MŰEGYETEM 1782

Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Measurement and Information Systems

Abstraction Techniques for the Analysis and Synthesis of Critical Cyber-Physical System Architectures

PhD Thesis Booklet

Kristóf Marussy

Thesis supervisor:
István Majzik, PhD

Budapest, 2024

1 Introduction

Cyber-physical systems (CPS) are smart systems that include highly interconnected digital, analog, physical, and human components. They provide new functionalities that improve quality of life and enable technological advances in critical areas, such as personalized healthcare, traffic flow management, smart manufacturing, energy supply, autonomous vehicles, and intelligent buildings [NIST17].

The pervasive interconnectedness of computational and physical aspects, as well as the often critical roles these systems are employed lead to *specific characteristics* of CPS that go beyond traditional system and application design:

- CPS may be deployed in a *wide variety of configurations* and on *heterogeneous computing* and *communication platforms* and are often composed with other systems as part of a *System of Systems.* Engineering such systems needs a methodology to ensure the interoperability of components, manage the evolution of the requirements and the design, and deal with any harmful phenomena emerging from the complex interactions.

- The *potential impact* of CPS on the physical world, up to the possibility of massive economic damage or even the loss of life, places a heightened demand for trustworthy systems. These concerns result in the need for rigorously proven security, privacy, safety, reliability, and resilience. The systems may rely on runtime adaptation techniques to ensure that their dependability objectives are met in the face of changes in the operating environment and failure processes of the hardware [Epi+09; FB16; IW17].

- The interactions between CPS and their operating environment are often *time sensitive.* There may be hard real-time requirements placed on certain functions. For example, we must ensure that the latency between sensing and actuation remains limited for correct control loops behavior.

Therefore, CPS have to satisfy stringent *extra-functional requirements*, such as maintainability, reusability, extensibility, cost, security, privacy, safety, dependability, scalability, and performance, in addition to their functional requirements [Fel03]. *Model-driven engineering* has been widely applied as a methodology to facilitate CPS design in accordance with these goals, especially in the context of changing and evolving requirements and architectures [Vog+15].

1.1 Architecture-based analysis of extra-functional requirements

A large fraction of the extra-functional requirements is *quantitative*, i.e., they can be evaluated in a mathematically precise way by computing some *metric* on either the CPS design artifacts or an auxiliary *analysis model.* For complex metrics, the architecture models (e.g., block diagrams) of the CPS are usually not sufficient, and we have to adopt specific mathematical formalisms for constructing analysis models. Among these, stochastic models, such as *fault trees* [Xia+11; Gha+17; Get+18][c17], *Markov chains* [KB09], *queuing networks* [KR08], and *Generalized Stochastic Petri Nets* (GSPN) [BMM99; BDD04; LMC04; Ndi+16; CET18][c7; c12], serve as analysis models for reliability, availability, and performance metrics.

Manual creation of analysis models requires specialized expertise and meticulous work for each architectural alternative to be analyzed. For complex design spaces, a large (or possibly even infinite) amount of candidate architectures must be analyzed [CA05; KR11; GTC15]. Such large-scale manual analyses are infeasible. To alleviate this problem, *architecture based* analysis techniques, e.g., [BMM99; BDD04; FD16; KR08; Gha+17], have been developed, which rely on *model transformations* to automatically construct analysis models from the architectural design artifacts. However, these techniques can only handle static architecture models: the possible runtime changes have to be fully described in the analysis model, e.g., as failure processes and other random processes in the case of stochastic modeling.

To enable runtime monitoring and adaptation, the models@run.time paradigm [BBF09; Che+11; Búr+20] facilitates the capture of runtime knowledge about the system and its environment as a continuously maintained model. Although this allows applying model-driven engineering techniques to create adaptation and reconfiguration strategies, the mathematically precise extra-functional analysis of such strategies raises a need for evaluating the extra-functional properties of dynamic (changing) architecture models [Cal+12].

The first research objective investigated in this thesis aims to address this issue by extending the architecture-based approach to the runtime adaptation strategies.

Research objective 1 Quantitative extra-functional analysis of dynamic system architectures

1.2 Automated generation of design candidates

The large number of possible system configurations each with highly varying extra-functional characteristics poses a major challenge to find the most suitable system architecture in an early design phase [NIST17]. Engineers can manually inspect only a handful subset of candidate architectures. Therefore, various automated *design space exploration* (DSE) tools have been developed *for system architecture synthesis* to assist in finding viable candidate architectures that satisfy all functional and extra-functional constraints while optimizing for a target objective. DSE tools tackling this challenge are broadly classified into two categories:

(Meta-)heuristic techniques, such as genetic algorithms or multi-objective optimization [Mar+10; Abd+14; GTC15; BZS18; Arc+18; FTW16], can support a wide variety of analyses directly inside the DSE process to derive near-optimal design candidates. However, they do not guarantee a complete (exhaustive) enumeration of the design space and the optimality of the generated candidates [Ker+13]. Therefore, engineers are not informed about the cause of failure (e.g., an *unsatisfiable core*) if the exploration fails to produce the desired candidates. Moreover, encoding hard constraints (which must be satisfied at all times) either requires approximations by custom soft constraints (which can be violated, but violating solutions are penalized), objective functions and mutation operators, or it could significantly degrade the performance or scalability of the exploration [SNP13; BZJ21].

Logic solver based DSE techniques (e.g., [Jac02; KJS10]) have guaranteed *soundness* and *completeness*. They usually allow encoding complex logical hard constraints and logical formulas or model queries [CCR07; KHG11; Ujh+15; SNV18] and may provide an explanation when the synthesis task is unsatisfiable. However, purely logical constraints cannot capture most extra-functional requirements that rely on an external numerical solver to analyze. Thus, solvers have to be specifically extended for optimization tasks, such as in [Li+14; BPF15] to handle both logical and numerical constraints. Unfortunately, for complex extra-functional analysis tasks, such optimizing solvers are often unavailable.

Recently, logic-solver based graph model generation has been suggested, which takes advantage of *partial modelling* [RSW04; RD06] to explicitly represent design decisions yet to be made in the internal representation of the solver as a graph model [SV17; SNV18]. This represents a potential for applying architecture-based extra-functional analysis directly in the logic solvers to generate design candidates with completeness guarantees while keeping expressiveness similar to that of (meta)heuristic techniques.

The second research objective in this thesis focuses on this issue:

Research objective 2 Synthesis of candidate system architectures with completeness guarantees according to quantitative extra-functional requirements and objective functions

1.3 Representing uncertainty and variability

Both **Research objective 1** and **Research objective 2** require a representation of CPS architectures to analyze and synthesize. Architecture modeling languages, such as SysML, Palladio [Mar+10], Æmilia [Arc+18], and domain specific languages rely on *graph models* to represent system architectures, configurations, and deployments on heterogeneous computing and communication infrastructures [Vog+15]. In this setting, *graph transformations* can express reconfigurations as endogenous (in-place) transformations of the architecture models, while exogenous transformations can automatically derive analysis models as target models from the source architecture models [Koz10].

However, both **Research objective 1** and **Research objective 2** pose challenges not yet tackled by existing graph model representations.

1.3.1 Representations for view transformations

In **Research objective 1**, (endogenous) transformations for applying reconfiguration occur at the same time as (exogenous) *view transformations* for maintaining analysis models. It is possible to execute the view transformation from scratch after each reconfiguration, but this can have significant performance costs, and leads to the loss of valuable *traceability* information that connects the prior and current versions of the analysis model. Therefore, an ideal *view transformation engine* [Ber+12; CET18] is *reactive* (i.e. reacts to source model changes), target *incremental* (i.e., updates only affected target elements), *consistent* (i.e., continuously maintains a transformation relation between source and target models) and *validating* (i.e., the target model is a valid instance of the target language).

A validating engine that satisfies all well-formedness constraints of the target language is at odds with reactive and incremental execution: depending on the changes to the architecture model, it may be the case that no valid target model exists at some time [c7]. By semantically preserving inconsistencies in an *inconsistency-tolerant* knowledge base, a large fragment of the source and view models can be kept sync in case of such model editing operations. This allows preserving traceability links and provides hippocratic behavior (i.e. avoids the unnecessary deletion and recreation of elements).

Moreover, the development of the analysis model transformation usually requires the collaboration of experts from multiple domains (e.g., embedded systems, communications infrastructure, hardware reliability). Integrating their results requires combining *partial* information obtained by model transformations created according to multiple viewpoints to construct the complete analysis model [CNS12].

1.3.2 Representations for design-space exploration

Regarding **Research objective 2**, the need for *partial* and *inconsistency-tolerant* representations of CPS architectures is even more explicit.

By introducing a *fixed number* of explicit points of variability into a system model, such as the number of redundant component instances and the possible allocations of functions, a genotype vector for systems models can be constructed [Mar+10]. This enables the use of efficient meta-heuristic algorithms either on the level of architecture models [Ale+09; Mar+10; Li+11; BFK19] or directly on the level of analysis models [GTC15].

While the aforementioned approaches offer scalability due to the fixed-length, domain-specific genotype encoding, such encodings are not directly applicable for problems with a *variable number of objects and connections*, such as communication network topologies. To this end, logic solver based approaches for model synthesis [SV17; SNV18] rely on partial graph models to encode design decisions yet to be made as unknown aspects of the design candidate being worked on.

Conversely, representing inconsistencies that arise during the execution of decision procedures can pinpoint contradictions in the requirements.

Because existing approaches with partial graph models have limited support for evaluating extra-functional metrics [FFJ12], there is a need for extending partial graph model formalisms with such support for the quantitative extra-functional analysis of CPS architectures. The contributions presented in the thesis in connection with **Research objective 2** aim to provide reasoning capabilities over these extended partial model representations.

In summary, the following research objective arises as the common theoretical background for **Research objective 1** and **Research objective 2**:

Research objective 3 Represent uncertainty caused by design decisions yet to be made, run-time reconfigurations, as well as inconsistencies in complex system architectures explicitly

2 Background and challenges

In this section, we overview the state-of-the-art results connected to our research objectives and identify five *challenges* to be addressed later in this thesis.

2.1 State space explosion

As we discussed in Section 1.2, the large number of possible hardware architectures, software component allocations, and configurations poses a significant difficulty in the verification and synthesis of design candidates. Even if we consider only a single node (component) type and a single edge (link) type, there are $2^{10\times 10} \approx 10^{30}$ possible graph models with 10 nodes. Real system design artifacts (e.g., SysML, Palladio, or domain-specific models) have both more node and edge types, as well as more components. Thus, the *design space* formed by possible graphs models can be much larger, or potentially infinite when we consider node attributes with continuous range (e.g., probabilities and failure rates).

Approaches based on *(meta-)heuristics* aim to tackle this problem by only exploring a fraction of the design space. The heuristics, such as genetic algorithms, are used to sample parts of the design space in an adaptive manner that are most likely to contain optimal solutions of the design challenge. In this case, the explorations task consists of various *hard* and *soft constraints* on the design candidates and one or more *objectives* (usually corresponding to extra-functional metrics) to be optimized. As a benefit, a wide variety of constraints and objectives can be supported, even in a *multi-objective* setting where *Pareto-optimal* [REJ09] solutions according to multiple objectives are sought.

2.1.1 Genotype-based approaches

By translating the design candidates into a specialized parametric *exploration representation*, such as a finite-dimensional vector, approaches like ArcheOpteryx [Ale+09], AQOSA [Li+11], PerOpteryx [Mar+10], EvoChecker [GTC15], and Rodes [Cal+17] can execute genetic algorithms on design candidates with a high probability of convergence to near-optimal solutions. For example, a fixed number of parameters might be introduced for the allocations of software components to the hardware platform and for the available component variants. However, in many graph-like problems, were the number of components and their interconnections are also variable, it is either impossible to introduce such a finite parametrization, or the required number of parameters (e.g., at least n^2 parameters for links in a graph with n nodes) grows infeasibly large.

2.1.2 Graph-based approaches

In contrast, graph-based techniques use graph transformations [Agr+02] or refactorings to generate candidate designs as graph models. They either rely on *model-based* search, where a graph model is being mutated, or *rule-based* search, where solutions are encoded as a sequence of graph transformation operations [Joh+19]. MOMoT [FTW16] and MDEOptimiser [BZS18] rely on the Henshin model transformation language [Are+10] for model-based exploration. Hard constraints pose a challenge for such approaches: they are either handled by relaxation into *soft* constraints or by encoding them in the transformation rules [BZJ21].

Viatra-DSE [Abd+14; HHV15] is a rule-based DSE tool that relies on the Viatra [Ujh+15] language. SHEPhERd [CMP15] and EASIER [Arc+18] aim to derive sequences of software architecture refactorings according to extra-functional criteria. Synthesizing long chains of model transformations might be challenging in the presence of hard constraints; e.g., the effectiveness of evolutionary *crossover* operators is diminished compared to *mutation* operators [Abd+14].

Challenge 1 How to represent design candidates and perform search and analysis in the extremely large design and configuration spaces arising from complex CPS?

2.2 Functional and structural constraints

In model-driven engineering, general-purpose or domain-specific *systems modeling languages* (e.g., UML, SysML, Palladio [BKR08]) are used to represent system architectures. Such languages are defined by (1) a *metamodel* to capture the vocabulary of language with classes and references, while (2) consistent (valid) systems models also have to satisfy *well-formedness* (WF) constraints.

We will assume that *first-order logic* (FOL) can uniformly formalize (a) the constraints associated with the systems modeling language, (b) additional design rules, and (c) functional requirements. For example, if the system modeling language can describe the physical and functional system architecture and their allocations, FOL constraints can capture whether all functions are allocated to physical components in a valid manner. Thus, as a theoretical basis, we may use FOL signatures and logic formulas to capture metamodels and WF constraints, respectively.

Structural constraints, including *type hierarchy*, *type compliance*, *multiplicity* constraints, *inverse relations*, and the *containment hierarchy* are often imposed by model management frameworks to ensure that the model artifact remains serializable [Ste+09]. Several highly expressive languages exist for the specification of well-formedness constraints, e.g., OCL [OCL] and VIATRA [Ber+11].

As such, generating valid models *requires solving a logical program with the satisfaction of user-provided FOL formulas* as *hard constraints*, which is highly challenging even in the absence of extra-functional metrics [CCR07; KHG11; Var+18]. Typically, only a small fraction of the possible model candidates will satisfy the complex logical hard constraints.

FOL hard constraints often pose a challenge in (meta-)heuristic DSE approaches. Burdusel et al. [BZJ21] proposed the automated generation of transformation rules that preserve a limited class of WF constraints (multiplicity constraints). PLEDGE [SSB20] combines evolutionary search with logic solving to preserve WF constraints over object attributes. Other, more complex FOL constraints cannot be encoded and have to be relaxed into soft constraints.

Logic solver based techniques for FOL problems include constraint programming, such as in UMLtoCSP [CCR07] and DesertFD [ENS10], SAT solving, such as in Alloy [Jac02] and CoBaSA [MVS07], and Satisfiability Modulo Theories (SMT), such as in FORMULA [KJS10].

SAT and SMT solvers commonly rely on the Davis-Putnam-Logemann-Loveland (DPLL) [DLL62; Kat+16] and the abstract DPLL [NOT06; Bra+13] algorithms for an efficient, *complete* enumeration of the design space. However, scalability of logic solvers may be limited in the case of graph-like synthesis problems [SNV18].

Recently, partial modeling [RSW04; Ren06] based graph generation was proposed [SNV18] as an implementation of a DPLL decision procedure for graph models, which improves performance over SAT solver for graph model generation. Nevertheless, integrating partial modeling based DPLL with other *background theories* for SMT solving, e.g., for attribute constraints, remains challenging.

Challenge 2 How to ensure that functional and structural constraints are satisfied in architecture and analysis models?

2.3 Dependability analysis models for reconfigurable systems

Specifying and analyzing extra-functional metrics and requirements in an architecture-based manner may pose significant challenges in complex, adaptive and reconfigurable systems. Not only an analysis model has to be constructed from the system architecture automatically, but the analysis itself must also remain tractable.

Methods for the construction of stochastic analysis models are widespread in the evaluation of dependability (including reliability and availability), and performance metrics based on architecture models, especially for component-based design [Koz10; BMP12]. Underlying analysis formalisms include *fault trees* [JVB17; Xia+11; Gha+17; Get+18], *Markov chains* [KB09], *queuing networks* [KR08], and *Generalized Stochastic Petri Nets* (GSPN) [MPB02; BDD04; MB15; Ndi+16; CET20].

The *dependability attributes* and *failure processes* may be described in two ways: (a) They may be represented directly in the architecture model using, e.g., UML stereotypes [MPB02], MARTE annotations [Iqb+15] or the AADL Error Annex [JVB17; Ale+09]. Alternatively, (b) the transformation definition itself may encode the dependability attributes of the components (as analysis model fragments). To retain flexibility in the transformation definition and aggregate knowledge for multiple domain experts, transformation definitions may be composed *sequentially* [Anj+14; Heg+16] or in *parallel* [CNS12; DXC11], where the various elementary transformations to be composed describe the dependability attributes of different parts of the system.

The analysis of adaptive systems poses further challenges. If the adaptations are *dynamically* synthesized in response to failures and events from the environment, *quantiative verification* must be employed at runtime [Cal+12] to verify the proposed adaptations. If the system uses a *static* adaptation strategy, an analysis model that includes the behaviors of all possible configurations of the system along with the strategy may grow so large that it makes the analysis intractable.

To alleviate this issue, *Phased-Mission System* (PMS) [MB99] models were developed, where *phases*, which are stochastic models (e.g., fault trees, Markov chains, GSPNs) describing the behaviors of system configurations, are connected by a *high-level* model describing the adaptation strategy. Numerical [SRA92; MB99] or combinatorial [Xin07; WT07] analysis methods can process phases one-by-one, connecting the individual models later according to the high-level model. Therefore, this formulation reduces both the memory and computation demands of extra-functional analysis.

However, to our best knowledge, while analysis models for individual phases can be constructed from architecture models by model transformations, no approach supports simultaneously deriving all phases and the high-level model for the PMS analysis of an adaptation strategy.

Challenge 3 How to compactly specify and analyze dependability measures for families of reconfigurable architecture model candidates?

2.4 Witness models for Worst-Case Execution Time Analysis

In embedded systems, runtime monitoring programs are integral components of the system that analyze events and execution traces [Bar+18] in order to detect potentially critical situations that violate a requirement. Task schedulability and real-time requirements demand the use of runtime monitors that adhere to *Worst-Case Execution Time* (WCET) constraints [Pik+10; HR02].

According to the models@run.time [BBF09; Che+11] paradigm, runtime monitors are continuously executed on runtime *snapshots*, i.e., models describing the runtime state of the system. The *heavily data-driven* execution of such runtime monitors compared to the *low expressiveness* of traditional runtime monitoring solutions [Hav15] makes producing analysis models for safe and tight WCET estimation challenging [Búr+18; Har+19; DBB18]. This necessitates the use of *semantic-aware WCET estimation* [Mai+17] to consider limitations on input data (i.e., constraints restricting the possible runtime snapshots).

The *Implicit Path Enumeration Technique* (IPET) [LM97] constructs *Integer Linear Programs* (ILP) as WCET analysis models according to the *Control Flow Graph* (CFG) on the runtime monitor programs. The IPET ILP can accurately represent the *microarchitectural timing characteristics* of the target execution platform along with the control flow of the program.

The effectiveness of WCET analysis relies on precise program *flow facts* (e.g., loop bounds, infeasible paths). Although there are several algorithms to derive additional constraints on the program flow [Gus+06; Erm+07; CJ11; KKZ13; Lis14], there is still a significant manual effort needed to specify flow facts [Abe+15] to represent *domain-specific* semantic information about program inputs.

We identified three domain-specific analysis scenarios for runtime monitors: (i) WCET based on a single runtime snapshot, (ii) WCET based on the metamodel and well-formedness constraints of the runtime snapshots, and (iii) WCET based on a *partial* runtime snapshot. The latter enables reasoning when some information is fixed at design time (e.g., a railway network) while the rest of the model changes at runtime (e.g., the positions of trains).

Classical WCET analysis methods (without domain-specific analysis) can handle scenario (i) by *data flow analysis*. However, they require over-approximations based on *loops bounds* for scenario (ii), because they cannot take well-formedness constraints of runtime models into account. They fail to address scenario (iii), because data flow analysis cannot process partial data that without a pre-allocated memory layout.

In [Búr21][j5], Búr proposed *witness models* to tackle scenarios (ii) and (iii). The witness model is a concrete runtime snapshot that maximizes the estimated upper bound of the execution time of the runtime monitor. Thus, it provides a safe and tight estimate of the WCET. However, *synthesizing* the witness model based on the domain constraints remains challenging.

Challenge 4 How to synthesize witness models for domain-specific Worst-Case Execution Time evaluation?

2.5 Numerical reasoning with extra-functional attributes

The analysis of quantiative extra-functional properties requires *numerical* (quantitative) *reasoning* for evaluating *numerical constraints* over the attributes of the architecture models, as well as computing *extra-functional metrics* during synthesis.

In the case of (meta-)heurisic synthesis approaches, a concrete candidate architecture is available at all times during synthesis. Thus, analysis models can be constructed by model transformations and analyzed using analysis tools, such as in [Ale+09; Mar+10; Li+11; FTW16; BZS18].

In logic solvers, the supported metrics and objectives is limited by the input language and the supported *background theories* [Kat+16]. For supported theories, optimizing SMT solvers provide capabilities for finding globally optimal solutions [Li+14; BPF15]. The Nelson–Oppen procedure [NO79] is widely employed as means of combining background theories.

However, the scalability of traditional logic solvers may be limited in the case of graph-like synthesis problems [SNV18]. Tableau-based reasoning for graphs has been proposed in [Pen08; SLO17; ADW16], but these approaches lack the support for reasoning with extra-functional properties.

In abstract interpretation, *abstract domains* [CH78; SPV18] are used to reason about the (numerical) values of program variables and expressions. *Relational* abstract domains, such as *polyhedron abstraction* [CH78; BHZ08] can represent arbitrary numerical relationships between arbitrary program variables. Reasoning techniques for polyhedra include *Linear Programming* (LP) and *Integer Programming* (IP) solvers [Clp; Cbc], as well as specialized representations [Nin15].

In contrast, *non-relational* domains, such as *interval abstraction* reduce expressiveness by restricting each representable numerical constraint to only refer to a *single* variable, but offer more efficient reasoning e.g., by *interval arithmetic* [Kul09] and *propagators* [Nin15].

Even though partial modeling [RSW04; Ren06] based graph generation has been recently proposed [SNV18] as an implementation of a DPLL [Kat+16] decision procedure for graph models, the support for numerical reasoning remained lacking. There are two sources of quantitative values in a graph model: (i) objects of the model may be equipped with numerical attributes (e.g., elementary performance or dependability metrics), and (ii) the *size* or *cost* measures may depend on the number of objects.

For (i), the combination of abstract domains and partial models [Mag+07; MRS10; FFJ12] was proposed, especially for the *value analysis* of heap and pointer-based programs [APV09], where partial models represent possible program states. For (ii), the most closely related works are *structural counter abstraction* [Ban+13] for graph transformation systems and *model-based quantifier instantiation* [Rey+13] in SMT solvers. Representing models with numerical attributes where the model size (and thus the number of attribute values to be represented) is not known in advance has been tackled by *summarized dimensions* [Gop+04].

However, the aforementioned techniques do not in themselves provide a sound

Table 1: Research objectives and challenges

	Ch. 1	Ch. 2	Ch. 3	Ch. 4	Ch. 5
Research objective 1	○	●	●	○	○
Research objective 2	●	●	○	●	●
Research objective 3	○	○	●	○	●

and complete decision procedure for graph generation, and their use remains limited to program verification.

Partial modeling allows a sound and complete decision procedure for graph generation and synthesis by computing *under-* and *over-approximations* of the constraints to be satisfied [SV17] following the abstract DPLL [NOT06; Bra+13] framework. However, its use is currently limited to logical (FOL) well-formedness constraints. Thus, the use of numerical abstractions over (i) model attributes and (ii) model size, as well as the combination of various reasoning techniques in the context of graph models remains challenging.

Challenge 5 How to reason about the quantitative extra-functional aspects of complete or incomplete system architectures automatically and efficiently?

The relationships between our research objectives and the aforementioned challenges are summarized in Table 1.

3 Research method

My research method has been aligned with the best practices of software engineering research. **Research objectives 1–3** were motivated by generalizing industrial problems and case studies. The feasibility and applicability of the contributions were demonstrated by open source prototype implementations, while their scalability was evaluated by synthetic and real-world performance benchmarks. The prototypes are integrated with industrial and academic tools, such as EMF [Ste+09], Viatra Query [Ujh+15], Viatra Generator [SNV18], and PetriDotNet [j6].

In particular, the following benchmarks and case studies were used to motivate and evaluate the contributions:

The **Interferometry Mission Architecture Optimization** [Her+17] case study was adapted from the NASA Jet Propulsion Laboratory. It aims at deriving design candidates for interferometry missions with multiple satellites (including CubeSats and small satellites) communicating over radio links with each other and a ground station using an optimization approach based on model transformations.

The **Train Benchmark** [Szá+17] is an open source framework for measuring the performance of continuous model transformations, with a particular emphasis on the performance of incremental query reevaluation. It has been actively used within the model-driven engineering community as a cross-technology performance benchmark. It uses a domain-specific model of a railway system originating from the MOGENTES EU FP7 [MOG] project.

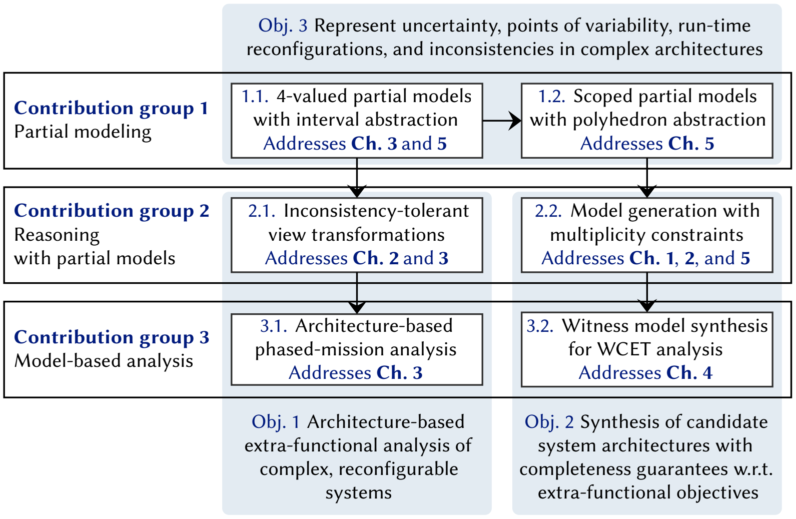
Obj. 3 Represent uncertainty, points of variability, run-time reconfigurations, and inconsistencies in complex architectures
Contribution group 1
Partial modeling
1.1. 4-valued partial models with interval abstraction
Addresses Ch. 3 and 5
1.2. Scoped partial models with polyhedron abstraction
Addresses Ch. 5
Contribution group 2
Reasoning with partial models
2.1. Inconsistency-tolerant view transformations
Addresses Ch. 2 and 3
2.2. Model generation with multiplicity constraints
Addresses Ch. 1, 2, and 5
Contribution group 3
Model-based analysis
3.1. Architecture-based phased-mission analysis
Addresses Ch. 3
3.2. Witness model synthesis for WCET analysis
Addresses Ch. 4
Obj. 1 Architecture-based extra-functional analysis of complex, reconfigurable systems
Obj. 2 Synthesis of candidate system architectures with completeness guarantees w.r.t. extra-functional objectives

Figure 1: Overview of research objectives and contributions

The **Model-Based Demonstrator for Smart and Safe Cyber-Physical Systems (MoDeS3)** [Vör+18b] showcases various challenges in intelligent safety-critical CPS. It extends the *structure model* of a railway system from the Train Benchmark with a *runtime model* [BBF09] that tracks the positions of trains, which enables detecting and correcting potentially unsafe situations by model-based runtime monitoring [Bar+18; Búr+20].

Lastly, the **Flexible Manufacturing System (FMS)** [CT93] case study illustrates the performability analysis of an industrial CPS system using Petri nets.

4 Contribution overview

This thesis presents the contributions related to **Research objectives 1–3** in three groups. The groups are arranged starting from the theoretical foundations to concrete extra-functional analyses. Figure 1 shows the relationship between **Research objectives 1–3**, **Challenges 1–5**, and **Contributions 1–3**.

Contribution group 1 addresses **Research objective 3** by proposing novel *partial model* formalisms to represent graphs models with unknown or inconsistent information along with extra-functional attributes of model elements.

Contribution group 2 provides *structural* and *numerical reasoning* capabilities over the partial models to efficiently construct analysis models and provide *over-* and *under-approximation* of extra-functional metrics. Taking advantage of these capabilities, **Contribution group 3** presents two architecture analysis methods.

The contribution groups are split into two contributions each. *Contribution 2.1* provides reasoning capabilities with analysis models for dynamic architecture models, which are exploited in *Contribution 3.1* for the *phased-mission analysis* of reconfigurable CPS to address **Research objective 1**. Likewise, *Contribution 2.2* provides reasoning capabilities over systems of linear equations and the sizes of models, which are exploited in *Contribution 3.2* for the WCET analysis of *monitor query programs* to address **Research objective 2**.

As per university regulations, contributions in this section are formally proposed using the *first-person singular* ("I"). The highlighted contributions, published in the works cited therein, are the work of the author of this thesis alone. The rest of the thesis follows the conventions of the field by using the *first-person plural* ("we").

4.1 Partial modeling for quantitative extra-functional analysis

In order to tackle **Challenges 3** and **5**, the first group of contributions in this thesis provides extended FOL structures to compute quantitative extra-functional metrics of incomplete or inconsistent architecture models.

In *Contribution 1.1.1*, we proposed *4-valued partial models* [j2], which adopt the 4-valued Belnap-Dunn logic [Bel77; KO17] to represent *incomplete* or *inconsistent* models. We also introduced *multi-objects* to compactly represent from zero to many potential model elements with a single object by interpreting the existence and equality of objects with 4-valued logic. Object equality can be also used to denote the *merging* of elements from multiple sources of information [SE06; CNS12], where inconsistency-tolerance is exploited to mark merge conflicts and conflicting executions of composite view transformations [c7; d21] (**Challenge 3**).

In *Contribution 1.1.2*, we proposed an *abstraction for numerical attribute values and graph metrics* [j2] (**Challenge 5**) in partial models by exploiting *interval abstraction* [Kul09] to combine partial modeling with *value analysis* [FFJ12].

Lastly, in *Contribution 1.2*, we proposed *scoped partial models* [j1] as a conceptual core for evaluating *model size constraints* over partial models (**Challenge 5**). By employing *polyhedron abstraction* [BHZ08], they offer finer-grained abstraction over the number of model elements represented by multi-objects. This allows representing extra-functional constraints on the number of model elements (e.g., model size constraints, linear cost functions) within the partial model. As a limitation, scoped partial models rely on *3-valued logic* [SV17] and lack inconsistency-tolerance.

Contribution group 1 I defined *4-valued* and *scoped* extensions of the partial graph model formalism for the quantitative extra-functional analysis of system architecture models with unknown properties and pending design decisions.

Contribution 1.1 I formalized *4-valued partial models* as a conceptual core to introduce *inconsistency-tolerance* into architecture models and other graph models [j2; c7; d21].

1.1.1. I proposed the use of 4-valued Belnap–Dunn logic to explicitly capture both *unknown properties* and *pending design decisions* (paracompleteness), as well as *inconsitencies* (paraconsistency).

1.1.2. I proposed the use of *interval abstraction* to handle *unknown* and *uncertain attribute values*, which enables the *under-* and *over-approximation* of logic formulas over models containing *complex numerical constraints.*

Contribution 1.2 I formalized *scoped partial models* as a conceptual core for model generation with linear numerical constraints, proposing the use of *polyhedron abstraction* to express constraints on model size, number of graph pattern matches, and costs [j1].

Added value The contributions in this contribution group provide abstractions for incomplete and inconsistent architecture models for evaluating structural and numerical constrains and quantitative extra-functional metrics. As such, they serve as a theoretical basis for the contributions in **Contributions groups 2–3**.

In particular, the under- and over-approximation of constraint satisfaction enables the use of the abstract DPLL [NOT06; Bra+13] algorithms for the synthesis of architecture models. They aid in detecting surely violated functional and extra-functional requirements. Hence, when designing an architecture, mistakes and inconsistencies can be detected even before the full model is ready [CNS12; SV17][c7]. In automated synthesis, the detection of inconsistencies results in backtracking in the search space [SNV18; Var+18][j1] to speed up model generation.

To our best knowledge, these contributions are the first to address the sound and complete abstract DPLL synthesis of candidate architecture models.

Applications and related contributions Contributions in this contribution group were published as part of joint work with Dániel Varró, Oszkár Semeráth, and Aren A. Babikian.

Joint work also with Zoltán Micskei, András Vörös, Zoltán Szatmári, and Csaba Hajdu [c8] proposed an end-to-end framework for the run-time monitoring and testing of autonomous vehicles with coverage metrics based on qualitative graph abstractions. Building on this framework, joint work also with Anqi Li [j3; c9; d20] investigated the synthesis of test cases for data processing systems and self-driving vehicles by adopting *constant abstraction* to represent unknown attribute values, which are then filled by an SMT solver [MB08]. This approach introduces *refinement units* as a means to unify various reasoning techniques with partial graph models and abstract domains.

4.2 Reasoning with partial models

The second group of contributions in this thesis is aimed at providing reasoning capabilities over partial models for extra-functional analyses.

Contribution 2.1 [c7; d21] proposes a novel *view transformation language* and a *transformation engine* as an open source prototype implementation primarily aimed at the construction of analysis models from architecture models.

The transformation language provides *parallel composition* of transformation to enable the integration of knowledge (such as the dependability attributes of various system components in **Challenge 3**) from multiple experts. We introduced *relation-based composition*, where the composition can be fine-tuned by providing a *glue transformation* without having to modify the composed transformations.

Thanks to the change-driven execution of the underlying VIATRA *Query* and *Transformation* framework [Ujh+15], a *reactive* and target *incremental* transformation engine is obtained. Exploiting the *inconsistency-tolerance* of 4-valued partial models from *Contribution 1.1*, the proposed transformation engine remains *validating* (**Challenge 2**) and explicitly marks inconsistencies even if the composed transformations or the state of the source model are not consistent. An implicitly constructed *traceability model* links source and target elements so that analysis results can be *back-annotated* to the architecture model.

Contribution 2.2 [j1] targets the synthesis of design candidates with multiplicity constraints (i.e., structural constraints with lower or upper bounds on the number of model objects or relationships) in **Challenge 5**. The proposed model generator also supports an extended version of *class scopes*, which are a popular approach for constraining model generation problems introduced by Alloy [Jac02].

More complex logical well-formedness constraints, as in **Challenge 2**, can also be handled by manual translation to linear inequalities and scoped partial models (*Contribution 1.2*).

Numerical reasoning is provided by Linear Programming (LP) and Integer Linear Programming (ILP) solvers, such as [Clp; Cbc]. Based on these low-level background theories, we provide *scope propagation* operators for scoped partial models that *refine* under- and over-approximations for partial model metrics. The refined approximations for metrics are used to *prune* the search space (**Challenge 1**) in an abstract DPLL [NOT06; Bra+13] decision procedure.

Contribution group 2 I proposed two reasoning techniques for deriving quantitative extra-functional metrics from partial models of systems and considering multiplicity constraints on architecture models.

Contribution 2.1 I proposed a fully compositional view transformation language and developed a reactive, incremental, validating, and inconsistency-tolerant view transformation engine for executing view transformations using 4-valued partial models [c7; d21].

2.1.1. I identified the levels of *parallel composition* support in view transformation languages, as well as the *consistent* and *validating* properties of view transformation engines.

2.1.2. I defined a *fully compositional view transformation language.*

2.1.3. I proposed a *reactive, incremental, validating, and inconsistecy-tolerant* transformation engine for unidirectional view transformations based on 4-valued partial models.

2.1.4. I evaluated the practical applicability and scalability of the approach using the open source Train Benchmark framework.

Contribution 2.2 I proposed a model generation approach that combines a DPLL-like decision procedure based on partial modeling with LP and ILP background solvers to synthesize and optimize models according to objective functions defined as linear programs, such as *multiplicity constraints*, *total model size*, and *cost functions*, using scoped partial models [j1].

2.2.1. I defined a mapping of *structural* and *well-formedness constraints* into linear numerical constraints that can *under-* and *over-approximated* on scoped partial models to efficiently guide model generation.

2.2.2. I extended an open-source graph solver with scoped partial model support by integrating LP and ILP solvers for numerical reasoning.

2.2.3. I evaluated the effectiveness of the approach using multiple industrial case studies, including a satellite constellation synthesis task from NASA Jet Propulsion Laboratory.

Added value To our best knowledge, the presented transformation engine is the first to support *reactive*, *validating*, *inconsistency-tolerant* execution of view transformations with *parallel composition* [c7]. This allows for the collaboration of multiple domain experts in the design of transformations, and the combination of transformations relating to multiple viewpoints, while still enabling the validation of target model structural constraints and the incremental maintenance of the target model according to source model updates.

In our empirical evaluations, reasoning with scoped partial models provided a *7-fold reduction in running time* for model generation in problems with complex multiplicity constraints [j1]. Even in domains without such constraints, the overhead added by scope propagation is minimal. While unsatisfiable problems usually pose a difficulty to model generators based on partial modeling, scope propagation operators can efficiently detect unsatisfiable multiplicity constraints with LP and ILP background solvers, leading to much better scalability.

Applications and related contributions Contributions in this contribution group were published as part of joint work with Dániel Varró and Oszkár Semeráth.

The proposed ViewModel view transformation engine is available as an open source project under the Eclipse Public License 1.0 at `https://github.com/ftsrg/viewmodel` and in [d21]. The tools has been applied for automatically constructing Stochastic Petri Net (SPN) reliability analysis models for design candidates of a redundant automotive subsystem in an industrial project with *thyssenkrupp Hungary Kft.*

The implementation of the proposed scope propagation strategies is available as part of the open source Viatra-Generator model generation framework under the Eclipse Public License 1.0 at `https://github.com/viatra/VIATRA-Generator`.

Joint work also with Gábor Szárnyas, Aren A. Babikian, Boqi Chen, and Chuning Li investigated the generation of *realistic test cases* for modelling tools by focusing the generator towards practically relevant corner cases [j4]. To ensure the realistic distribution of model elements by type, we relied on numerical reasoning techniques introduced in this contribution group.

Related contributions by my student Máté Földiák focused on numerical reasoning with reliability and performability metrics *directly over partial models* (instead of constructing separate analysis models) by computing under- and over-approximations. An initial prototype [c10; d22] was created that synthesizes optimal satellite constellation mission architectures [Her+17] according to performability objectives based on the *refinement unit* framework [j3; c9].

Joint work also with Boqi Chen and Sebastian Pilarski investigated the use logic reasoning with graphs for ensuring consistency in image recognition [c11].

4.3 Model-based quantitative extra-functional analysis

The third group of contributions presents concrete applications of partial model based reasoning for extra-functional analysis problems.

Contribution 3.1 [c12; l18; r19] provides a model-based technique for the automatic performability evaluation (**Challenge 3**) of reconfigurable system architectures based on Stochastic Petri Net (SPN) analysis models. As a specification language for performability properties, we adopt the incremental view transformations from *Contribution 2.1* to derive SPN models for each possible configuration.

We introduced *mission automata* as an extension of the *Graph Transformation Abstract State Machine* (GT+ASM) [VB07] formalism. The possible system failures are described by the *observable* attributes of a *run-time* version of the architecture model, which are defined based on the state of the SPN model and the traceability relationships between the architecture and SPN models. The mission automaton reacts to failures by changing the structure or the *controllable* attributes of the run-time model with graph transformations. We construct a *Phased-Mission System* (PMS) [MB99] from the SPN analysis models, where the reachable state space of the mission automaton serves as the high-level phase model.

Contribution 3.2 [j5] presents *witness model synthesis* in the domain-specific *Worst-Case Execution Time* (WCET) analysis for *graph query based runtime monitor programs* (**Challenge 4**). This allows us to tackle (i) WCET based on a single runtime snapshot, (ii) WCET based on the metamodel and well-formedness constraints of the runtime snapshots, and (iii) WCET based on a *partial* runtime snapshot.

Our analysis combines low-level *Implicit Path Enumeration Technique* (IPET) WCET analysis with a high-level, *domain specific analysis* that takes into account the well-formedness constraints of runtime models. We adapted the low-level IPET analyses from the OTAWA [Bal+10] and aiT WCET analysis tools.

To obtain a high-level analysis, we associate *auxiliary graph queries* based on the runtime monitor program with the variables of the IPET Integer Program to form a graph-based description of the WCET analysis problem. We synthesize the witness model using the extended generator from *Contribution 2.2*.

Contribution group 3 I proposed analysis methods for the reliability and worst-case execution time estimation based on reasoning with partial models.

Contribution 3.1 I proposed a model-based technique for automatically deriving phased-mission stochastic Petri net models for complex reconfigurable systems based on fully compositional view transformations [c12; l18; r19].

3.1.1. I defined *mission automata*, a formalism that leverages graph transformation abstract state machines (GT+ASM), as well as *observable* and *controllable* runtime features in architecture models, for a high-level description of runtime reconfigurations in cyber-physical systems.

3.1.2. I proposed a technique to construct *phased-mission stochastic Petri nets* for the dependability analysis of reconfigurable cyber-physical systems, where the reconfigurations are captured by a mission automaton, while the reliability processes of the system are described by individual stochastic Petri nets automatically constructed from the system architecture models via a view transformation.

3.1.3. I evaluated the practical applicability and scalability of the approach on a case study based on the analysis of a reconfigurable production cell.

Contribution 3.2 I proposed an approach for finding *witness models* of worst-case execution times of query-based runtime monitor programs in critical embedded systems by model generation using scoped partial models with linear programming [j5].

3.2.1. I proposed a *high-level static analysis* technique for query-based runtime monitor programs to estimate execution time on a given runtime model snapshot that combines *domain-specific* information from query plans with state-of-the-art IPET *low level analysis* output about the microarchitectural characteristics of the target execution platform.

3.2.2. I formulated the *witness generation* problem as a model generation task with a linear program objective.

3.2.3. I evaluated the practical applicability and scalability of the proposed approach by generating witness models for queries from the open source Train Benchmark in the context of the MoDeS3 CPS demonstrator.

Added value Model-based phased-mission system generation is an efficient, architecture-based analysis method for reconfigurable CPS. In our experiments, analysis models with up to 1028 different configurations could be constructed by exploring the state space of mission automata within 20 sec.

In the MoDeS3 CPS demonstrator, domain-specific WCET analysis reduced the metamodel-based WCET estimate with up to 12% compared to aiT and 25% compared to OTAWA on queries where the well-formedness constraints of the possible runtime snapshots impacted the execution time. It also managed to provide partial model based WCET estimates, while aiT provided an incorrect (lower than the actual program execution time of hardware) estimate due to its lack of partial model based estimate support.

Applications and related contributions Contributions in this contribution group were published as part of joint work with Dániel Varró, Brett H. Meyer, and Márton Búr. The concept of a *witness model* [j5] was introduced by Búr [Búr21].

Our background work related to dependability and performability analysis published jointly also with Miklós Telek, Tamás Bartha, Dániel Darvas, Ákos Hajdu, Attila Klenik, and Vince Molnár focused on the numerical solution of *Generalized Stochastic Petri Net* (GSPN) models. In [c13; c14], we proposed a configurable GSPN analysis framework based on the *block Kronecker decomposition*. In [c15], we proposed a symbolic state-space exploration engine for GSPN models. The analysis framework is available as part of the PetriDotNet [j6] tool at `https://inf.mit.bme.hu/en/research/tools/petridotnet` and was applied for the reliability analysis of a redundant automotive subsystem in an industrial project with *thyssenkrupp Hungary Kft.*

Related contributions by my student Simon Nagy investigated hierarchical reliability modeling and analysis based on statecharts and probabilistic programming [c16]. The resulting tool is available as part of the open source *Gamma Statechart Composition Framework* under the Eclipse Public License 1.0 at `https://github.com/FTSRG/gamma`. The tool was also used in our industrial project and a case study model adapted from the project is available in the *Models for Formal Analysis of Real Systems* (MARS) repository at `http://mars-workshop.org/repository/028-EPAS.html`. Related contributions by my student Dániel Szekeres investigated efficient numeric solution of large *Continuous-Time Markov*

Chains (CTMC) arising from *Static Fault Tree* (SFT) analysis using the *Tensor Train* decomposition for large matrices [c17]. An open source prototype implementation of the analysis tool is available under the Apache License 2.0 at `https://github.com/ftsrg/StoATT`.

5 Future work

Our long-term research goal is to *facilitate the design of complex system architectures* according to various, stringent functional and extra-functional requirements by *unifying and developing reasoning techniques* for their analysis and synthesis.

In particular, we highlight the following open challenges for future research:

- *Integration of reasoning techniques within sound and complete design-space exploration.* While this work addressed the use of 4-valued logic [Bel77][c7; j2] to explicitly highlight inconsistencies in architecture and analysis models, as well as polyhedron abstraction [CH78; BHZ08][j1] to reason about model size constraints and other linear inequalities, the simultaneous use of such abstractions remains unresolved. More broadly, the use of relational abstractions [Min04] or other abstract domains could lead to more efficient design-space exploration by pruning unsuitable design candidates earlier.

- *Integration of external extra-functional analysis and reasoning tools.* Partial model based analysis approaches can reason about many variants of a given design at once by explicitly encoding design decisions that are yet to be made. Hence, feedback about candidate architectures can be provided early in the design process. An initial attempt at integrating existing analyses as *refinement units* was made in [j3; c9], where external SMT solvers [MB08] are used to determine attribute values, and in [c10], where the performability of architectures is estimated with Markov chains. Nevertheless, further research is needed to improve analysis performance and explore the set of extra-functional analyses which can be executed on partial architectures.

- *Reasoning about system behavior during adaptation at runtime.* Self-adaptive strategies for run-time adaptation, where new system configurations are synthesized at runtime in response to failure events of the system, *require quantitative verification of the configurations at runtime* [Cal+12]. Sound and complete design-space exploration techniques for configuration synthesis could support provably sound adaptation strategies.

Publications

Number of publications	23
Number of peer-reviewed journal papers (written in English)	6
Number of articles in journals indexed by WoS or Scopus	5
Number of publications (in English) with at least 50% contribution	5
Number of peer-reviewed publications	21
Number of independent citations (as of January 2023)	98

Publications related to the contributions

	Journal papers	International conference and workshop papers	Reports and artifacts
Group 1	[j1]*, [j2], [j3]	[c7]*, [c8], [c9]	[d20]
Group 2	[j1]*, [j4]	[c7]*, [c10], [c11]	[d21], [d22]
Group 3	[j5], [j6]	[c12], [c13], [c14], [c15], [c16], [c17]	[l18], [r19]

*Publication related to multiple contribution groups

This classification follows the PhD publication scoring system of the faculty.

Journal papers

[j1] **Kristóf Marussy**, Oszkár Semeráth, and Dániel Varró. "Automated Generation of Consistent Graph Models with Multiplicity Reasoning". In: *IEEE Trans. Softw. Eng.* 48.5 (2022), pp. 1610–1629. DOI: 10.1109/TSE.2020.3025732

[j2] **Kristóf Marussy**, Oszkár Semeráth, Aren A. Babikian, Dániel Varró. "A Specification Language for Consistent Model Generation based on Partial Models". In: *J. Obj. Technol.* 19.3, 12 (2021). DOI: 10.5381/jot.2020.19.3.a12

[j3] Aren A. Babikian, Oszkár Semeráth, Anqi Li, **Kristóf Marussy**, Dániel Varró. "Automated Generation of Consistent Models by Combining Geometric and Qualitative Reasoning". In: *Softw. Syst. Model.* (2021). DOI: 10.1007/s10270-021-00918-6

[j4] Oszkár Semeráth, Aren A. Babikian, Boqi Chen, Chuning Li, **Kristóf Marussy**, Gábor Szárnyas, Dániel Varró. "Automated generation of consistent, diverse and structurally realistic graph models". In: *Softw. Syst. Model.* (2021). DOI: 10.1007/s10270-021-00884-z

[j5] Márton Búr, **Kristóf Marussy**, Brett H. Meyer, Dániel Varró. "Worst-Case Execution Time Calculation for Query-Based Monitors by Witness Generation". In: *ACM Trans. Embed. Comput. Syst.* 20.6 (2021), pp. 1–36. DOI: 10.1145/3471904. arXiv: 2102.03116 [cs.SE]

[j6] András Vörös, Dániel Darvas, Ákos Hajdu, Attila Klenik, **Kristóf Marussy**, Vince Molnár, Tamás Bartha, István Majzik. "Industrial applications of the PetriDotNet modelling and analysis tool". In: *Sci. Comput. Program.* 157 (2018), pp. 17–40. DOI: 10.1016/j.scico.2017.09.003

International conference and workshop papers

[c7] **Kristóf Marussy**, Oszkár Semeráth, and Dániel Varró. "Incremental View Model Synchronization Using Partial Models". In: *Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems.* ACM, 2018, pp. 323–333. DOI: 10.1145/3239372.3239412

[c8] István Majzik, Oszkár Semeráth, Csaba Hajdu, **Kristóf Marussy**, Zoltán Szatmári, Zoltán Miskei, András Vörös, Aren A. Babikian, Dániel Varró. "Towards System-Level Testing with Coverage Guarantees for Autonomous Vehicles". In: *ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems.* IEEE, 2019. DOI: 10.1109/MODELS.2019.00-12

[c9] Oszkár Semeráth, Aren A. Babikian, Anqi Li, **Kristóf Marussy**, Dániel Varró. "Automated generation of consistent models with structural and attribute constraints". In: *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems.* ACM, 2020, pp. 187–199. DOI: 10.1145/3365438.3410962

[c10] Máté Földiák, **Kristóf Marussy**, Dániel Varró, István Majzik. "System Architecture Synthesis for Performability by Logic Solvers". In: *Proceedings of the 25th ACM / IEEE International Conference on Model Driven Engineering Languages and Systems.* ACM, 2022. DOI: 10.1145/3550355.3552448

[c11] Boqi Chen, **Kristóf Marussy**, Sebastian Pilarski, Oszkár Semeráth, Dániel Varró. "Consistent Scene Graph Generation by Constraint Optimization". In: *37th IEEE/ACM International Conference on Automated Software Engineering.* ACM, 2022. DOI: 10.1145/3551349.3560433

[c12] **Kristóf Marussy** and István Majzik. "Constructing Dependability Analysis Models of Reconfigurable Production Systems". In: *IEEE 14th International Conference on Automation Science and Engineering.* IEEE, 2018. DOI: 10.1109/COASE.2018.8560551

[c13] **Kristóf Marussy**, Attila Klenik, Vince Molnár, András Vörös, István Majzik, Miklós Telek. "Efficient Decomposition Algorithm for Stationary Analysis of Complex Stochastic Petri Net Models". In: *PETRI NETS 2016.* LNCS 9698. Springer, 2016, pp. 281–300. DOI: 10.1007/978-3-319-39086-4_17

[c14] **Kristóf Marussy**, Attila Klenik, Vince Molnár, András Vörös, Miklós Telek, István Majzik. "Configurable numerical analysis for stochastic systems". In: *2016 International Workshop on Symbolic and Numerical Methods for Reachability Analysis (SNR).* IEEE, 2016. DOI: 10.1109/SNR.2016.7479383

[c15] **Kristóf Marussy**, Vince Molnár, András Vörös, István Majzik. "Getting the Priorities Right: Saturation for Prioritised Petri Nets". In: *PETRI NETS 2017.* LNCS 10258. Springer, 2017, pp. 223–242. DOI: 10.1007/978-3-319-57861-3_14

[c16] Simon József Nagy, Bence Graics, **Kristóf Marussy**, András Vörös. "Simulation-based Safety Assessment of High-level Reliability Models". In: *Proceedings of the 4th Workshop on Models for Formal Analysis of Real Systems.* EPTCS 316. 2020. DOI: 10.4204/EPTCS.316.9. arXiv: 2004.13290 [cs.SE]

[c17] Dániel Szekeres, **Kristóf Marussy**, and István Majzik. "Tensor-based reliability analysis of complex static fault trees". In: *17th European Dependable Computing Conference.* IEEE, 2021. DOI: 10.1109/EDCC53658.2021.00012

Local workshop paper

[l18] **Kristóf Marussy** and István Majzik. "Architecture-based Dependability Analysis of Reconfigurable and Adaptive Systems". In: *Proceedings of the 26th Minisymposium of the Department of Measurement and Information Systems.* BME Méréstechnika és Információs Rendszerek Tanszék, 2019

Technical report (not peer reviewed)

[r19] **Kristóf Marussy** and István Majzik. *Constructing Phased-Mission Systems for Dependability Analysis of Reconfigurable Production Systems.* Tech. rep. 2018. URL: `http://doi.org/10.5281/zenodo.1290661`

Artifacts (peer reviewed as part of the corresponding publications)

[d20] Oszkár Semeráth, Aren A. Babikian, Anqi Li, **Kristóf Marussy**, Dániel Varró. *Artifacts for "Automated Generation of Consistent Models with Structural and Attribute Constraints".* Approved as *reuseable* by the MODELS '20 Artifact Evaluation Committee [c9]. 2020. DOI: `10.5281/zenodo.3950552`

[d21] **Kristóf Marussy**, Oszkár Semeráth, and Dániel Varró. *ViewModel Tool and Benchmark Results for "Incremental View Model Synchronization Using Partial Models".* Approved by the MODELS '18 Artifact Evaluation Committee [c7]. 2018. DOI: `10.5281/zenodo.1318156`

[d22] Máté Földiák, **Kristóf Marussy**, Dániel Varró, István Majzik. *Artifacts for "System Architecture Synthesis for Performability by Logic Solvers".* Approved as *functional* by the MODELS '22 Artifact Evaluation Committee [c10]. 2022. DOI: `10.5281/zenodo.6974248`

Additional publications (not related to the contributions)

Book chapter

[b23] Nenad Tomašev, Krisztián Buza, **Kristóf Marussy**, Piroska Buzáné Kis. "Hubness-Aware Classification, Instance Selection and Feature Construction: Survey and Extensions to Time-Series". In: *Feature Selection for Data and Pattern Recognition.* SCI 584. Springer, 2015, pp. 231–261. DOI: `10.1007/978-3-662-45620-0_11`

International conference and workshop papers

[c24] **Kristóf Marussy** and Krisztián Buza. "SUCCESS: A New Approach for Semi-supervised Classification of Time-Series". In: *ICAISC 2013.* LNCS 7894. Springer, 2013, pp. 437–447. DOI: `10.1007/978-3-642-38658-9_39`

[c25] Krisztián Buza, Júlia Koller, and **Kristóf Marussy**. "PROCESS: Projection-Based Classification of Electroencephalograph Signals". In: *ICAISC 2015.* LNCS 9120. Springer, 2015, pp. 91–100. DOI: `10.1007/978-3-319-19369-4_9`

[c26] **Kristóf Marussy**, Ladislav Peska, and Krisztián Buza. "Recommendations of Unique Items Based on Bipartite Graphs". In: *Proceedings of the 9th Hungarian-Japanese Symposium on Discrete Mathematics and Its Applications.* Kyushu University, 2015

Local workshop paper

[l27] **Kristóf Marussy** and Krisztián Buza. “Hubness-based indicators for semi-supervised time-series classification”. In: *8th Japanese-Hungarian Symposium on Discrete Mathematics and Its Applications*. BME, 2013

Poster

[a28] **Kristóf Marussy** and Krisztián Buza. “PROGRESS: Projection-Based Gene Expression Classification”. In: *Innovations in Medicine Conference*. Akadémiai Kiadó, 2014

References

[Abd+14] Hani Abdeen et al. “Multi-objective optimization in rule-based design space exploration”. In: *ASE ’14*. ACM, 2014, pp. 289–300. DOI: `10.1145/2642937.2643005`.

[Abe+15] Jaume Abella et al. “WCET analysis methods: Pitfalls and challenges on their trustworthiness”. In: *10th IEEE International Symposium on Industrial Embedded Systems*. IEEE, 2015, pp. 39–48. DOI: `10.1109/SIES.2015.7185039`.

[ADW16] Ahmad Salim Al-Sibahi, Aleksandar S. Dimovski, and Andrzej Wasowski. “Symbolic execution of high-level transformations”. In: *SLE*. Springer, 2016, pp. 207–220.

[Agr+02] Aditya Agrawal et al. “Generative Programming via Graph Transformations in the Model-Driven Architecture”. In: *Workshop on Generative Techniques in the Context of Model Driven Architecture, OOPSLA*. 2002.

[Ale+09] Aldeida Aleti et al. “ArcheOpterix: An extendable tool for architecture optimization of AADL models”. In: *MOMPES*. IEEE, 2009, pp. 61–71. DOI: `10.1109/MOMPES.2009.5069138`.

[Anj+14] Anthony Anjorin et al. “Efficient Model Synchronization with View Triple Graph Grammars”. In: *ECMFA 2014*. Springer, 2014. DOI: `10.1007/978-3-319-09195-2_1`.

[APV09] Saswat Anand, Corina S. Păsăreanu, and Willem Visser. “Symbolic execution with abstraction”. In: *Int. J. Softw. Tools Technol. Transf.* 11.1 (2009), pp. 53–67. DOI: `10.1007/s10009-008-0090-1`.

[Arc+18] Davide Arcelli et al. “EASIER: An Evolutionary Approach for Multi-objective Software ArchItecturE Refactoring”. In: *ISCA*. IEEE, 2018, pp. 105–114. DOI: `10.1109/ICSA.2018.00020`.

[Are+10] Thorsten Arendt et al. “Henshin: Advanced Concepts and Tools for In-Place EMF Model Transformations”. In: *MODELS 2010*. Springer, 2010, pp. 121–135. DOI: `10.1007/978-3-642-16145-2_9`.

[Bal+10] Clément Ballabriga et al. “OTAWA: An Open Toolbox for Adaptive WCET Analysis”. In: *Software Technologies for Embedded and Ubiquitous Systems*. LNCS 6399. Springer, 2010, pp. 35–46. DOI: `10.1007/978-3-642-16256-5_6`.

[Ban+13] Kshitij Bansal et al. “Structural Counter Abstraction”. In: *TACAS 2013*. Vol. 7795. LNCS. Springer, 2013, pp. 62–77. DOI: `10.1007/978-3-642-36742-7_5`.

[Bar+18] Ezio Bartocci et al. “Specification-Based Monitoring of Cyber-Physical Systems: A Survey on Theory, Tools and Applications”. In: *Lectures on Runtime Verification*. Springer, 2018, pp. 135–175.

[BBF09] Gordon S. Blair, Nelly Bencomo, and Robert B. France. “Models@run.time”. In: *IEEE Computer* 42.10 (2009), pp. 22–27. DOI: 10.1109/MC.2009.326.

[BDD04] Simona Bernardi, Susanna Donatelli, and Giovanna Dondossola. “Towards a Methodological Approach to Specification and Analysis of Dependable Automation Systems”. In: Springer, 2004, pp. 36–51. DOI: 10.1007/978-3-540-30206-3_5.

[Bel77] Nuel D. Belnap. “A Useful Four-Valued Logic”. In: *Modern Uses of Multiple-Valued Logic*. Springer, 1977, pp. 5–37. DOI: 10.1007/978-94-010-1161-7_2.

[Ber+11] Gábor Bergmann et al. “A Graph Query Language for EMF models”. In: *ICMT*. LNCS 6707. Springer, 2011, pp. 167–182. DOI: 10.1007/978-3-642-21732-6_12.

[Ber+12] Gábor Bergmann et al. “Change-driven model transformations”. In: *Softw. Syst. Model.* 11.3 (2012), pp. 431–461. DOI: 10.1007/s10270-011-0197-9.

[BFK19] Axel Busch, Dominik Fuchss, and Anne Koziolek. “PerOpteryx: Automated Improvement of Software Architectures”. In: *ICSA*. IEEE, 2019, pp. 162–165. DOI: 10.1109/ICSA-C.2019.00036.

[BHZ08] Roberto Bagnara, Particia M. Hill, and Enea Zaffanella. “The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems”. In: *Sci. Comput. Program.* 72.1-2 (2008), pp. 3–21. DOI: 10.1016/j.scico.2007.08.001.

[BKR08] Steffen Becker, Heiko Koziolek, and Ralf Reussner. “The Palladio component model for model-driven performance prediction”. In: *J. Sys. Softw.* 82.1 (2008), pp. 3–22. DOI: 10.1016/j.jss.2008.03.066.

[BMM99] Andrea Bondavalli, Ivan Mura, and István Majzik. “Automatic Dependability Analysis for Supporting Design Decisions in UML”. In: *Proc. 4th IEEE Int. Symp. High-Assur. Syst. Eng.* IEEE, 1999, pp. 64–74. DOI: 10.1109/HASE.1999.809476.

[BMP12] Simona Bernardi, José Merseguer, and Dorina C. Petriu. “Dependability modeling and analysis of software systems specified with UML”. In: *ACM Comput. Surv.* 45.1 (2012). DOI: 10.1145/2379776.2379778.

[BPF15] Nikolaj Bjørner, Anh-Dung Phan, and Lars Fleckenstein. “νZ – An Optimizing SMT Solver”. In: *TACAS*. Vol. 9035. LNCS. Springer, 2015, pp. 194–199. DOI: 10.1007/978-3-662-46681-0_14.

[Bra+13] Martin Brain et al. “An Abstract Interpretation of DPLL(T)”. In: *VMCAI 2013*. LNCS 7737. Springer, 2013, pp. 455–475. DOI: 10.1007/978-3-642-35873-9_27.

[Búr+18] Márton Búr et al. “Distributed graph queries for runtime monitoring of cyber-physical systems”. In: LNCS 10802. 2018, pp. 111–128. DOI: 10.1007/978-3-319-89363-1_7.

[Búr+20] Márton Búr et al. “Distributed graph queries over models@run.time for runtime monitoring of cyber-physical systems”. In: *Int. J. Softw. Tools Technol. Transf.* 22 (2020), pp. 79–102. DOI: 10.1007/s10009-019-00531-5.

[Búr21] Márton Búr. “Query-Based Runtime Monitoring in Real-Time and Distributed Systems”. PhD thesis. McGill University, 2021. URL: https://imbur.github.io/phd/marton-bur-thesis.pdf.

[BZJ21] Alexandru Burdusel, Steffen Zschaler, and Stefan John. "Automatic generation of atomic multiplicity-preserving search operators for search-based model engineering". In: *Softw. Syst. Model.* 20.6 (2021), pp. 1857–1887. DOI: 10.1007/s10270-021-00914-w.

[BZS18] Alexandru Burdusel, Steffen Zschaler, and Daniel Strüber. "MDEoptimiser: A Search Based Model Engineering Tool". In: *MODELS*. ACM, 2018, pp. 12–16. DOI: 10.1145/3270112.3270130.

[CA05] Krzysztof Czarnecki and Michał Antkiewicz. "Mapping Features to Models: A Template Approach Based on Superimposed Variants". In: *GPCE*. Springer, 2005, pp. 422–437. DOI: 10.1007/11561347_28.

[Cal+12] Radu Calinescu et al. "Self-Adaptive Software Needs Quantitative Verification at Runtime". In: *Comm. ACM* 55.9 (2012), pp. 69–77. DOI: 10.1145/2330667.2330686.

[Cal+17] Radu Calinescu et al. "RODES: A Robust-Design Synthesis Tool for Probabilistic Systems". In: *QEST*. Vol. 10503. LNCS. Springer, 2017, pp. 304–308. DOI: 10.1007/978-3-319-66335-7_20.

[Cbc] COIN-OR. *Cbc*. URL: https://github.com/coin-or/Cbc.

[CCR07] Jordi Cabot, Robert Clarisó, and Daniel Riera. "UMLtoCSP: a tool for the formal verification of UML/OCL models using constraint programming". In: *ASE*. ACM, 2007, pp. 547–548. DOI: 10.1145/1321631.1321737.

[CET18] Vittorio Cortellessa, Romina Eramo, and Michele Tucci. "Availability-Driven Architectural Change Propagation Through Bidirectional Model Transformations Between UML and Petri Net Models". In: *ICSA*. IEEE, 2018. DOI: 10.1109/ICSA.2018.00022.

[CET20] Vittorio Cortellessa, Romina Eramo, and Michele Tucci. "From software architecture to analysis models and back: Model-driven refactoring aimed at availability improvement". In: *Inf. Softw. Technol.* 127 (2020), p. 106362. DOI: 10.1016/j.infsof.2020.106362.

[CH78] Patrick Cousot and Nicolas Halbwachs. "Automatic discovery of linear restraints among variables of a program". In: *POPL*. ACM, 1978, pp. 84–96. DOI: 10.1145/512760.512770.

[Che+11] Betty H. C. Cheng et al. "Using Models at Runtime to Address Assurance for Self-Adaptive Systems". In: *Models@run.time*. LNCS 8378. Springer, 2011, pp. 101–136. DOI: 10.1007/978-3-319-08915-7_4.

[CJ11] Duc Hiep Chu and Joxan Jaffar. "Symbolic simulation on complicated loops for WCET path analysis". In: *Proc. of the 9th ACM International Conference on Embedded Software*. IEEE, 2011, pp. 319–328. DOI: 10.1145/2038642.2038692.

[Clp] COIN-OR. *Clp*. URL: https://github.com/coin-or/Clp.

[CMP15] Vittorio Cortellessa, Raffaela Mirandola, and Pasqualina Potena. "Managing the evolution of a software architecture at minimal cost under performance and reliability constraints". In: *Sci. Comput. Program.* 98 (2015), pp. 439–463. DOI: 10.1016/j.scico.2014.06.001.

[CNS12] Marsha Chechik, Shiva Nejati, and Mehrad Sabetzadeh. "A relationship-based approach to model integration". In: *Innov. Syst. Softw. Eng.* 8.1 (2012), pp. 3–18. DOI: 10.1007/s11334-011-0155-2.

[CT93] Gianfranco Ciardo and Kishor S. Trivedi. "A decomposition approach for stochastic reward net models". In: *Perform. Eval.* 18.1 (1993), pp. 37–59. DOI: 10.1016/0166-5316(93)90026-Q.

[DBB18] Wei Dou, Domenico Bianculli, and Lionel Briand. "Model-Driven Trace Diagnostics for Pattern-Based Temporal Specifications". In: *21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems.* ACM, 2018, pp. 278–288. DOI: 10.1145/3239372.3239396.

[DLL62] Martin Davis, George Logemann, and Davis Loveland. "A machine program for theorem-proving". In: *C. ACM* 5.7 (1962), pp. 394–397. DOI: 10.1145/368273.368557.

[DXC11] Zinovy Diskin, Yingfei Xiong, and Krzysztof Czarnecki. "Specifying Overlaps of Heterogeneous Models for Global Consistency Checking". In: *MODELS 2010.* Springer, 2011, pp. 165–179. DOI: 10.1007/978-3-642-21210-9_16.

[ENS10] Brandon K. Eames, Sandeep Neema, and Rohit Saraswat. "DesertFD: a finite-domain constraint based tool for design space exploration". In: *Des. Autom. Embed. Syst.* 14.2 (2010), pp. 43–74. DOI: 10.1007/s10617-009-9049-z.

[Epi+09] Ilenia Epifani et al. "Model evolution by run-time parameter adaptation". In: *ICSE.* IEEE, 2009, pp. 111–121. DOI: 10.1109/ICSE.2009.5070513.

[Erm+07] Andreas Ermedahl et al. "Loop bound analysis based on a combination of program slicing, abstract interpretation, and invariant analysis". In: *7th International Workshop on Worst-Case Execution Time Analysis (WCET'07).* Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2007.

[FB16] Simon Fürst and Markus Bechter. "AUTOSAR for Connected and Autonomous Vehicles: The AUTOSAR Adaptive Platform". In: *DSN-W.* IEEE, 2016. DOI: 10.1109/DSN-W.2016.24.

[FD16] Peter Feiler and Julien Delange. "Automated Fault Tree Analysis from ADL Models". In: *ACM SIGAda Ada Letters* 36.2 (2016), pp. 39–46. DOI: 10.1145/3092893.3092900.

[Fel03] Massimo Felici. "Taxonomy of evolution and dependability". In: *Proc. 2nd Workshop Unanticip. Softw. Evol.* 2003, pp. 95–104.

[FFJ12] Pietro Ferrara, Rafael Fuchs, and Uri Juhasz. "TVAL+: TVLA and value analyses together". In: *SEFM.* ACM, 2012, pp. 63–77. DOI: 10.1007/978-3-642-33826-7_5.

[FTW16] Martin Fleck, Javier Troya, and Manuel Wimmer. "Search-Based Model Transformations with MOMoT". In: *ICMT@STAF.* LNCS 9765. Springer, 2016, pp. 79–87. DOI: 10.1007/978-3-319-42064-6_6.

[Get+18] Sinem Getir et al. "Supporting semi-automatic co-evolution of architecture and fault tree models". In: *J. Syst. Softw.* 142 (2018), pp. 115–135. DOI: 10.1016/j.jss.2018.04.001.

[Gha+17] Majdi Ghadhab et al. "Model-Based Safety Analysis for Vehicle Guidance Systems". In: *SAFECOMP.* Springer, 2017, pp. 3–19. DOI: 10.1007/978-3-319-66266-4_1.

[Gop+04] Denis Gopan et al. "Numeric Domains with Summarized Dimensions". In: *TACAS 2004.* Vol. 2988. LNCS. Springer, 2004, pp. 512–529. DOI: 10.1007/978-3-540-24730-2_38.

[GTC15] Simos Gerasimou, Giordano Tamburrelli, and Radu Calinescu. "Search-Based Synthesis of Probabilistic Models for Quality-of-Service Software Engineering". In: *ASE*. IEEE, 2015. DOI: 10.1109/ASE.2015.22.

[Gus+06] Jan Gustafsson et al. "Automatic derivation of loop bounds and infeasible paths for WCET analysis using abstract execution". In: *Proc. of Real-Time Systems Symposium*. 2006, pp. 57–66. DOI: 10.1109/RTSS.2006.12.

[Har+19] Thomas Hartmann et al. "GREYCAT: Efficient what-if analytics for data in motion at scale". In: *Information Systems* 83 (2019), pp. 101–117.

[Hav15] Klaus Havelund. "Rule-based runtime verification revisited". In: *Int. J. Software Tools Technol. Trans.* 17.2 (2015), pp. 143–170. DOI: 10.1007/s10009-014-0309-2.

[Heg+16] Ábel Hegedüs et al. "Query-driven soft traceability links for models". In: *Softw. Syst. Model.* 15.3 (2016), pp. 733–756. DOI: 10.1007/s10270-014-0436-y.

[Her+17] Sebastian J. I. Herzig et al. "Model-transformation-based computational design synthesis for mission architecture optimization". In: *IEEE Aerospace Conference*. IEEE, 2017. DOI: 10.1109/AERO.2017.7943953.

[HHV15] Ábel Hegedüs, Ákos Horváth, and Dániel Varró. "A model-driven framework for guided design space exploration". In: *Autom. Softw. Eng.* 22 (2015), pp. 399–436. DOI: 10.1007/s10515-014-0163-1.

[HR02] Klaus Havelund and Grigore Rosu. "Synthesizing Monitors for Safety Properties". In: *Tools and Algorithms for the Construction and Analysis of Systems*. LNCS 2280. 2002, pp. 342–356. DOI: 10.1007/3-540-46002-0_24.

[Iqb+15] Muhammad Zohaib Iqbal et al. "Applying UML/MARTE on industrial projects: challenges, experiences, and guidelines". In: *Softw. Syst. Model.* 14.4 (2015), pp. 1367–1385. DOI: 10.1007/s10270-014-0405-5.

[IW17] M. Usman Iftikhar and Danny Weyns. "ActivFORMS: A Runtime Environment for Architecture-Based Adaptation with Guarantees". In: *ICSAW*. IEEE, 2017. DOI: 10.1109/ICSAW.2017.21.

[Jac02] Daniel Jackson. "Alloy: a lightweight object modelling notation". In: *ACM Trans. Softw. Eng. Methodol.* 11.2 (2002), pp. 256–290. DOI: 10.1145/505145.505149.

[Joh+19] Stefan John et al. "Searching for Optimal Models: Comparing Two Encoding Approaches". In: *J. Object Technol.* 18.3 (2019), 6:1–22. DOI: 10.5381/jot.2019.18.3.a6.

[JVB17] Anjali Joshi, Steve Vestal, and Pam Binns. "Automatic generation of static fault trees from AADL models". In: *DSN Workshops*. Springer, 2017.

[Kat+16] Guy Katz et al. "Lazy proofs for DPLL(T)-based SMT solvers". In: *FMCAD*. IEEE, 2016, pp. 93–100. DOI: 10.1109/FMCAD.2016.7886666.

[KB09] Heiko Koziolek and Franz Brosch. "Parameter Dependencies for Component Reliability Specifications". In: *Elec. Note. Theor. Comput. Sci.* 253 (1 2009), pp. 23–38. DOI: 10.1016/j.entcs.2009.09.026.

[Ker+13] Aleksandr A Kerzhner et al. "Architecting cellularized space systems using model-based design exploration". In: *AIAA SPACE 2013 Conference and Exposition*. 2013, p. 5371.

[KHG11] Mirco Kuhlmann, Lars Hamann, and Martin Gogolla. "Extensive Validation of OCL Models by Integrating SAT Solving into USE". In: *TOOLS*. Vol. 6705. LNCS. 2011, pp. 290–306.

[KJS10] Eunsuk Kang, Ethan Jackson, and Wolfram Schulte. "An Approach for Effective Design Space Exploration". In: *Monterey Workshop*. Vol. 6662. LNCS. Springer, 2010, pp. 33–54. DOI: 10.1007/978-3-642-21292-5_3.

[KKZ13] Jens Knoop, Laura Kovács, and Jakob Zwirchmayr. "WCET squeezing". In: *International Conference on Real-Time Networks and Systems*. ACM, 2013, pp. 161–170. DOI: 10.1145/2516821.2516847.

[KO17] Norihiro Kamide and Hitoshi Omori. "An Extended First-Order Belnap-Dunn Logic with Classical Negation". In: *LORI*. Springer, 2017, pp. 79–93. DOI: 10.1007/978-3-662-55665-8_6.

[Koz10] Heiko Koziolek. "Performance evaluation of component-based software systems: A survey". In: *Perform. Eval.* 67.8 (2010), pp. 634–658. DOI: 10.1016/j.peva.2009.07.007.

[KR08] Heiko Koziolek and Ralf Reussner. "A Model Transformation from the Palladio Component Model to Layered Queueing Networks". In: (2008), pp. 58–57. DOI: 10.1007/978-3-540-69814-2_6.

[KR11] Anne Koziolek and Ralf Reussner. "Towards a generic quality optimisation framework for component-based system models". In: *CBSE*. ACM, 2011, pp. 103–108. DOI: 10.1145/2000229.2000244.

[Kul09] Ulrich W. Kulisch. "Complete Interval Arithmetic and Its Implementation of the Computer". In: *Numerical Validation in Current Hardware Architectures*. LNCS. Springer, 2009, pp. 7–26. DOI: 10.1007/978-3-642-01591-5_2.

[Li+11] Rui Li et al. "An evolutionary multiobjective optimization approach to component-based software architecture design". In: *IEEE CEC*. IEEE, 2011, pp. 432–439. DOI: 10.1109/CEC.2011.5949650.

[Li+14] Yi Li et al. "Symbolic optimization with SMT solvers". In: *POPL*. ACM, 2014, pp. 607–618. DOI: 10.1145/2535838.2535857.

[Lis14] Björn Lisper. "SWEET – a tool for WCET flow analysis". In: *International Symposium On Leveraging Applications of Formal Methods, Verification and Validation*. Springer. 2014, pp. 482–485.

[LM97] Y.-T.S. Li and Sharad Malik. "Performance analysis of embedded software using implicit path enumeration". In: *IEEE T. Comput. Aid. D.* 16.12 (1997), pp. 1477–1487. DOI: 10.1109/43.664229.

[LMC04] Juan Pablo López-Grao, José Merseguer, and Javier Campos. "From UML activity diagrams to Stochastic Petri nets: application to software performance engineering". In: *WOSP*. ACM, 2004, pp. 25–36. DOI: 10.1145/974043.974048.

[Mag+07] Stephen Magill et al. "Arithmetic Strengthening for Shape Analysis". In: *SAS*. Vol. 4634. LNCS. Springer, 2007, pp. 419–436. DOI: 10.1007/978-3-540-74061-2_26.

[Mai+17] Claire Maiza et al. "The W-SEPT Project: Towards Semantic-Aware WCET Estimation". In: *17th International Workshop on Worst-Case Execution Time Analysis (WCET 2017)*. Vol. 57. OpenAccess Series in Informatics (OASIcs). Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2017. DOI: 10.4230/OASIcs.WCET.2017.9.

[Mar+10] Anne Martens et al. "Automatically improve software architecture models for performance, reliability, and cost using evolutionary algorithms". In: *Proc. 1st Joint WOSP/SIPEW Int. Conf. Perf. Eng.* ACM, 2010, pp. 105–116. DOI: 10.1145/1712605.1712624.

[MB08] Leonardo de Moura and Nikolaj Bjørner. "Z3: An Efficient SMT Solver". In: *TACAS*. Vol. 4963. LNCS. Springer, 2008, pp. 337–340.

[MB15] Alexis Marechal and Didier Buchs. "Generalizing the Compositions of Petri Nets Modules". In: *Fundam. Inform.* 137.1 (2015), pp. 87–116. DOI: 10.3233/FI-2015-1171.

[MB99] I. Mura and A. Bondavalli. "Hierarchical modeling and evaluation of phased-mission systems". In: *IEEE Tran. Reliability* 48.4 (1999), pp. 360–368. DOI: 10.1109/24.814518.

[Min04] Antoine Miné. "Weakly Relational Numerical Abstract Domains". PhD thesis. École Normale Supérieure, 2004.

[MOG] The MOGENTES project. *Model-based generation of tests for dependable embedded systems, 7th EU Framework Programme*. 2011. URL: http://mogentes.eu/.

[MPB02] István Majzik, András Pataricza, and Andrea Bondavalli. "Stochastic Dependability Analysis of System Architecture Based on UML Models". In: *Architecting Dependable Systems*. Springer, 2002, pp. 219–244. DOI: 10.1007/3-540-45177-3_10.

[MRS10] Bill McCloskey, Thopas Reps, and Mooly Sagiv. "Statically Inferring Complex Heap, Array, and Numeric Invariants". In: *SAS*. Vol. 6337. LNCS. Springer, 2010, pp. 71–99. DOI: 10.1007/978-3-642-15769-1_6.

[MVS07] Panagiotis Manolios, Daron Vroon, and Gayatri Subramanian. "Automating component-based system assembly". In: *ISSTA*. ACM, 2007, pp. 61–72. DOI: 10.1145/1273463.1273473.

[Ndi+16] Moulaye Ndiaye et al. "Practical Use of Coloured Petri Nets for the Design and Performance Assessment of Distributed Automation Architectures". In: *PNSE*. CEUR-WS, 2016, pp. 113–131. URL: http://ceur-ws.org/Vol-1591/paper10.pdf.

[Nin15] Jordan Ninin. "Global Optimization Based on Contractor Programming: An Overview of the IBEX Library". In: *MACIS*. Vol. 9582. LNCS. Springer, 2015, pp. 555–559. DOI: 10.1007/978-3-319-32859-1_47.

[NIST17] Cyber-Physical Systems Public Working Group. *Framework for Cyber-Physical Systems: Volume 1, Overview*. NIST Special Publication 1500-201. National Institute of Standards and Technology, 2017. DOI: 10.6028/NIST.SP.1500-201.

[NO79] Greg Nelson and Derek C. Oppen. "Simplification by Cooperating Decision Procedures". In: *ACM Trans. Program. Lang. Syst.* 1.2 (1979), pp. 245–257. DOI: 10.1145/357073.357079.

[NOT06] Robert Nieuwenhuis, Albert Oliveras, and Cesar Tinelli. "Solving SAT and SAT Modulo Theories: From an abstract Davis-Putnam-Logemann-Loveland procedure to DPLL(T)". In: *J. ACM* 53.6 (2006), pp. 937–977. DOI: 10.1145/1217856.1217859.

[OCL] The Object Management Group. *Object Constraint Language, v2.4*. 2014. URL: https://www.omg.org/spec/OCL/2.4.

[Pen08] Karl-Heinz Pennemann. "Resolution-like theorem proving for high-level conditions". In: *ICGT*. Vol. 5214. LNCS. Springer, 2008, pp. 289–304.

[Pik+10] Lee Pike et al. "Copilot: A Hard Real-Time Runtime Monitor". In: *Runtime Verification*. Vol. 6418. 2010, pp. 345–359. DOI: 10.1007/978-3-642-16612-9_26.

[RD06] Arend Rensink and Dino Distefano. "Abstract graph transformation". In: *Electr. Notes Theor. Comput. Sci.* 157.1 (2006), pp. 39–59.

[REJ09] Derek Rayside, H.-Cristian Estler, and Daniel Jackson. *The guided improvement algorithm for exact, general-purpose, many-objective combinatorial optimization.* Tech. rep. MIT-CSAIL-TR-2009-033. Massachsetts Institue of Technology, 2009.

[Ren06] Arend Rensink. "Isomorphism Checking in GROOVE". In: *GaBaTS*. Vol. 4549. LNCS. Springer, 2006.

[Rey+13] Andrew Reynolds et al. "Finite Model Finding in SMT". In: *CAV*. Vol. 8044. LNCS. Springer, 2013, pp. 640–655. DOI: 10.1007/978-3-642-39799-8_42.

[RSW04] Thomas W Reps, Mooly Sagiv, and Reinhard Wilhelm. "Static program analysis via 3-valued logic". In: *CAV*. 2004, pp. 15–30.

[SE06] Mehrdad Sabetzadeh and Steve Easterbrook. "View merging in the presence of incompleteness and inconsistency". In: *Requir. Eng.* 11.3 (2006), pp. 174–193. DOI: 10.1007/s00766-006-0032-y.

[SLO17] Sven Schneider, Leen Lambers, and Fernando Orejas. "Symbolic model generation for graph properties". In: *FASE*. Vol. 10202. LNCS. Springer, 2017, pp. 226–243.

[SNP13] Jaroslaw Skaruz, Artur Niewiadomski, and Wojciech Penczek. "Evolutionary Algorithms for Abstract Planning". In: *PPAM*. LNTCS 8384. Springer, 2013, pp. 392–401. DOI: 10.1007/978-3-642-55224-3_37.

[SNV18] Oszkár Semeráth, András Szabolcs Nagy, and Dániel Varró. "A Graph Solver for the Automated Generation of Consistent Domain-Specific Models". In: *ICSE '18*. ACM, 2018, pp. 969–980.

[SPV18] Gagandeep Singh, Markus Püschel, and Martin Vechev. "A Practical Construction for Decomposing Numerical Abstract Domains". In: *Proc. ACM Program. Lang.* 2.POPL (2018). Article no. 2. DOI: 10.1145/3158143.

[SRA92] A. K. Somani, J. A. Ritcey, and S. H. L. Au. "Computationally-efficient phased-mission reliability analysis for systems with variable configurations". In: *IEEE Tran. Reliability* 41.4 (1992), pp. 504–511. DOI: 10.1109/24.249576.

[SSB20] Ghanem Soltana, Mehrdad Sabetzadeh, and Lionel C. Briand. "Practical Constraint Solving for Generating System Test Data". In: *ACM Trans. Softw. Eng. Methodol.* 29.2 (2020), 11:1–11:48. DOI: 10.1145/3381032.

[Ste+09] Dave Steinberg et al. *EMF: Eclipse Modeling Framework.* 2nd ed. Addison-Wesley Prof., 2009.

[SV17] Oszkár Semeráth and Dániel Varró. "Graph Constraint Evaluation over Partial Models by Constraint Rewriting". In: *ICMT*. 2017, pp. 138–154. DOI: 10.1007/978-3-319-61473-1_10.

[Szá+17] Gábor Szárnyas et al. "The Train Benchmark: cross-technology performance evaluation of continuous model queries". In: *Softw. Syst. Model.* (2017). DOI: 10.1007/s10270-016-0571-8.

[Ujh+15] Zoltán Ujhelyi et al. "EMF-IncQuery: An integrated development environment for live model queries". In: *Sci. Comput. Program.* 98.1 (2015), pp. 80–99. DOI: 10.1016/j.scico.2014.01.004.

[Var+18] Dániel Varró et al. “Towards the Automated Generation of Consistent, Diverse, Scalable and Realistic Graph Models”. In: *Graph Transformation, Specifications, and Nets – In Memory of Hartmut Ehrig*. LNCS 10800. Springer, 2018, pp. 285–312. DOI: 10.1007/978-3-319-75396-6_16.

[VB07] Dániel Varró and András Balogh. “The Model Transformation Language of the VIATRA2 Framework”. In: *Sci. Comput. Program.* 68.3 (2007), pp. 214–234. DOI: 10.1016/j.scico.2007.05.004.

[Vog+15] Birgit Vogel-Heuser et al. “Evolution of software in automated production systems: Challenges and research directions”. In: *J. Syst. Softw.* 110 (2015), pp. 54–84. DOI: 10.1016/j.jss.2015.08.026.

[Vör+18b] András Vörös et al. “MoDeS3: Model-Based Demonstrator for Smart and Safe Cyber-Physical Systems”. In: *NASA Formal Methods*. 2018, pp. 460–467.

[WT07] Dazhi Wang and Kishor S. Trivedi. “Reliability Analysis of Phased-Mission System With Independent Component Repairs”. In: *IEEE Tran. Reliability* 56.3 (2007), pp. 540–551. DOI: 10.1109/TR.2007.903268.

[Xia+11] Jianwen Xiang et al. “Automatic Synthesis of Static Fault Trees from System Models”. In: *SSIRI*. IEEE, 2011. DOI: 10.1109/SSIRI.2011.32.

[Xin07] Liudong Xing. “Reliability Evaluation of Phased-Mission Systems With Imperfect Fault Coverage and Common-Cause Failures”. In: *IEEE Tran. Reliability* 56.1 (2007), pp. 58–68. DOI: 10.1109/TR.2006.890900.