Project Description – Project Proposals

Prof. Dr. rer. nat. habil. Jan Peleska, Bremen

CPOT-SM – Complete Property-oriented Testing with Symbolic Methods

---

Project Description

1 State of the art and preliminary work

STATE OF THE ART

Complete model-based testing. The investigation of complete theories for testing against models has a long tradition. Initially, only finite state machine (FSM) models were considered. This started with the W-Method published by (Vasilevskii, 1973) and (Chow, 1978), who were the first to show that implementations whose true behavior is represented by completely specified deterministic FSMs with at most m states can be tested against reference models with $n \leq m$ states, such that any violation of language equivalence will be uncovered by finitely many test cases derived from the model. Since then, these initial results have been extended to incompletely specified and nondeterministic finite state machines, and to language inclusion as an alternative conformance relation to language equivalence. Moreover, many of the new theories guaranteed complete fault coverage with considerably less test cases than needed according to the W-Method, such as, for example, the Wp-Method (Fujiwara et al., 1991) and (Luo et al., 1994). As of today, the most effective complete FSM-based testing method for testing language equivalence is the H-Method published by (Dorofeeva et al., 2005). For testing nondeterministic FSMs w.r.t. language inclusion, adaptive testing methods are preferred, as published in (Hierons, 2004) and (Petrenko and Yevtushenko, 2011). Apart from FSM-based MBT, complete testing methods for many other semantic models or concrete modelling formalisms have been elaborated in the past. In (Gaudel, 1995), completeness criteria for algebraic formalism were stated; these criteria (uniformity hypothesis and regularity hypothesis) turned out to be generically applicable for arbitrary formalisms. For the class of formalisms based on labelled transition system (LTS) semantics, conformance testing methods based on the ioco relation (see, for example, (Tretmans, 1996) and (Brinksma and Tretmans, 2000)) seem to be the most widely used today. Much earlier, it has been shown in (Hennessy, 1988) that (not necessarily finite) test suites can be used to determine refinement relations between processes with LTS semantics. For concrete formalisms, the literature on MBT is vast. For process algebras, we name (Schneider, 1999) and (Schneider, 1995) for testing against (timed) CSP process algebra models, (Springintveld et al., 2001) for a complete testing theory based on Timed Automata, and the more recent work (Cavalcanti and Gaudel, 2011) and (Alberto et al., 2017) on testing against models specified in the Circus formalism which combines both data and control-oriented modelling techniques from the Z specification language and the CSP process algebra. Further examples are given below in the description of our own preliminary work. In modeling formalisms that are more general than finite state machines and use less restrictive fault models, it is not always guaranteed that finite complete test suites exist; see, for example (Cavalcanti and Simao, 2017), where examples requiring test cases of infinite length are presented for the CSP process algebra.

Symbolic methods and MBT. Similar to infinite-state model checking, symbolic methods are applied in MBT for reducing the size of a model's input domain, internal state space, or output

Prof. Dr. rer. nat. habil. Jan Peleska
University of Bremen
Department of Mathematics and Computer Science
Bibliothekstraße 1
28359 Bremen

domain. Apart from our own work in this field which is described below, the following contributions are significant for the exploration of this line of research. In (Clarke et al., 2002),, (Frantzen et al., 2005), (Jeannet et al., 2005), and (Jéron, 2009), first suggestions about test generation with symbolic methods were made. Full fault coverage, however, was only established for infinite test suites. In contrast to that, complete testing theories for symbolic state machine models requiring *finite* test suites have been elaborated in (Petrenko and Simao, 2015) and (Petrenko, 2016). They are applicable in a more general context and have less stringent requirements concerning the first-order predicates involved than our theories described below, but – not surprisingly – this generality leads to weaker fault models that do not admit, for example, SUTs with additional (faulty) states. In (Taromirad and Mousavi, 2017), our concepts for black-box equivalence class partition testing described below have been extended to gray-box testing: the authors show that considerable test suite reductions can be gained if it is not mandatory to perform black-box testing, but some internal information about the SUT can be obtained.

Property-oriented testing. The property-oriented testing (POT) approach has been characterized in the overview article (Machado et al., 2007). As stated there, the main motivation for POT is the possibility to reduce test suite size in comparison to a complete model-based conformance test suite: while the latter requires test cases to verify all possible behaviors in relation to a reference model, the former only requires investigating the subset of possible behaviors which is relevant to verify a certain set of properties (i.e. requirements).
A typical approach to property-oriented testing is to state desired system requirements as formulas in a temporal logic (see e.g. (Fernandez et al., 2003) and (Li and Qi, Zhichang, 2004)) and construct test cases based on that formula. Since we are only interested in testable properties, the set of requirements is usually restricted to safety properties. For linear safety properties (usually expressed as LTL formulas), the test oracle problem is solved by noting that the test executions violating the property (the so-called *bad prefixes*) are inside the language of a finite automaton, as described in (Baier, Christel and Katoen, Joost-Pieter, 2008). This holds for the important class of regular safety properties. In the more general case, property violations can be characterized by accepting states of Büchi automata (Peleska, 2015) or Rabin automata (Safra, 1988). For test generation, the subsets of model executions are of interest, where the premises associated with the property under consideration are fulfilled, so that the expected SUT reactions can be observed. To our best knowledge, no general theory which guarantees complete fault coverage for the general class of safety properties has been elaborated yet: the publications (Fernandez et al., 2003) and (Li and Qi, Zhichang, 2004), for example, only suggest heuristics for generating suitable test suites, without any guarantees concerning test strength or fault coverage. Noteworthy tool support for property-oriented testing is given, for example, by the QuickCheck tool (Derrick et al., 2010), where properties are extracted as algebraic equations extracted from software API specifications.

PRELIMINARY WORK

Model-based testing and symbolic methods. While our research work always had a special focus on MBT[1], the project described in this application is based on the more recent results obtained in the EU FP7 project COMPASS[2] and the research project ITTCPS[3] funded by the University of Bremen in the context of the German Universities Excellence Initiative[4]. The main MBT-related results obtained there can be summarized as follows.

[1] See, for example, the early publications (Peleska and Siegel, 1996), (Peleska and Siegel, 1997) about testing against CSP process algebra models.
[2] Comprehensive Modelling of Advanced Systems of Systems, see http://www.compass-research.eu
[3] Implementable Testing Theories for Cyber-physical Systems, see http://www.informatik.uni-bremen.de/agbs/projects/ittcps/index.html
[4] See https://en.wikipedia.org/wiki/German_Universities_Excellence_Initiative

(1) We have shown that every complete testing theory established for (deterministic or nondeterministic) FSMs induces a complete equivalence class testing theory for a more general class of systems with Kripke Structure or Extended Finite State Machine (EFSM) semantics (Huang and Peleska, 2017a). This class of SUTs is characterized by potentially infinite input domains and finite domains for internal states and outputs. The class is very important for the domain of safety-critical control systems; examples are thrust reversal systems for aircrafts, airbag controllers in cars, speed monitoring systems for automated train protection, or even railway interlocking systems, where train positions and point states represent a very large input domain which cannot be completely enumerated for test purposes. It can be shown that every model of this class can be abstracted to a symbolic state machine whose input alphabet consists of first-order expressions characterizing the input equivalence classes needed to capture every possible behavior. Applying complete FSM testing theories to the symbolic state machine yields a symbolic test suite which can be translated into a concrete one by calculating input class representatives by means of an SMT solver. The underlying fault domains are specified by two hypotheses: First, the symbolic state machine abstracting the true SUT behavior has at most m states, and second, the granularity of the equivalence class partition is sufficient to capture every behavior of the SUT (this is the variant of the generic uniformity hypothesis applicable to this class of systems). Effective algorithms for calculating the equivalence classes from the reference models have been published in (Huang and Peleska, 2016a) for deterministic systems and in the lecture notes (Huang and Peleska, 2016b) for the nondeterministic case. General approaches to translating testing theories between different formalisms can be based on the theory of institutions or on the Unified Theory of Programming UTP; this has been investigated in (Cavalcanti et al., 2015; Haxthausen and Peleska, 2016; Peleska, 2015).
(2) Regarding the theoretical foundations of complete testing theories, we have shown that the existence of such theories depends on very general and rather weak factorization properties of the observation language. The class of FSMs is just a special case of a formalism where all models fulfil these properties. These theoretical foundations also indicate how smallest complete test suites can be practically calculated, and it is shown that these suites are independent from the concrete model representation. Preliminary results had been published in (Peleska and Huang, 2016), and the full theory has been elaborated in (Huang and Peleska, 2017b).
(3) Regarding the practical application of complete testing theories, we have shown how input equivalence partition testing can be combined with adaptive random testing, such that random selections are performed from the input classes selected according to the complete strategy. This random selection preserves the suites' completeness properties. At the same time, it considerably increases the test strength for situations where the SUT does not conform to the underlying fault hypotheses. These practical considerations and associated experiments with models from different application domains have been published in (Braunstein et al., 2014; Hübner et al., 2015; Peleska et al., 2016) and, most comprehensively, in (Hübner et al., 2017).
(4) Regarding the relationship between complete testing methods and model checking, we have published a first result that shows how the equivalence class partition methodology discussed above can be effectively used to solve infinite-state model checking problems (Krafczyk and Peleska, 2017).

Property-oriented testing. Our own research in the field of property-oriented testing first focused on the generation of test oracles for passive testing against safety requirements specified in LTL, see (Peleska et al., 2014a) and (Peleska, 2015). A new contribution to complete property-oriented test generation has been published in (Huang and Peleska, 2017c): it has been shown that an important class of safety properties can be characterized by output abstractions of deterministic finite state machines. Using both the original DFSM model and its abstraction, it is possible to elaborate a safety-complete test generation algorithm: the suites generated guarantee to uncover every safety violation of the SUT, while non-critical violations of language equivalence may remain undiscovered. This lesser degree of test strength results in a

significantly reduced test suite size (up to 50%), when compared to test suites with full fault coverage of language equivalence violations. Meanwhile, these first results have been extended by using a more effective test generation algorithm which is based on the H-Method instead of the Wp-Method applied before. Therefore, it leads to still smaller safety-complete test suites, see (Huang et al., 2017). Moreover, extensive experiments have been performed, confirming the test suite reductions achievable when reducing the fault coverage requirements to safety-completeness.

Open-source library fsmlib-cpp. In (Peleska et al., 2017), we have published an open-source C++ library containing the crucial algorithms for FSM manipulation and model-based test generation from FSM models, such as W-Method, Wp-Method, H-Method, and their safety-complete variants according to (Huang and Peleska, 2017c) and (Huang et al., 2017). FSM-related algorithms to be explored in this project will be implemented as extensions to this library and made publicly available as open source.

Model-based testing tool RT-Tester. Our research group has an ongoing cooperation with Verified Systems International GmbH, concerning the development of tools for safety-critical systems verification and validation. In this context, an academic version of Verified Systems' model-based testing tool RT-Tester can be used to integrate the new test generation algorithms to be developed in the context of this project, as far as more complex modelling formalisms are involved than FSMs. The tool is already in use by one of our academic research partners and can also be made available to the others listed in Section **Error! Reference source not found.**. The capabilities of RT-Tester have been described in (Peleska et al., 2011b, 2011a; Peleska, 2013a). In 2015, Verified Systems International has been awarded the runner-up trophy of the EU Innovation Radar Prize for integrating the latest model-based testing technology into the RT-Tester tool.[5]

The key principles of the tool design are given as follows. (1) The test model is parsed into an abstract syntax tree (AST). (2) From the AST, the model's transition relation is automatically calculated, assigning a formal semantics to SysML models whose behavior is specified by sequential operations and concurrent state machines, including timing conditions. This semantics is consistent with the semi-formal description of the SysML standard (OMG, 2015). (3) Test cases can be automatically identified by exploiting the SysML satisfy-relationship between model elements and requirements which is visible in the AST. In addition, user-defined test cases may be specified using LTL formulas whose free variables are model variables and timing constraints. (4) Test data is calculated by solving *symbolic test cases* of the form shown in Figure 1. In this formula, the initial condition *I* specifies the model state from where the next test objective should be solved. Symbol ϕ denotes the transition relation connecting consecutive states (s_{i-1}, s_i), and *G* specifies the test objective. If the test objective has been specified in LTL, the bounded model checking semantics for LTL is applied to transform the formula into a first-order expression, as explained in (Biere et al., 2006). By solving an instance of τ using an SMT solver, concrete input values and timing conditions for the test case to be executed are determined. (5) With the concrete test data calculated according to (4), a model interpreter executes the model to determine the post-state to be reached by this test case. This is necessary, since the calculation according to (4) is frequently performed on a sub-model which has been obtained, for example, by cone-of-influence reduction techniques. (6) Starting from such a post-state, the next test objective is solved, so that the resulting test procedure performs stimulations and checks of SUT responses corresponding to a sequence of test cases.

RT-Tester is distinguished from competing tools like TGV (Jard and Jéron, 2005), UPPAAL-TRON (Larsen et al., 2005), and QuickCheck (Derrick et al., 2010) by supporting all test levels (from software unit testing to hardware-in-the-loop system testing). Moreover, it works with SysML models whose formal semantics has been extracted from the semi-formal OMG SysML

[5] see https://www.verified.de/publications/papers-2015/eu-innovation-radar-price-runner-up-trophy-for-verified-systems-international/

standard (OMG, 2015), whereas TRON is based on Timed Automata models, QuickCheck focuses on software APIs, and TGV applies a proprietary modeling language.

Industrial application. During the last 5 years, our research group has supported Verified Systems International in optimizing the MBT approach for industrial-size projects. Meanwhile, the company has performed numerous MBT projects for Airbus (HW/SW integration testing for the Airbus Cabin Communication System and for the Smoke Detection Controller) and Siemens (MBT of railway communication protocols).

$$\tau \equiv I(s_0) \wedge \bigwedge_{i=1}^{k} \Phi(s_{i-1}, s_i) \wedge G(s_0, \ldots, s_k)$$

Figure 1. Symbolic test case.

1.1 Project-related publications

1.1.1 Articles published by outlets with scientific quality assurance

Huang, W., Peleska, J., 2017a. Complete model-based equivalence class testing for nondeterministic systems. Form. Asp. Comput. 29, 335–364. https://doi.org/10.1007/s00165-016-0402-2

Huang, W., Peleska, J., 2017b. Model-based testing strategies and their (in)dependence on syntactic model representations. Int. J. Softw. Tools Technol. Transf. 1–25. https://doi.org/10.1007/s10009-017-0479-9

Huang, W., Peleska, J., 2017c. Safety-Complete Test Suites, in: Testing Software and Systems, Lecture Notes in Computer Science. Presented at the IFIP International Conference on Testing Software and Systems, Springer, Cham, pp. 145–161. https://doi.org/10.1007/978-3-319-67549-7_9

Huang, W., Peleska, J., 2016a. Complete model-based equivalence class testing. Int. J. Softw. Tools Technol. Transf. 18, 265–283. https://doi.org/10.1007/s10009-014-0356-8

Hübner, F., Huang, W., Peleska, J., 2017. Experimental evaluation of a novel equivalence class partition testing strategy. Softw. Syst. Model. 1–21. https://doi.org/10.1007/s10270-017-0595-8

Krafczyk, N., Peleska, J., 2017. Effective Infinite-State Model Checking by Input Equivalence Class Partitioning, in: Testing Software and Systems, Lecture Notes in Computer Science. Presented at the IFIP International Conference on Testing Software and Systems, Springer, Cham, pp. 38–53. https://doi.org/10.1007/978-3-319-67549-7_3

Peleska, J., 2015. Translating Testing Theories for Concurrent Systems, in: Meyer, R., Platzer, A., Wehrheim, H. (Eds.), Correct System Design, Lecture Notes in Computer Science. Springer International Publishing, pp. 133–151. https://doi.org/10.1007/978-3-319-23506-6_10

Peleska, J., 2013. Industrial-Strength Model-Based Testing - State of the Art and Current Challenges. Electron. Proc. Theor. Comput. Sci. 111, 3–28. https://doi.org/10.4204/EPTCS.111.1

Peleska, J., Huang, W., Hübner, F., 2016. A Novel Approach to HW/SW Integration Testing of Route-Based Interlocking System Controllers, in: Lecomte, T., Pinger, R., Romanovsky, A. (Eds.), Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification, Lecture Notes in Computer Science. Springer International Publishing, pp. 32–49. https://doi.org/10.1007/978-3-319-33951-1_3

Peleska, J., Vorobev, E., Lapschies, F., 2011b. Automated Test Case Generation with SMT-Solving and Abstract Interpretation, in: Bobaru, M., Havelund, K., Holzmann, G.J., Joshi, R. (Eds.), NASA Formal Methods, Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 298–312. https://doi.org/10.1007/978-3-642-20398-5_22

1.1.2 Other publications

none

1.1.3 Patents

It is not planned to apply for any patents. Instead, all results will be published.

2 Objectives and work programme

2.1 Anticipated total duration of the project

3 years

2.2 Objectives

The objectives of this project proposal are highlighted in Figure 2. The arrows in the diagram are interpreted as "*arrow source is enabler of arrow target*".

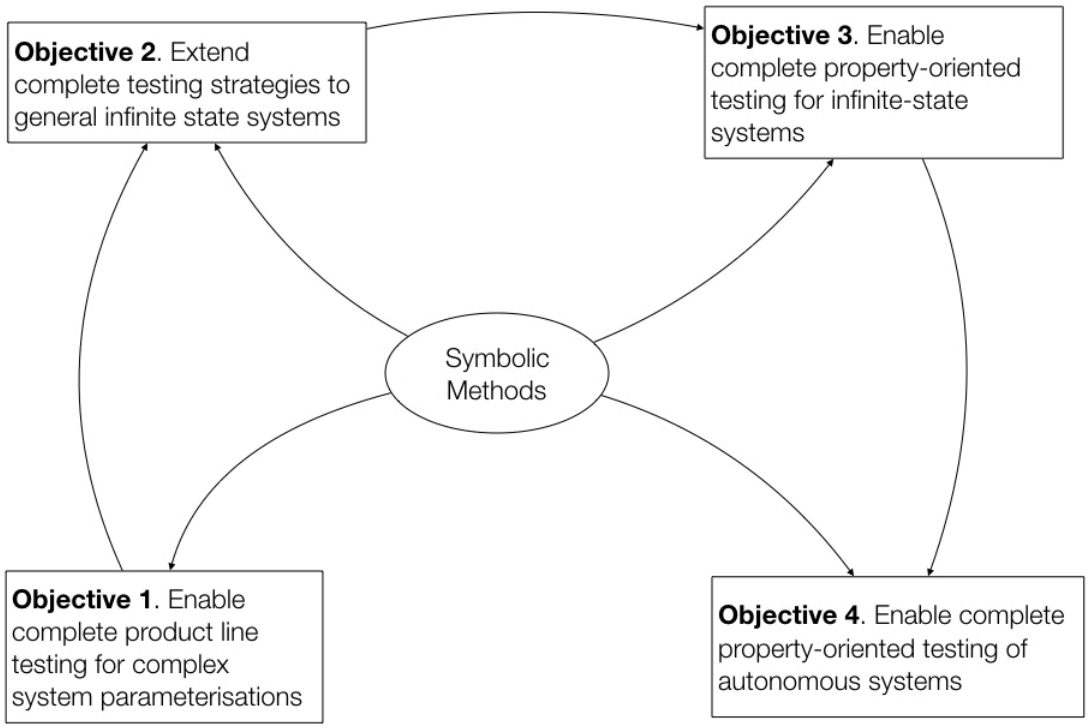

Figure 2. Proposal objectives.

The main objective is labelled as

Objective 3. Enable complete property-oriented testing for infinite-state systems.

We will elaborate new contributions to the field of property-oriented testing of systems with conceptually infinite domains for inputs, outputs, and internal state. The innovations will concentrate on three main aspects (see also the detailed description in the work package specifications below).

1. Facilitate the specification of properties by means of mixed model-based and formula-based techniques.
2. Investigate sufficient conditions for completeness.
3. Investigate necessary conditions for the existence of complete finite test suites in presence of infinite domains.

The term *infinite* is used here always in the sense that the respective domains

- are infinite in the mathematical sense, such as real-valued physical observables, or
- too large to be explicitly enumerated for test purposes, such as variables with very large integral ranges or floating-point types.

The term *symbolic methods* is used here in the wider sense that subsets of the input domain, the internal model state space, or the output domain are represented by formulas specified in some temporal logic or first-order logic. Symbolic methods are essential whenever infinite domains and spaces are involved. This understanding of the term contrasts with the narrower sense sometimes used in the context of model checking, where *symbolic model checking* stands for utilization of ordered binary decision diagrams for state space representations (Clarke et al., 1999).

The main objective is supported by two sub-ordinate objectives.

Objective 2. Extend complete testing strategies to general infinite-state systems.

This objective focuses on MBT. It is motivated by the fact that property-oriented testing is closely related to MBT, so that we follow a mixed property-based and model-based approach to solve the challenges of the main objective. For Objective 2, we will investigate complete test suites checking conformance between reference and implementation models. The existing solution elaborated in our group for systems with infinite input domains and finite domains for internal states and outputs will be extended to more general applications, where the present finiteness restrictions are further relaxed, and time-continuous behavior is considered. This will be based on the results presented by other researchers described in Section 1, but we will extend their work to stronger completeness properties, exploiting the possibilities of our language-based theory of complete testing (Huang and Peleska, 2017b). Moreover, we plan to exploit the possibilities to increase the strength of test suites by adding knowledge about SUT internals, thereby moving from a pure black-box approach to a grey-box testing paradigm, as suggested in (Taromirad and Mousavi, 2017).

Objective 1. Enable complete product line testing for complex system parameterizations.

This objective is crucial to support objectives 2 and 3: today's typical embedded control systems depend on complex configurations, allowing for their application in different operational environments. As a consequence, the reachability of a test goal always depends on the chosen configuration. This induces a special problem when generating sequences of test cases, such that the post-state of one test case execution is to be used as pre-state of the next test case. In the naïve approach, the SMT solver fixes all configuration parameters when calculating the configuration and input data for the first test case, regardless of whether a parameter is really needed for the first goal or not. When calculating test data for the second test case, the configuration data is already fixed, so that no adjustments can be made anymore. This problem leads to the first research goal concerning parameterizations in product line testing.

- Develop an algorithm for symbolic determination of SUT parameterizations, such that the configuration parameters can be incrementally refined while generating sequences of consecutive test cases.

Moreover, the question of test suite completeness also needs to address the configuration space, in addition to the input domains of the SUT and its internal state and output domains. This leads to the second research goal related to product line testing.

- Extend the complete testing theories elaborated in the context of Objective 2 to parameter spaces.

Objective 4. Enable complete property-oriented testing of autonomous systems.

This last objective is concerned with the application and evaluation of the results achieved during the research on objectives 1 – 3. We will investigate property-oriented testing of cyber-physical systems (CPS) with mixed discrete and time-continuous observables and associated control state components. As application domains, case studies from robotics, autonomous vehicles, and railway control systems[6] will be selected and executed in collaboration with the international partners listed in Section 2.7. All three domains present interesting challenges from the hybrid systems domain. The first two domains contain additional challenges regarding the

- dynamicity of configurations, and the
- evolving behavior due to learning effects.

All objectives depend on the utilization of symbolic methods and SMT solvers in a crucial way.

2.3 Work programme incl. proposed research methods

The work packages specified in this proposal are in one-to-one correspondence with the objectives described above; they are shown in Figure 3. It should be emphasized that the work programme specified here is not to be performed by the persons alone, for whom the funding has been applied for in this proposal: the funded persons are integrated into the team specified in Section **Error! Reference source not found.**, who jointly work on this research programme. The funded persons will, however, contribute to each of the work packages defined below.

The effort spent on each work package is shown in Figure 4. The effort of 72 PM for the two funded persons (see Section **Error! Reference source not found.**) is complemented by 78 PM of the project group funded by other resources, as described in Section **Error! Reference source not found.**. The following **milestones** have been planned.

- **M1 (Year 1, End of IV).** The main results for WP1 have been elaborated[7]. First publications about the investigated methods and algorithms are made.
- **M2 (Year 2, End of IV).** The main results for WP2 and WP3 have been produced. First publications about the methods and algorithms investigated in WP2 and WP3 are made.
- **M3 (Year 3, End of III).** The case studies of WP4 have been completed, and the adaptations, extensions, and, based on the feedback from case studies, improvements of the results obtained in WP1, WP2, WP3 have been performed. Further publications on WP1, WP2, WP3, which are now justified by the experimental evaluations are submitted.
- **M4. (Year 3, End of IV).** Final reporting has been completed.

The main focus of the funded PhD student will be on WP1, WP3, and WP4, but an in-depth understanding of the foundations and results associated with WP2 is necessary for the successful investigation of the topics in WP3. The funded post-doc will contribute to work packages WP1, WP2, WP3 uniformly; for WP4 she has supervision and evaluation responsibilities. The work on WP1, WP2, WP3 in Year 3 focuses on technical improvements based on the feedback obtained during the case studies, and on publication of the results obtained for each work package, based on the experimental evaluation in the case studies.

[6] in particular, automated train protection systems
[7] see specification of outputs at the end of each WP description

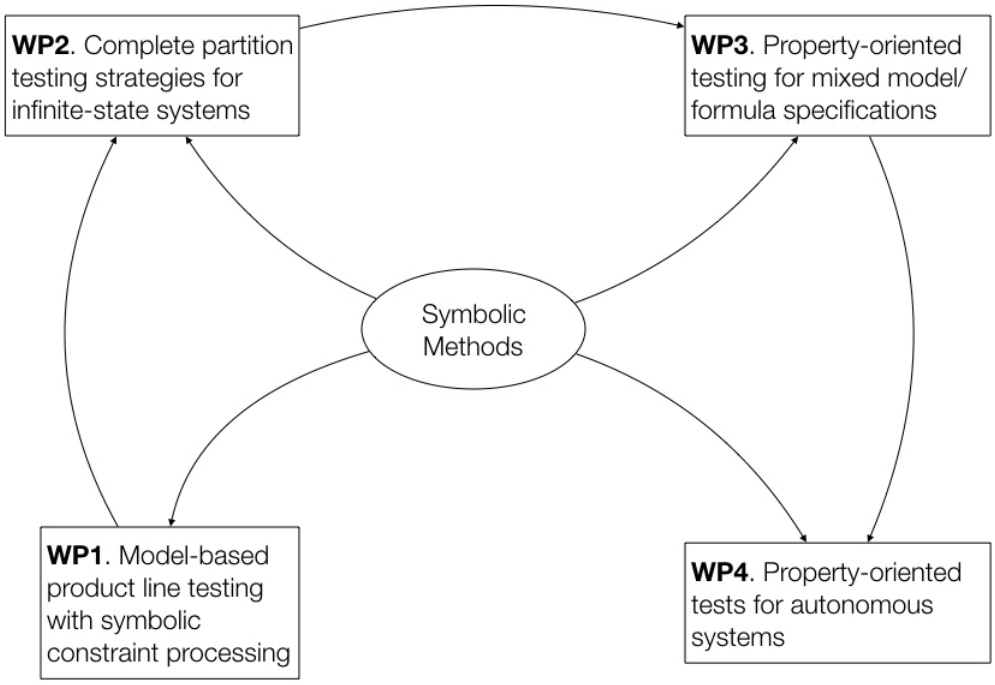

Figure 3. Work packages derived from project objectives.

Figure 4. Work programme – effort table.

Work Packages	Year 1				Year 2				Year 3				PM	
	I	II	III	IV	I	II	III	IV	I	II	III	IV	Δ	Σ
WP1. Model-based product line testing with symbolic constraint processing	1	1	2	2	0,5	0,5						1	8	
	1	1	1	1					0,9	0,9	0,9	0,9	7,6	26
	2	2	2	2					1			1	10	
WP2. Complete partition testing strategies for inifinite-state systems	1	1	0,5	0,5			0,5	0,5					4	
	1	1	1	1	1,4	1,4	1,4	1,4	1	1	1	1	13,6	42
	2	2	2	2	2	2	2	2	2	2	2	2	24	
WP3. Property-oriented testing for mixed model/formula specifications	1	1	0,5	0,5	2	2	2	2	1	1	1	1	15	
	1	1	1	1	1,5	1,5	1,5	1,5	1	1	1	1	14	63
	2,5	2,5	2,5	2,5	4	4	4	4	2	2	2	2	34	
WP4. Property-oriented tests for autonomous systems					0,5	0,5	0,5	0,5	2	2	2	1	9	
					0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,8	20
					0,5	0,5	0,5	0,5	2	2	2	2	10	

	= funded person **PhD student**	Overall PM (funded person **PhD student**) 36
	= Funded person **Post-doc**	Overall PM (funded person **Post-doc**) 36
	= Project team funded from other resources (see Section 5.3)	Overall PM (additional researchers, see Section 5.3) 78
		Overall PM (total) 150

WP1. Model-based product line testing with symbolic constraint processing. The starting point for this work package is the simple product line testing support currently implemented in the RT-Tester tool: configuration parameters are considered as a special class of inputs that can only be set once at system startup and remain invariant while the system is running. In this work package, this simple approach will be extended to comprehensive support for configuration handling in product line testing.

As a first step, we will extend the equivalence class testing theory for nondeterministic Kripke structures or Extended Finite State Machines (EFSMs) from (Huang and Peleska, 2017a) to *parameterized* systems. This requires extending the equivalence class calculation method presented there to parameters, which are special inputs, to be set only once on system startup and to remain invariant during the execution. This results in symbolic state machine abstractions whose input alphabet consists of constraints X_i involving guard conditions on inputs in combination with invariant parameters. Symbolic tests cases can be represented by symbolic input sequences

$$X_1.X_2.X_3 \ldots$$

More complex, so-called *adaptive* test cases, are represented by trees branching according to the SUT reactions, which influences the selection of the next input (Hierons, 2004; Petrenko and Yevtushenko, 2011). Using SMT solvers, concrete test input data can be calculated by solving the constraint formula shown in Figure 5. Each X_i is a formula with free variables representing either inputs *x* or parameters *p*. While the input variables *x* can be freely chosen for the solution of each X_i, all parameters *p* keep their initial value $s_0(p)$.

In the second step, this concept will be extended to *concurrent* SysML state machines, each machine abstracted according to Step 1. We will experiment with constructing symbolic product automata from the individual abstracted FSMs but expect that this will lead to combinatorial explosion problems. Therefore, the main focus of Step 2 will be on search techniques from artificial intelligence which help to find paths through the abstracted concurrent system, leading to the specified test goal, such that concrete test data can again be constructed by means of SMT solvers.

Output WP1. Publication(s) on method and algorithms for symbolic constraint processing and its applications in product line testing. Prototype implementation in RT-Tester.

$$\bigwedge_{i=1}^{k} s_i \models X_i \wedge s_i(\vec{p}) = s_0(\vec{p})$$

Figure 5. Constraint for concrete test data generation presence of configuration parameters.

WP2. Complete partition testing strategies for infinite-state systems. Elaboration of finite testing theories for infinite-state systems typically tries to partition input domains, state spaces, output domains, and configuration spaces in such a way that Gaudel's *uniformity hypothesis* (Gaudel, 1995) is fulfilled in each partition. If the hypothesis is fulfilled, it suffices to test single representatives from each partition in order to conclude that the SUT will behave correctly everywhere. If the partition is finite, it is usually possible to prove a *regularity hypothesis*, showing that each test case representative can be chosen to be finite, because all relevant partition classes have been covered after finitely many test steps. For the class of Kripke structures or EFSMs with infinite input domains and finite domains for internal states and outputs, we have already proven the existence of suitable theories (Huang and Peleska, 2016a,

2017a). Initial investigations from our own group (Peleska et al., 2014b) and more recent research from others (see, e.g. (Petrenko, 2016; Taromirad and Mousavi, 2017)) have shown that the restriction to finite state and output domains can be relaxed when one is either willing to accept lesser test strength or when switching from black-box to grey-box testing, so that some internal information about the SUT can be exploited. We will analyze these results and extend them in such a way that necessary and sufficient conditions for the existence of complete testing theories are elaborated. We will try to prove or disprove our conjecture that complete theories with full fault coverage can be elaborated for hybrid systems whose time-continuous evolutions show piecewise linear behavior. This conjecture is based on decidability investigations about hybrid systems, such as (Henzinger et al., 1998), and on our own result (Huang and Peleska, 2017b). Based on the latter result, we will first prove that a regularity condition[8] of the observation language is not only sufficient (this already has been shown), but also necessary for the existence of finite complete testing theories. With this result at hand, we will investigate for various formalisms, which subsets of models fulfil this regularity condition. For test oracles of hybrid systems, we will extend the existing oracle concept for time-discrete observations (Peleska, 2013b) to time-continuous ones, following the approach of (Mohaqeqi and Mousavi, 2016).

For certain model classes, the algorithms for generating complete test suites will be implemented in RT-Tester. In contrast to other test tools which also apply symbolic techniques (Jard and Jéron, 2005), (Sijtema et al., 2011), our implementation can guarantee complete fault coverage for a well-specified fault domain.

Output WP 2. Publication(s) of methods and algorithms for complete MBT theories applicable to infinite-state systems. Prototype implementation in RT-Tester. Integration of FSM-related algorithms into the fsmlib-cpp (Peleska et al., 2017).

WP 3. Property-oriented testing for mixed model/formula specifications. It is well known that for most formalisms of interest in the CPS domain, property-oriented testing can be theoretically solved by MBT alone: for a given property, it is possible to generate a most non-deterministic model[9] *M* which just fulfils the property (and, of course, its implications) but "nothing more". As a consequence, one could in theory derive a complete test suite from *M* which verifies that the SUT is a refinement of this model. This shows that the SUT fulfils the same property as *M*. In practice, however, this approach is not advisable: the nondeterministic model *M* will usually have far less states than the real SUT. It is well known from FSM testing theory that the number of test cases to be constructed for a complete suite is exponential in the difference between the number of (assumed) SUT states and the number reference model states. As a consequence, constructing tests from *M* alone will frequently lead to test suites of infeasible size.

Our results obtained for testing a restricted class of safety-properties (Huang et al., 2017; Huang and Peleska, 2017c) indicate that another approach is far more promising: tests are generated by means of a method taking into account both the full reference model showing all behaviors and its abstraction to a model which is specialized according to the property to be tested. The abstracted model can be automatically generated from the full model, applying the property to be tested as an "abstraction recipe". As a first step, we will generalize this approach to a wider class of safety-properties specified in LTL. To this end, the automated abstraction technique also needs to be extended to this wider class of properties. This solution, however, still leaves us with the disadvantage that the effort for creating the full model has to be invested.

[8] This regularity condition applies to languages with infinite alphabets; their grammar can be specified using the classical regular expression notation in combination with a symbolic notation specifying subsets of the alphabet. The "classical" regular languages that can be generated by FSMs are just a special case of this more general regularity notion.
[9] e.g., a Büchi automaton, or a nondeterministic process in a process algebra

Therefore, in a second step, it will be investigated how *partial* models can be used instead of complete functional models, since it is desirable to reduce the modeling time for situations where it is already known that only a given set of properties should be tested, and not the complete system behavior. To facilitate this step in practice, we will investigate how learning-based testing methods (see, e.g. (Khosrowjerdi et al., 2017)) can help to construct partial models from SUT observations in an automated way. This approach seems promising, since the partial model is only used to determine the "search depth" for exploring the SUT. The test oracles are directly derived from the property specification.

In a third step, the theory will be extended to CPS with dynamic configurations and evolving behavior, so that Objective 4 described above can be fulfilled. This challenge will be solved by means of a novel testing approach exploiting the fact that in tests with finite duration, only finitely many different configurations can be reached, and only finitely many learning steps can lead to updates of the rules governing the system behavior.[10] As a consequence, it is not necessary to apply higher-order logic for investigating these dynamic changes, as is often required in the formal verification of mobile processes and systems with evolving behavior (Sangiorgi, 2003). Instead, the SUT can only perform finitely many configuration and mode changes that can be selected from pre-defined finite sets. The elements of each set are symbolic constraints capturing equivalence classes of behaviors or configuration states. It remains to show whether these finite sets can be selected in such a way that from finitely many tests, conclusions about the SUT behavior in presence of *arbitrary* possible configurations can be drawn. Again, the investigations in (Henzinger et al., 1998) and further publications of the same author indicate which kinds of systems can be completely verified with finitely many tests.

Outputs WP3. Publication(s) about methods and algorithms for complete property-oriented testing. Prototype implementation in RT-Tester. Integration of FSM-related algorithms into the fsmlib-cpp (Peleska et al., 2017).

WP 4. Property-oriented testing for autonomous systems. In this work package, three case studies will be defined in cooperation with the research partners listed in Section 2.7, covering the application domains robotics, autonomous vehicles, and automated train protection. As SUTs, simulations of robots, vehicles, and trains will be used. For each case study, a list of safety-critical or mission-critical properties will be derived from the system requirements. The fault domains ensuring complete fault coverage will be specified. The test strategies from WP3 will be applied to test the SUT with respect to property violations. For each system, mutations will be generated that are either inside or outside the fault domain where completeness can be guaranteed. For each test suite, the (1) effort for test generation, (2) test suite size, (3) test execution time, and (4) number of killed mutants[11] are measured and used as input to the overall evaluation.

Outputs WP4. Publication(s) about the evaluation of the methods elaborated in WP1 – WP3, when applied to the case studies. Publication of test models and evaluation data on suitable web sites, such as www.mbt-benchmarks.org.

2.4 Data handling

All project results will be presented on international conferences and will be published in journals. We expect 3 dissertations to be completed on topics related to the project. Algorithms related to finite state machines will be made publicly available through the open source library (Peleska et al., 2017). Other algorithms will be implemented in the RT-Tester tool which is also

[10] These facts are ensured by the *finite variability* of real-world real-time systems that cannot change their behavior with infinite or unboundedly accelerating speed.
[11] This number will be 100% for SUTs inside the fault domain but may be less for SUTs outside the domain.

freely available to academic partners for non-commercial applications. Experimental data will be made available on the MBT benchmarks website (“Embedded Systems Testing Benchmarks Site,” 2011).

2.5 Other information

none

2.6 Descriptions of proposed investigations involving experiments on humans, human materials or animals

None

2.7 Information on scientific and financial involvement of international cooperation partners

It is planned to cooperate with the following international research partners for the exchange of scientific results and for the definition and evaluation of case studies. There is no financial involvement of these partners – they have their own funding.

Denmark Technical University, Prof. Anne E. Haxthausen. We have a long-standing cooperation with Prof. Haxthausen, concerning the application of formal methods to the railway domain, see, for example, (Haxthausen et al., 2011; Haxthausen and Peleska, 2000, 2015). In this project, we will cooperate for the definition, execution, and evaluation of the case study in the railway domain.

University of York, Professors Ana Cavalcanti and Jim Woodcock. The cooperation with Professors Cavalcanti and Woodcock started during the COMPASS project (“COMPASS - Comprehensive Modelling for Advanced Systems of Systems,” 2014), where we investigated verification and MBT of Systems of Systems, see (Nielsen et al., 2015), (Cavalcanti et al., 2015). Together with other British scientists, Ana Cavalcanti and Jim Woodcock have received a major grant from the British Engineering and Physical Sciences Research Council for investigating simulation and testing methods for mobile autonomous robots; the project will start in April 2018, see (EPSRC, 2018). From this project, we will receive requirements regarding the testing of autonomous learning systems and input for defining and evaluating the robotics case study we have planned. Conversely, the British research team plans to use RT-Tester in their testing experiments.

University of Leicester, Prof. Mohammad Mousavi. Prof. Mousavi is a specialist on MBT, with special emphasis on product line and cyber-physical systems testing. We will cooperate with him in the context of extending testing theories by exploiting grey-box knowledge about the SUT, and furthermore in the case studies regarding autonomous CPS behavior in the automotive domain.

Computer Research Institute of Montreal CRIM, Prof. Alexandre Petrenko. Prof. Petrenko is a specialist on model-based testing, with special emphasis on FSM and EFSM testing. During the last years, he has become one of the key players researching symbolic methods in MBT, with objectives and results complementing our own strategies. We plan to cooperate with him for the investigation of completeness results that are achievable with symbolic approaches and for researching the necessary restrictions that cannot be relaxed without giving up the finiteness of complete test suites and test cases.

3 Bibliography

Alberto, A., Cavalcanti, A., Gaudel, M.-C., Simão, A., 2017. Formal mutation testing for Circus. Inf. Softw. Technol. 81, 131–153. https://doi.org/10.1016/j.infsof.2016.04.003

Baier, Christel, Katoen, Joost-Pieter, 2008. Principles of Model Checking [WWW Document]. URL http://is.ifmo.ru/books/_principles_of_model_checking.pdf (accessed 12.8.17).

Biere, A., Heljanko, K., Junttila, T., Latvala, T., Schuppan, V., 2006. Linear Encodings of Bounded LTL Model Checking. Log. Methods Comput. Sci. 2. https://doi.org/10.2168/LMCS-2(5:5)2006

Braunstein, C., Haxthausen, A.E., Huang, W., Hübner, F., Peleska, J., Schulze, U., Hong, L.V., 2014. Complete Model-Based Equivalence Class Testing for the ETCS Ceiling Speed Monitor, in: Merz, S., Pang, J. (Eds.), Formal Methods and Software Engineering, Lecture Notes in Computer Science. Springer International Publishing, pp. 380–395. https://doi.org/10.1007/978-3-319-11737-9_25

Brinksma, E., Tretmans, J., 2000. Testing Transition Systems: An Annotated Bibliography, in: Cassez, F., Jard, C., Rozoy, B., Ryan, M. (Eds.), Summer School MOVEP'2k – Modelling and Verification of Parallel Processes. Nantes, pp. 44–50.

Cavalcanti, A., Gaudel, M.-C., 2011. Testing for refinement in Circus. Acta Inform. 48, 97–147. https://doi.org/10.1007/s00236-011-0133-z

Cavalcanti, A., Huang, W., Peleska, J., Woodcock, J., 2015. CSP and Kripke Structures, in: Leucker, M., Rueda, C., Valencia, F.D. (Eds.), Theoretical Aspects of Computing - ICTAC 2015, Lecture Notes in Computer Science. Springer International Publishing, pp. 505–523. https://doi.org/10.1007/978-3-319-25150-9_29

Cavalcanti, A., Simao, A., 2017. Fault-Based Testing for Refinement in CSP, in: Testing Software and Systems, Lecture Notes in Computer Science. Presented at the IFIP International Conference on Testing Software and Systems, Springer, Cham, pp. 21–37. https://doi.org/10.1007/978-3-319-67549-7_2

Chow, T.S., 1978. Testing Software Design Modeled by Finite-State Machines. IEEE Trans. Softw. Eng. SE-4, 178–186.

Clarke, D., Jéron, T., Rusu, V., Zinovieva, E., 2002. STG: A Symbolic Test Generation Tool, in: Tools and Algorithms for the Construction and Analysis of Systems, Lecture Notes in Computer Science. Presented at the International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Springer, Berlin, Heidelberg, pp. 470–475. https://doi.org/10.1007/3-540-46002-0_34

Clarke, E.M., Grumberg, O., Peled, D.A., 1999. Model Checking. The MIT Press, Cambridge, Massachusetts.

COMPASS - Comprehensive Modelling for Advanced Systems of Systems [WWW Document], 2014. URL http://www.compass-research.eu/ (accessed 1.5.18).

Derrick, J., Walkinshaw, N., Arts, T., Earle, C.B., Cesarini, F., Fredlund, L.-A., Gulias, V., Hughes, J., Thompson, S., 2010. Property-Based Testing - The ProTest Project, in: Formal Methods for Components and Objects, Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 250–271. https://doi.org/10.1007/978-3-642-17071-3_13

Dorofeeva, R., El-Fakih, K., Yevtushenko, N., 2005. An Improved Conformance Testing Method, in: Formal Techniques for Networked and Distributed Systems - FORTE 2005, Lecture Notes in Computer Science. Presented at the International Conference on Formal Techniques for Networked and Distributed Systems, Springer, Berlin, Heidelberg, pp. 204–218. https://doi.org/10.1007/11562436_16

Embedded Systems Testing Benchmarks Site [WWW Document], 2011. URL http://www.mbt-benchmarks.org (accessed 1.5.18).

EPSRC, 2018. RoboTest: : Systematic Model-Based Testing and Simulation of Mobile Autonomous Robots [WWW Document]. URL http://gow.epsrc.ac.uk/NGBOViewGrant.aspx?GrantRef=EP/R025479/1 (accessed 1.5.18).

Fernandez, J.-C., Mounier, L., Pachon, C., 2003. Property Oriented Test Case Generation, in: Formal Approaches to Software Testing, Lecture Notes in Computer Science. Presented

at the International Workshop on Formal Approaches to Software Testing, Springer, Berlin, Heidelberg, pp. 147–163. https://doi.org/10.1007/978-3-540-24617-6_11
Frantzen, L., Tretmans, J., Willemse, T.A.C., 2005. Test Generation Based on Symbolic Specifications, in: Grabowski, J., Nielsen, B. (Eds.), Formal Approaches to Software Testing, Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 1–15.
Fujiwara, S., Bochmann, G. v, Khendek, F., Amalou, M., Ghedamsi, A., 1991. Test Selection Based on Finite State Models. IEEE Trans. Softw. Eng. 17, 591–603. https://doi.org/10.1109/32.87284
Gaudel, M.-C., 1995. Testing can be formal, too, in: Mosses, P.D., Nielsen, M., Schwartzbach, M.I. (Eds.), TAPSOFT '95: Theory and Practice of Software Development, Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 82–96.
Group, O.M., 2010. OMG Systems Modeling Language (OMG SysML). Object Management Group.
Haxthausen, A.E., Peleska, J., 2016. On the Feasibility of a Unified Modelling and Programming Paradigm, in: Leveraging Applications of Formal Methods, Verification and Validation: Discussion, Dissemination, Applications, Lecture Notes in Computer Science. Presented at the International Symposium on Leveraging Applications of Formal Methods, Springer, Cham, pp. 32–49. https://doi.org/10.1007/978-3-319-47169-3_4
Haxthausen, A.E., Peleska, J., 2015. Model Checking and Model-Based Testing in the Railway Domain, in: Drechsler, R., Kühne, U. (Eds.), Formal Modeling and Verification of Cyber-Physical Systems. Springer Fachmedien Wiesbaden, pp. 82–121. https://doi.org/10.1007/978-3-658-09994-7_4
Haxthausen, A.E., Peleska, J., 2000. Formal development and verification of a distributed railway control system. IEEE Trans. Softw. Eng. 26, 687–701. https://doi.org/10.1109/32.879808
Haxthausen, A.E., Peleska, J., Kinder, S., 2011. A formal approach for the construction and verification of railway control systems. Form. Asp. Comput. 23, 191–219. https://doi.org/10.1007/s00165-009-0143-6
Hennessy, M., 1988. Algebraic Theory of Processes. MIT Press, Cambridge, MA, USA.
Henzinger, T.A., Kopke, P.W., Puri, A., Varaiya, P., 1998. What's Decidable about Hybrid Automata? J. Comput. Syst. Sci. 57, 94–124. https://doi.org/10.1006/jcss.1998.1581
Hierons, R.M., 2004. Testing from a nondeterministic finite state machine using adaptive state counting. IEEE Trans. Comput. 53, 1330–1342. https://doi.org/10.1109/TC.2004.85
Huang, W., Özoguz, S., Peleska, J., 2017. Safety-complete Test Suites. Softw. Qual. J. Under review.
Huang, W., Peleska, J., 2017a. Complete model-based equivalence class testing for nondeterministic systems. Form. Asp. Comput. 29, 335–364. https://doi.org/10.1007/s00165-016-0402-2
Huang, W., Peleska, J., 2017b. Model-based testing strategies and their (in)dependence on syntactic model representations. Int. J. Softw. Tools Technol. Transf. 1–25. https://doi.org/10.1007/s10009-017-0479-9
Huang, W., Peleska, J., 2017c. Safety-Complete Test Suites, in: Testing Software and Systems, Lecture Notes in Computer Science. Presented at the IFIP International Conference on Testing Software and Systems, Springer, Cham, pp. 145–161. https://doi.org/10.1007/978-3-319-67549-7_9
Huang, W., Peleska, J., 2016a. Complete model-based equivalence class testing. Int. J. Softw. Tools Technol. Transf. 18, 265–283. https://doi.org/10.1007/s10009-014-0356-8
Huang, W., Peleska, J., 2016b. Test Automation - Foundations and Applications of Model-based Testing, Lecture Notes.
Hübner, F., Huang, W., Peleska, J., 2017. Experimental evaluation of a novel equivalence class partition testing strategy. Softw. Syst. Model. 1–21. https://doi.org/10.1007/s10270-017-0595-8
Hübner, F., Huang, W., Peleska, J., 2015. Experimental Evaluation of a Novel Equivalence Class Partition Testing Strategy, in: Blanchette, J.C., Kosmatov, N. (Eds.), Tests and Proofs, Lecture Notes in Computer Science. Springer International Publishing, pp. 155–172. https://doi.org/10.1007/978-3-319-21215-9_10

Jard, C., Jéron, T., 2005. TGV: theory, principles and algorithms. Int. J. Softw. Tools Technol. Transf. 7, 297–315. https://doi.org/10.1007/s10009-004-0153-x

Jeannet, B., Jéron, T., Rusu, V., Zinovieva, E., 2005. Symbolic Test Selection Based on Approximate Analysis, in: Tools and Algorithms for the Construction and Analysis of Systems, Lecture Notes in Computer Science. Presented at the International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Springer, Berlin, Heidelberg, pp. 349–364. https://doi.org/10.1007/978-3-540-31980-1_23

Jéron, T., 2009. Symbolic Model-based Test Selection. Electron. Notes Theor. Comput. Sci., Proceedings of the Eleventh Brazilian Symposium on Formal Methods (SBMF 2008) 240, 167–184. https://doi.org/10.1016/j.entcs.2009.05.051

Khosrowjerdi, H., Meinke, K., Rasmusson, A., 2017. Learning-Based Testing for Safety Critical Automotive Applications, in: Bozzano, M., Papadopoulos, Y. (Eds.), Model-Based Safety and Assessment. Springer International Publishing, pp. 197–211.

Krafczyk, N., Peleska, J., 2017. Effective Infinite-State Model Checking by Input Equivalence Class Partitioning, in: Testing Software and Systems, Lecture Notes in Computer Science. Presented at the IFIP International Conference on Testing Software and Systems, Springer, Cham, pp. 38–53. https://doi.org/10.1007/978-3-319-67549-7_3

Larsen, K.G., Mikucionis, M., Nielsen, B., Skou, A., 2005. Testing Real-time Embedded Software Using UPPAAL-TRON: An Industrial Case Study, in: Proceedings of the 5th ACM International Conference on Embedded Software, EMSOFT '05. ACM, New York, NY, USA, pp. 299–306. https://doi.org/10.1145/1086228.1086283

Li, S., Qi, Zhichang, 2004. Property-Oriented Testing: An Approach to Focusing Testing Efforts on Behaviours of Interest [WWW Document]. URL http://subs.emis.de/LNI/Proceedings/Proceedings58/article3512.html (accessed 12.27.17).

Luo, G., Bochmann, G.V., Petrenko, A., 1994. Test selection based on communicating nondeterministic finite-state machines using a generalized Wp-method. IEEE Trans. Softw. Eng. 20, 149–162. https://doi.org/10.1109/32.265636

Machado, P.D.L., Silva, D.A., Mota, A.C., 2007. Towards Property Oriented Testing. Electron. Notes Theor. Comput. Sci., Proceedings of the Second Brazilian Symposium on Formal Methods (SBMF 2005) 184, 3–19. https://doi.org/10.1016/j.entcs.2007.06.001

Mohaqeqi, M., Mousavi, M.R., 2016. Towards an Approximate Conformance Relation for Hybrid I/O Automata. Electron. Proc. Theor. Comput. Sci. 232, 53–64. https://doi.org/10.4204/EPTCS.232.8

Nielsen, C.B., Larsen, P.G., Fitzgerald, J., Woodcock, J., Peleska, J., 2015. Systems of Systems Engineering: Basic Concepts, Model-Based Techniques, and Research Directions. ACM Comput Surv 48, 18:1–18:41. https://doi.org/10.1145/2794381

OMG, 2015. About the OMG System Modeling Language Specification Version 1.5 [WWW Document]. URL http://www.omg.org/spec/SysML/1.5/ (accessed 1.24.18).

Peleska, J., 2015. Translating Testing Theories for Concurrent Systems, in: Meyer, R., Platzer, A., Wehrheim, H. (Eds.), Correct System Design, Lecture Notes in Computer Science. Springer International Publishing, pp. 133–151. https://doi.org/10.1007/978-3-319-23506-6_10

Peleska, J., 2013a. Industrial-Strength Model-Based Testing - State of the Art and Current Challenges. Electron. Proc. Theor. Comput. Sci. 111, 3–28. https://doi.org/10.4204/EPTCS.111.1

Peleska, J., 2013b. Industrial-Strength Model-Based Testing - State of the Art and Current Challenges, in: Petrenko, A.K., Schlingloff, H. (Eds.), \rm Proceedings Eighth Workshop on Model-Based Testing, \rm Rome, Italy, 17th March 2013, Electronic Proceedings in Theoretical Computer Science. Open Publishing Association, pp. 3–28. https://doi.org/10.4204/EPTCS.111.1

Peleska, J., Dottel, G., Hilken, C., 2017. fsmlib-cpp: A C++ library containing algorithms for processing finite state machines and deriving test cases from FSMs. AGBS University of Bremen.

Peleska, J., Honisch, A., Lapschies, F., Löding, H., Schmid, H., Smuda, P., Vorobev, E., Zahlten, C., 2011a. Embedded Systems Testing Benchmark.

Peleska, J., Huang, W., 2016. Model-Based Testing Strategies and Their (In)dependence on Syntactic Model Representations, in: Critical Systems: Formal Methods and Automated Verification, Lecture Notes in Computer Science. Springer, Cham, pp. 3–21. https://doi.org/10.1007/978-3-319-45943-1_1

Peleska, J., Huang, W., Hübner, F., 2016. A Novel Approach to HW/SW Integration Testing of Route-Based Interlocking System Controllers, in: Lecomte, T., Pinger, R., Romanovsky, A. (Eds.), Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification, Lecture Notes in Computer Science. Springer International Publishing, pp. 32–49. https://doi.org/10.1007/978-3-319-33951-1_3

Peleska, J., Huang, W., Schulze, U., 2014a. D34.3 Contract Support for Evolving Sos.

Peleska, J., Huang, W., Schulze, U., 2014b. D34.2 Specialised Test Strategies.

Peleska, J., Siegel, M., 1997. Test Automation of Safety-Critical Reactive Systems. South Afr. Comput. Jounal 19, 53–77.

Peleska, J., Siegel, M., 1996. From testing theory to test driver implementation, in: Gaudel, M.-C., Woodcock, J. (Eds.), FME'96: Industrial Benefit and Advances in Formal Methods, Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 538–556. https://doi.org/10.1007/3-540-60973-3_106

Peleska, J., Vorobev, E., Lapschies, F., 2011b. Automated Test Case Generation with SMT-Solving and Abstract Interpretation, in: Bobaru, M., Havelund, K., Holzmann, G.J., Joshi, R. (Eds.), NASA Formal Methods, Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 298–312. https://doi.org/10.1007/978-3-642-20398-5_22

Petrenko, A., 2016. Checking Experiments for Symbolic Input/Output Finite State Machines, in: 2016 IEEE Ninth International Conference on Software Testing, Verification and Validation Workshops (ICSTW). Presented at the 2016 IEEE Ninth International Conference on Software Testing, Verification and Validation Workshops (ICSTW), pp. 229–237. https://doi.org/10.1109/ICSTW.2016.9

Petrenko, A., Simao, A., 2015. Checking Experiments for Finite State Machines with Symbolic Inputs, in: Testing Software and Systems, Lecture Notes in Computer Science. Presented at the IFIP International Conference on Testing Software and Systems, Springer, Cham, pp. 3–18. https://doi.org/10.1007/978-3-319-25945-1_1

Petrenko, A., Yevtushenko, N., 2011. Adaptive Testing of Deterministic Implementations Specified by Nondeterministic FSMs, in: Wolff, B., Zaïdi, F. (Eds.), Testing Software and Systems, Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 162–178.

Safra, S., 1988. On the complexity of omega;-automata, in: [Proceedings 1988] 29th Annual Symposium on Foundations of Computer Science. Presented at the [Proceedings 1988] 29th Annual Symposium on Foundations of Computer Science, pp. 319–327. https://doi.org/10.1109/SFCS.1988.21948

Sangiorgi, D., 2003. The Pi-Calculus: A Theory of Mobile Processes. Cambridge University Press.

Schneider, S., 1999. Abstraction and Testing, in: FM'99 — Formal Methods, Lecture Notes in Computer Science. Presented at the International Symposium on Formal Methods, Springer, Berlin, Heidelberg, pp. 738–757. https://doi.org/10.1007/3-540-48119-2_41

Schneider, S., 1995. An Operational Semantics for Timed CSP. Inf Comput 116, 193–213. https://doi.org/10.1006/inco.1995.1014

Sijtema, M., Stoelinga, M.I.A., Belinfante, A., Marinelli, L., 2011. Experiences with Formal Engineering: Model-Based Specification, Implementation and Testing of a Software Bus at Neopost., in: Formal Methods for Industrial Critical Systems, Lecture Notes in Computer Science. Presented at the International Workshop on Formal Methods for Industrial Critical Systems, Springer, Berlin, Heidelberg, pp. 117–133. https://doi.org/10.1007/978-3-642-24431-5_10

Springintveld, J.G., Vaandrager, F.W., D'Argenio, P.R., 2001. Testing timed automata. Theor. Comput. Sci. 254, 225–257.

Taromirad, M., Mousavi, M.R., 2017. Gray-Box Conformance Testing for Symbolic Reactive State Machines, in: Fundamentals of Software Engineering, Lecture Notes in Computer Science. Presented at the International Conference on Fundamentals of Software Engineering, Springer, Cham, pp. 228–243. https://doi.org/10.1007/978-3-319-68972-2_15

Tretmans, J., 1996. Conformance testing with labelled transition systems: Implementation relations and test generation. Comput. Netw. ISDN Syst., Protocol Testing 29, 49–79. https://doi.org/10.1016/S0169-7552(96)00017-7

Vasilevskii, M.P., 1973. Failure diagnosis of automata. Kibern. Transl 4, 98–108.