% of the requirements consist of quantification and timed operators, therefore, cannot be effectively expressed in ReSA. This limitation can an impact the analysis outcome if the missing specifications are interwinded heavily with the rest of the specification, so peer-review is crucial to isolate, analyze, and determine the confidence in doing so.

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. The language can be exntended to support specification of requirments with quantification and timed properties. 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. Moreover,

Bibliography

[1] *Handbook of Metaheuristics*. 2006.

[2] 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.

[3] Particle swarm optimization. *SpringerBriefs in Applied Sciences and Technology*, 2016.

[4] Rajeev Alur. Timed Automata. pages 8–22. 1999.

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

[6] 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.

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

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

[9] Mehul Bhatt and Christian Freksa. Spatial Computing for Design—an Artificial Intelligence Perspective. In John S Gero, editor, *Studying Visual and Spatial Reasoning for Design Creativity*, pages 109–127, Dordrecht, 2015. Springer Netherlands.

[10] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh. Handbook of Satisfiability. *New York*, 185(3):980, 2009.

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

[12] WN Borst. *Construction of Engineering Ontologies*. PhD thesis, 1997.

[13] 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.

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

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

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

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

[18] Swagatam Das, Sankha Subhra Mullick, and P.N. Suganthan. Recent advances in differential evolution â An updated survey. *Swarm and Evolutionary Computation*, 27:1–30, 4 2016.

[19] Alexandre David, Kim G. Larsen, Axel Legay, Marius Mikučionis, Danny BÃ¸gsted Poulsen, Jonas Van Vliet, and Zheng Wang. Statistical model checking for networks of priced timed automata. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 6919 LNCS, pages 80–96, 2011.

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

[21] 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.

[22] 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.

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

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

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

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

[33] A.L. Goel. Software Reliability Models: Assumptions, Limitations, and Applicability. *IEEE Transactions on Software Engineering*, SE-11(12):1411–1423, 12 1985.

[34] Teofilo F. Gonzalez. *Handbook of approximation algorithms and metaheuristics*. 2007.

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

[36] 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.

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

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

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

[40] 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.

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

[42] Simon Kramer, Dirk Ziegenbein, and Arne Hamann. Real World Automotive Benchmarks for Free. In *6th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)*, 2015.

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

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

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

[46] 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.

[47] 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.

[48] 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.

[49] 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.

[50] 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.

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

[52] 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.

[53] 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.

[54] MathWokrks. Simulink Design Verifier.

[55] 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.

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

[57] 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.

[58] 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.

[59] 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.

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

[61] Alan Rector and Jeremy Rogers. Ontological and Practical Issues in Using a Description Logic to Represent Medical Concept Systems: Experience from GALEN. In Pedro Barahona, FranÃ§ois Bry, Enrico Franconi, Nicola Henze,

and Ulrike Sattler, editors, *Reasoning Web: Second International Summer School 2006, Lisbon, Portugal, September 4-8, 2006, Tutorial Lectures*, pages 197–231. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[62] Dian Palupi Rini and Siti Mariyam Shamsuddin. Particle Swarm Optimization: Technique, System and Challenges. *International Journal of Applied Information Systems*, 2011.

[63] 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.

[64] Andreas Sailer, Stefan Schmidhuber, Michael Deubzer, Martin Alfranseder, Matthias Mucha, and Juergen Mottok. Optimizing the Task Allocation Step for Multi-Core Processors within AUTOSAR. In *2013 INTERNATIONAL CONFERENCE ON APPLIED ELECTRONICS (AE)*, 2013.

[65] 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.

[66] Saptarshi Sengupta, Sanchita Basak, Richard Alan, and Peters Ii. Particle Swarm Optimization: A survey of historical and recent developments with hybridization perspectives. *arXiv:1804.05319*, 2018.

[67] Saptarshi Sengupta, Sanchita Basak, Richard Alan, and Peters Ii. Particle Swarm Optimization: A survey of historical and recent developments with hybridization perspectives. *arXiv:1804.05319*, 2018.

[68] Lui Sha, Tarek Abdelzaher, Karl Erik Årzén, Anton Cervin, Theodore Baker, Alan Burns, Giorgio Buttazzo, Marco Caccamo, John Lehoczky, and Aloysius K. Mok. Real time scheduling theory: A historical perspective, 2004.

[69] 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.

[70] Rainer Storn and Kenneth Price. Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, 1997.

[71] 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.

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

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

[74] Peng Yeng Yin, Shiuh Sheng Yu, Pei Pei Wang, and Yi Te Wang. Task allocation for maximizing reliability of a distributed system using hybrid particle swarm optimization. *Journal of Systems and Software*, 80(5):724–735, 2007.

[75] Fuqing Zhao, Yi Hong, Dongmei Yu, Yahong Yang, Qiuyu Zhang, and Huawei Yi. A hybrid algorithm based on particle swarm optimization and simulated annealing to holon task allocation for holonic manufacturing system. *International Journal of Advanced Manufacturing Technology*, 2007.