1 Contact details

Applicants	Main applicant	Co-applicant
naam, voorletters en titel	Vaandrager, F.W., prof.dr.	Tretmans, G.J., dr.ir.
organisatie	Radboud Universiteit Nijmegen	Radboud Universiteit Nijmegen
telefoonnummer	024 - 365 2216	024 - 365 2069
e-mail	`F.Vaandrager@cs.ru.nl`	`tretmans@cs.ru.nl`
aanstellingspercentage	100%	20%
vast dienstverband	ja	ja

Title: Integrating Testing And Learning of Interface Automata (ITALIA)

Keywords: software, state machines, machine learning, model inference, testing, embedded controllers, network protocols

2 Summary

2.1 Research Summary

Humans often manage to learn the behavior of a device or computer program by just pressing buttons and observing the resulting behavior. Especially children are very good in doing this and know exactly how to use a game computer, iPod or microwave oven without ever consulting a manual. In such situations we construct a mental model of a *state diagram*: we determine in which global states a device can be and which state transitions and outputs occur in response to which input. This research proposal deals with the design of algorithms that will allow computers to learn complex state diagrams by providing inputs and observing outputs. The state diagrams that can be learned by current techniques have at most 30.000 states. In contrast, the state diagrams that govern the behavior of computing based systems (defined using dozens of state variables) typically have more than 10^{1000} states. This year we obtained a breakthrough in learning large state diagrams in collaboration with prof. Jonsson from the University of Uppsala: based on some global information about how an application handles data, our algorithm learned models of some realistic communication protocols (TCP, SIP and the new biometric passport). The research objective of the ITALIA project is to further develop this technique and to construct a tool set that will allow us to learn — *routinely and fully automatically* — state diagrams with up to 40 state variables. Our project is unique in bringing together research on automata learning with research on machine learning, model based testing, and computer-aided verification.

2.2 Utilisation Summary

Once they have high-level models of the behavior of software components, software engineers can construct better software in less time: behavioral models can be used to simulate a system and reason about it, they allow all stakeholders to participate in the development process and to communicate with each other, they can be used to generate and test implementations, and they facilitate reuse. A key problem in practice, however, is the construction of models for existing software components, for which no or only limited documentation is available. The solution that the ITALIA project will provide is technology that will allow engineers to infer state diagrams models fully automatically through observations and test, that is, through black box reverse engineering. We expect that our technology will be particularly effective for control oriented applications such as embedded controllers and network protocols. The ITALIA project will focus on the utilisation of model inference technology within the area of testing: once we have learned a model of a software component, we will use model checking technology to analyze this model (e.g. to detect security vulnerabilities) and the technology of model based testing to automatically infer test suites. Using these test suites we can then check, for instance, (a) whether no new faults have been introduced in a modified version of the component (regression testing), (b) whether an alternative implementation by some other vendor agrees with a reference implementation, or (c) whether some new implementation of legacy software is correct. The development of our inference/learning technology will be driven by challenging case studies from a number of areas:

embedded systems (Axini), secure transaction systems (Collis), internet protocols (NLnet Labs), printers (Océ-technologies) and wireless sensor networks (Chess). We will fully integrate our technology with the Axini TestManager, a commercial model based testing tool, and evaluate the effectiveness of our technology by a comparison with the commercial testing platforms of Axini and Collis. Our goal is to reach the point where it becomes interesting for commercial parties, including Axini and Collis, to integrate our technology within their testing tools.

2.3 Research Samenvatting

Mensen slagen er vaak in om het gedrag van een apparaat of computerprogramma te leren puur door op knoppen te drukken en het resulterende gedrag te observeren. Vooral kinderen zijn hier goed in en weten precies hoe ze een game computer, iPod of magnetron moeten bedienen zonder ooit een handleiding te hebben geraadpleegd. In dit soort situaties construeren we mentaal een *toestandsdiagram*: we weten in welke toestanden een apparaat of programma zich kan bevinden en welke toestandsovergangen plaatsvinden als gevolg van welke invoer. Dit onderzoeksvoorstel gaat over de vraag hoe we computers zover kunnen krijgen dat ze zelf, door systematisch knoppen in te drukken en de resulterende uitvoer te observeren, complexe toestandsdiagrammen kunnen leren van apparaten of programma's. Bestaande algoritmen slagen er in om toestandsdiagrammen te leren met maximaal 30.000 toestanden. Deze algoritmen zijn niet direct toepasbaar voor het leren van het gedrag van realistisch ICT toepassingen, aangezien deze toepassingen beschikken over geheugen en er bij invoer- en uitvoeracties ook vaak sprake is van data parameters (telefoonnummers bij een mobieltje, de kooktijd in minuten bij een magnetron, enz.) Zelfs wanneer we uitgaan van een simpel apparaat met een geheugen van slechts 450 bytes, dan heeft het resulterende toestandsdiagram potentieel meer dan $256^{450} \approx 10^{1000}$ toestanden. Toch zijn wij er in recente experimenten voor het eerst in geslaagd om toestandsdiagrammen te leren van enkele veelgebruikte communicatieprotocollen (SIP, TCP en het nieuwe biometrische paspoort). In het bijzonder de toestandsruimte van SIP is astronomisch groot met 17 toestandsvariabelen (waaronder diverse integers van 32 bits, character strings van 200 bytes, enz). Bij onze experimenten moest de gebruiker nog zelf globale informatie aanleveren over hoe het te leren systeem omgaat met datavariabelen. Wetenschappelijk doel van dit project is de nieuwe technieken verder te ontwikkelen en software te ontwikkelen waarmee we — *routinematig en volledig automatisch* — toestandsdiagrammen kunnen leren met tot 40 toestandsvariabelen. Wij denken dat dit mogelijk is door diverse bestaande theorieen te combineren (grammaticale inferentie, testtheorie, machine learning, en computerondersteunde verificatie).

2.4 Utilisatie Samenvatting

Indien ontwikkelaars van software beschikken over modellen van het gedrag van software componenten, dan stelt dit ze veelal in staat om betere software schrijven in minder tijd. Modellen kunnen bijvoorbeeld worden gebruikt om een systeem te simuleren voordat het wordt gebouwd, bij besprekingen tussen belanghebbenden, voor het automatisch genereren van software, voor het automatisch genereren van tests, en bij hergebruik van software. Bij de ontwikkeling van nieuwe systemen worden er daarom tegenwoordig vaak modellen geconstrueerd, bijvoorbeeld in de taal UML. Het construeren van modellen voor bestaande softwarecomponenten, waarvan veelal geen of geen goede documentatie beschikbaar is, vormt in de praktijk echter een enorm probleem. De oplossing die het ITALIA project wil leveren bestaat uit technologie waarmee een belangrijke klasse van modellen, namelijk toestandsdiagrammen, *automatisch* kunnen worden geleerd door "black box" interactie met de software. Wij denken dat onze technologie zeer effectief kan zijn voor softwarecomponenten die zich richten op besturing ('control'), zoals netwerk protocollen en besturingssoftware voor embedded systemen. Het ITALIA project wil zich primair richten op de ontwikkeling en toepassing van leertechnologie voor het testen van software: wanneer een model van een softwarecomponent is geleerd, dan willen we dit model eerst automatisch analyseren (om bijvoorbeeld security problemen op te sporen) en vervolgens automatisch tests genereren uit het model. Deze tests willen we dan gebruiken om vast te stellen of (a) er nieuwe fouten zijn geïntroduceerd bij het aanpassen van de software ("regressietesten"), of (b) een implementatie van

de software door een andere producent hetzelfde gedrag vertoont als een referentie-implementatie, of (c) een nieuwe implementatie van verouderde "legacy" software correct is. Bij de ontwikkeling van onze leertechnologie willen we ons laten leiden door uitdagende industriële case studies uit een aantal gebieden: embedded software (Axini), systemen voor veilige transacties (bijv. electronisch betalen), internet protocollen (NLnet Labs), besturingssoftware voor printers (Océ-Technologies) en draadloze sensornetwerken (Chess). Wij willen onze leertechnologie volledig integreren met de Axini TestManager, een commercieel software pakket voor modelgebaseerd testen. Tevens willen we de effectiviteit van onze technologie evalueren door een vergelijking met commerciële test platforms van Axini en Collis. Ons doel is om deze leertechnologie zover te ontwikkelen dat het interessant wordt voor commerciële partijen (i.h.b. Axini en Collis) om het te integreren in hun testplatforms.

3 The research group

Name	**Specialism**	**hrs/week**
prof.dr. F.W. Vaandrager	(applications of) computer aided verification	6
dr.ir. G.J. Tretmans	model based testing	2
drs. F. Aarts	PhD student, active learning of automata	40
NN	PhD student, validity queries and test coverage	40

Frits Vaandrager will act as project leader+promotor of the PhD students. Jan Tretmans will take care of the daily supervision of the second PhD student. Fides Aarts is an excellent candidate for the first PhD position within the ITALIA project. She obtained her MSc degree in Computer Science at the Radboud University in December 2009 with the distinction "cum laude". Her master thesis research [**Aar09**] on automata learning was carried out at the University of Uppsala under supervision of prof. Bengt Jonsson, a well-known specialist in this area. She coauthors publication [**AV10**], that has been presented at CONCUR'10 (acceptance rate 33%), publication [**AJU10**], that will be presented at ICTSS (acceptance rate 27%), and publication [**ASV10**], that will be presented at ISOLA'10. Aarts has a temporary position until early 2011 within the European Quasimodo project. This proposal intends to secure a PhD position for Aarts after Quasimodo.

4 Scientific description

4.1 Research contents/Introduction

4.1.1 Motivation In our daily life, we frequently infer simple *state diagrams* (also known as *state machines*, *automata*, *transition systems* [Kel76], *I/O automata* [LT87] or *interface automata* [dAH01])[1] with up to a dozen states. We quickly infer dynamic models of gadgets such as mobile phones by just pushing buttons and observing the resulting behavior. A major challenge is to let computers perform similar learning tasks in a rigorous manner for systems with large numbers of states. Tools that are able to infer large state machine models automatically by systematically providing inputs and observing the resulting outputs will have numerous applications in different domains. In particular they will help us to understand and analyze the behavior of software and hardware components.

4.1.2 State-of-the-art The fundamental problem of inducing, learning or inferring automata and (more generally) grammars has been studied for decades. Moore [Moo56] first proposed the problem of learning automata, provided an exponential algorithm and proved that the problem is inherently exponential. Only in recent years *grammatical inference* a.k.a. *grammar induction* has emerged as an independent field with connections to many scientific disciplines, including bio-informatics, computational linguistics and pattern recognition [Hig10]. Grammatical inference techniques aim at building a grammar or automaton for an unknown language, given some data about this language. Also recently, some important developments have taken

[1] At the technical level these terms have slightly different meanings, but in this proposal we use them as synonyms.

place on the borderline of verification, model-based testing and grammatical inference, see e.g. [BGJ+05, Leu06, RSBM09], and researchers have shown that it is possible (at least in principle) to infer models of software components. In *passive* learning the goal is to infer a grammar or automaton from a given set of data, whereas in *active* learning a model of a system is learned by actively performing experiments on that system.

The most efficient techniques use the setup of active learning and assume that a *learner* infers a state diagram through interaction with a *teacher*. The well-known L^* algorithm of Angluin [Ang87], for instance, assumes that the teacher knows a (finite) state machine $\mathcal{M}$. The learner (initially) only knows the set of actions and her task is to learn a machine that is equivalent to $\mathcal{M}$. The teacher will answer two types of questions: *membership queries* ("is string w in the language accepted by $\mathcal{M}$") and *equivalence queries* ("is a hypothesized machine $\mathcal{H}$ correct, i.e., equivalent to the machine $\mathcal{M}$?"). In case of a no-answer, the teacher will also provide a counterexample that proves that the learner's hypothesis is wrong, that is, a distinguishing word from the language. A typical behavior of a learner is to start by asking a sequence of membership queries. The learner maintains an observation table to systematically record the answers of the teacher. If the table satisfies certain "stability" conditions, the learner can extract a hypothesis $\mathcal{H}$, a minimal finite state machine that is consistent with the answers received thus far. If $\mathcal{H}$ is equivalent with $\mathcal{M}$ then the learner has succeeded. Otherwise the returned counterexample is used to update the observation table and the learner poses new membership queries until converging to a new hypothesis, etc. After posing a polynomial number (in the size of the resulting model) of queries, the algorithm terminates with a final hypothesis $\mathcal{H}$ that is equivalent to $\mathcal{M}$.

Figure 1 illustrates how active learning can be used to obtain models of reactive systems. The core of the teacher now is an *SUT (System Under Test)*, a (physical) system to which we can apply inputs and whose outputs we may observe. The learner interacts directly with the SUT to infer a model. Rather than membership queries, the learner poses *output queries*[2]: "which sequences of output events results from a given sequence of input events?" Also, equivalence queries are replaced by the (more general) notion of *validity queries*: "is a given hypothesized model a valid model of the SUT?" Since the SUT cannot respond to validity queries, the teacher is also equipped with a tool for model based testing (MBT). Given a hypothesized model, this tool "approximates" a validity query by generating a long test sequence using some model based testing algorithm. If the SUT passes this test, that is, the output that is generated by the SUT agrees with the output predicted by the model, then we assume that the model is valid. If the output of the SUT is different from the output of the model, this constitutes a counterexample that is forwarded to the learner. Hence, the task of the learner is to collect data by interacting with the SUT and to formulate hypotheses, and the task of the MBT tool is to test the validity of these hypotheses.[3] Note that in this setting the teacher, in general, is not perfect: due to incomplete coverage, it may occur that the SUT passes the test for a hypothesis $\mathcal{H}$, even though it does not conform to $\mathcal{H}$.

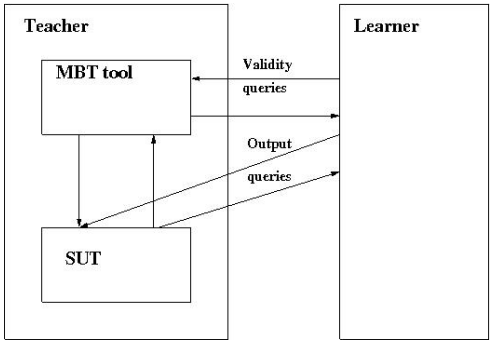

Figure 1: Active learning of reactive systems

Competitions for tools for grammatical inference such as Zulu[4] typically focus on (randomly generated) state machines with around 100 states and 20 symbols. LearnLib [MNRS04, RSB05,

[2]They are called "translation queries" by Vilar [Vil96].

[3]There is an instructive analogy with the elements of a scientific method. According to the standard definitions, see e.g. `http://en.wikipedia.org/wiki/Scientific_method`, a *scientific method* consists of the collection of data through observation and experimentation, and the formulation and testing of hypotheses. Thus one may argue that the Learner and the MBT tool together behave as a "scientist", be it in an extremely small universe.

[4]See `http://labh-curien.univ-st-etienne.fr/zulu/`

RSBM09, RMSM09], the winner of the 2010 Zulu competition [HSM10], is currently able to automatically learn state machines with up to 30.000 states. LearnLib implements both an Angluin style observation table technique for learning state machines, and standard algorithms for model based testing such as state cover, transition cover, W-method, and the UIO method (see [LY96]). The tool has been applied successfully to learn computer telephony integrated systems [HNS03]. The work on LearnLib indicates that learning of practical systems becomes within reach.

Still, practical systems are usually much larger, both in terms of the number of states (just 450 bytes of state variables potentially induces a state space of more than 10^{1000} states) and in the number of actions (due to data parameters in messages). This year, together with prof. Jonsson from Uppsala University, we obtained a breakthrough in learning models of large state machines [**Aar09**, **AJU10**]. The main idea is to place a so-called *mapper* $\mathcal{A}$ in between the SUT $\mathcal{M}$ and the learner, which transforms the interface of the SUT by an abstraction that maps (in a history dependent manner) the large set of actions of the SUT into a small set of abstract actions. By combining the abstract machine $\mathcal{H}$ learned in this way with information about the mapper $\mathcal{A}$, we can effectively learn an over-approximation of the behavior of SUT $\mathcal{M}$. Roughly speaking, the learner is responsible for learning the global "control modes" in which the system can be, and the transitions between those modes, whereas the mapper records some relevant data variables (typically computed from the parameters of previous input and output actions). The approach has been inspired by ideas from predicate abstraction [LGS$^+$95], which has been successful for extending finite-state model checking to larger and even infinite state spaces. Currently, the mapper has to be provided by the user, based on a priori knowledge of the SUT. The feasibility of the approach has been demonstrated by learning models of (fragments of) realistic protocols such as SIP and TCP [**Aar09**, **AJU10**], and of the new biometric passport [**ASV10**]. The learned SIP model is an extended finite state machine with 29 states, 3741 transitions, and 17 state variables with various types (booleans, enumerated types, (long) integers, character strings,..). This corresponds to a state machine with a (larger than) astronomical number of states and transitions, thus far fully out of reach of automata learning techniques.

4.1.3 Scientific objective The scientific objective of the ITALIA project is to further develop the abstraction technique of Aarts, Jonsson and Uijen [**AJU10**] and to construct a tool set that will allow us to learn — *routinely and fully automatically* — state diagrams of simple but realistic ICT applications and software. We will implement the developed algorithms and techniques on top of the LearnLib tool and within state-of-the-art MBT tools, leading to a tool set that can learn state machine models with up to 40 state variables and messages with up to 10 parameters, provided that the operations on the data are simple (boolean operations, basic arithmetic, list processing,..) or known beforehand.[5]

4.1.4 Research approach

Learning models using abstraction The approach of [**AJU10**] is fully automatic, except that the mapper has to be provided by the user based on a priori knowledge of the SUT. In practice, defining the mapper is typically not too hard and only requires a few iterations. The essence of these iterations is to eliminate nondeterminism introduced through abstraction. As explained below, we think we can automate also this part of the learning process. The observation table technique used by LearLib currently only works for fully deterministic systems: if, during the observation phase, a certain sequence of inputs induces two different sequences of outputs then the tool immediately stops learning and reports an error.[6] Typically, nondeterminism will arise when we apply abstraction: it may occur that the behavior of an SUT is fully deterministic but that due to the mapper (which, for instance, abstracts from the precise value of certain

[5] These estimates are based on the current performance of LearnLib and the Daikon tool for detection of likely invariants. Further improvements of these tools will allow us to learn more complex state machine models.

[6] Angluin-style learning of non-deterministic finite-state automata (NFA) is studied in several papers, see for instance [Yok94, BHKL09]. However, the motivation for this work is to obtain concise representations of the learned models: in general a deterministic finite-state automaton (DFA) might be exponentially bigger than (an equivalent) NFA. In the setting of [Yok94, BHKL09] the teacher still behaves deterministically.

input parameters), the system appears to behave nondeterministically from the perspective of the learner. If the abstraction is too coarse and the resulting system behaves nondeterministically from the learner's perspective, then the abstraction has to be refined in order to eliminate the source of this nondeterminism. We intend to formalize this refinement step in terms of a counterexample guided abstraction refinement (CEGAR) procedure, similar to the approach developed by Clarke et al [CGJ+03] in the context of model checking. Very recently, Howar at al [HSM11] developed and implemented such a CEGAR procedure for the special case of a learning setting in which the abstraction is static and does not depend on the execution history. In order to extend to the general (and more complicated), dynamic setting, we intend to use results from the area of machine learning on dynamic detection of likely invariants, in particular the Daikon tool [EPG+07]. Based on the outcomes of previous experiments, the learning algorithm needs to be able to see, for instance, that the third parameter of the second output action equals the first parameter of the first input action, or that the last parameter of an output action is a counter that is incremented modulo 8. Daikon is able to spot such relationships between variables. We also expect to benefit from earlier work of Grinchtein et al [GJP06] on learning timed automata, the work of Lorenzoli et al [LMP08] on passive learning of extended finite state machines, and results on model based testing of systems with data, e.g. the work of Campbell et al [CGN+05] on Spec Explorer, the work of Frantzen et al [**FTW06**] on testing symbolic transition systems, the work of Koopman et al [**KATP03**] on Gast, and the work of Claessen et al [CSH10] on QuickSpec.

Validity queries and test coverage When we compose the abstract state machine and mapper inferred using the techniques outlined in the previous paragraph, we obtain a large, extended finite state machine which, in general, is highly nondeterministic. Our task becomes to test whether this large state machine is a valid model of the SUT at hand. In recent work, we have shown that the so-called **ioco** relation arises naturally as a general notion of validity in a learning setting [**AV10**]. The **ioco** relation has been proposed earlier by co-applicant Tretmans as the basic notion of conformance [**Tre96**, **Tre08**] for transition system based models and forms the semantic basis for many model based testing tools. This link opens up the possibility to use state-of-the-art MBT tools for answering validity queries generated by our learning algorithms.

Once we allow for nondeterminism in models, there is no longer a unique valid model, but a class of them. Within this class one model can be "better" or "closer" to the SUT than another one, e.g., being less abstract and containing more precise details. We intend to formalize the notion of one model being "more precise" than another model, possibly using the notion of alternating simulation of [dAH01]. Approaches include relating different hypothesized models, e.g., by model-based testing of models, and specification-based mutation analysis, i.e., investigating by validation queries, the sensitivity of a model's validity with respect to small model mutations.

For large, nondeterministic systems model-based testing is never exhaustive, which means that any validity query introduces uncertainty. The aim is to measure and quantify this (un)certainty in order to assess the quality of the hypothesized model, as well as to optimize the model-based testing process in order to minimize the probability of accepting a non-valid hypothesized model. This is analogous to the problem of test coverage (specification coverage) and test selection from model-based testing, and our research will be inspired by results in that area, but extensions are needed. First, we consider the area of test coverage and test selection for model-based testing still immature and not practically applicable, in particar not for our domain of large, nondeterministic automata. Second, model learning is different from model-based testing in that now the model is the varying object and not the system under test. This opens new possibilities like specification-based mutation analysis, as described above. Our extensions will be inspired by basic, existing notions for state-model coverage such as state- or transition coverage, and, more importantly, by more sophisticated approaches like metric- (distance-, weight-) based approaches [Bri93, **FGMT02**, ST09, ACV93, KNSP09].

Nondeterminism of the SUT Modern systems are often multithreaded if not distributed. Controlling and observing those systems is a very difficult task and often requires complex testing architectures. Therefore, **ioco** theory and many MBT tools do not assume that it is possible to control the nondeterminism (that is, the scheduling of the SUT). Instead, they only observe the

behavior of the SUT and generate test strategies that try to cover as much as possible of the behavior of the SUT. The challenge therefore is to extend existing learning techniques to a setting of nondeterministic systems.

In practice, we often encounter systems (e.g. the biometric passport and the DNS protocol) whose normal behavior is deterministic, but which may exhibit nondeterministic behavior due to exceptions (e.g. a timeout because of long computations or network overload). As a first step we intend to implement a straightforward —but practical— extension of our approach in which we simply do not explore the behavior of a system after the occurrence of an exception.

As a second, more ambitious step we intend to extend existing learning techiques to nondeterministic systems. Here we expect to benefit from the links between learning and testing. There exists e.g. a close correspondence between the classical theory of conformance testing for deterministic state machines and the automata learning algorithms that have been implemented in LearnLib. In [BGJ+05], it is shown that the observation table technique of Angluin [Ang87] and the variant of it used by LearnLib is strongly related to the W-method for testing of [Cho78], and the observation pack learning technique of [BDG97] is closely related to a conformance testing technique described in [LY96]. Our plan is to adopt existing theory for testing nondeterministic systems, see e.g. [ACY95, NVS+04, VRC06], to the learning setting. Both in testing and in learning we want to drive the SUT to certain states in order to explore the behavior of the SUT from those states. Hence we may for instance reuse the game theory based algorithms of [NVS+04, VRC06] that compute optimal strategies to drive a nondeterministic SUT to certain states while at the same time minimizing the costs of traversal. We also expect to benefit from the work of Willemse [**Wil07**].

Integration of inference techniques The main scientific research topic of ITALIA is regular inference techniques in the spirit of Angluin and LearnLib. Yet, combinations with other learning techniques are important in order to choose the best cocktail of techniques for obtaining a model for a specific system. These learning techniques include passive learning [ARV+10, Ver10, MP05, LMP08, Dal10], inference of automata using homing sequences [RS89] (useful in situations in which the SUT can not be "reset"), black box checking [PVY02], the incremental (sequential) learning approach of [Mei10], other behaviour learning techniques [KPGSG09, BHKL09], and inclusion in the learning process of extra information in the form of mappers (as in [**AJU10**]), component-, partial-, aspect-, or structural models.

The goal of this research line is to combine different, complementary learning techniques in an effective way. This does require an overview of and insight in various existing techniques, and how they combine with our Angluin/LearnLib inspired approach, but it is not the intention to develop many new techniques, nor to develop a framework or taxonomy for all kinds of learning techniques. The approach will be experimental and case-study driven: based on a case study combinations of learning techniques are chosen that give the best solution for that particular case study.

Case studies Within ITALIA we intend to follow a research methodology in which tools are applied to challenging cases, the experience gained during this work is used to generate new theory and algorithms, which in turn are used to further improve the tools. The main reason why practical case studies are so important, also in the early stages, is that even though the theoretical complexity of learning is prohibitively large, learning of practical systems is often feasible since the behavior of these systems exhibits a certain regularity that can be exploited by analysis techniques.

Tooling We have set up a close collaboration with the group of prof. Steffen at the University of Dortmund, which has created and maintains the LearnLib tool. Currently, Learnlib is in charge both of formulating hypotheses and testing them. Given the fact that there are many model based testing tools available, both academic and commercial, and there is an active research community working in this area, we intend to use existing MBT tools for the task of testing hypotheses, thus restricting the use of LearnLib to the construction of hypotheses. Since LearnLib has been designed as an open tool, it should be easy to interface Learnlib with almost any MBT tool. Initially, we intend to experiment with our own MBT tools TorX and Torxakis, and the JTorX tool from Twente, but we will also interface to testing tools used by our industrial partners. In particular we want to use the MBT tool platform of our partner Axini B.V., the Axini TestManager.

4.2 Existing infrastructure

Most of the research will take place at the Institute for Computing and Information Sciences (ICIS) of the Radboud University Nijmegen. ICIS will provide basic computer hardware and software infrastructure for the ITALIA project. The user committee members will provide infrastructure for doing the case studies: specific systems (SUTs) for which we want to learn models, and the interfaces that allow out tools to interact with those systems. Some of the case studies and experiments will be carried out at the premises of the user committee members. For collaboration with the academic partners from Dortmund and Uppsala, longer research visits are planned.

4.3 Time plan and division of tasks

The total project duration is 4 years. The last 6 months of the project will be reserved for completion of the PhD theses of the two students. Following the description in Section 4.1.4, we identify 5 tasks for the initial 42 months. We view the implementation of new algorithms and the development of tool interfaces as an integral part of the other activities, and have decided not to introduce a separate task on tooling.

Task 1: Learning models using abstraction. PhD student 1 will spend 75% of her research time on this task during M1-18, and 25% during M19-42.

Task 2: Validity queries and test coverage. PhD student 2 will spend 75% of her research time on this task during M1-18, and 25% during M19-42.

Task 3: Nondeterminism of the SUT. Both PhD students will spend 50% of their time on this task during M19-30.

Task 4: Case studies. We intend to do one new case study every 6 months until M42 of the project. During this period, approximately 25% of the research time of both PhD students will be devoted to this task. During the initial phase of the project, we may only be able to learn simple and abstract models, and we may have to (partially) define the mappers manually. When the project progresses and our tools and techniques become more powerful, we may decide to revisit some of the early case studies.

Task 5: Integration of inference techniques. Both PhD students will spend 50% of their time on this task during M31-42. The specific topics that will be addressed will be determined by the needs arising from the case studies that are carried out in that period.

5 Utilisation plan

5.1 The problem and the proposed solution

According to a recent survey [VVP10], the software sector is very important for our country. With a turnover of 25 billion euro it contributes 2.8% to the Dutch economy. Excluded from the survey are all the software related activities within the Dutch high tech systems & materials industry, which has an estimated turnover of 74 billion euro [OHM10]. Clearly, software and ICT play a crucial role in many high tech systems. In today's automotive industry, for instance, 90% of the innovations are ICT related and it is estimated that ICT is responsible for 48% of the development costs of cars. New techniques that help us to develop better software in less time may therefore have enormous benefits for our society and economy.

Model-based system development is becoming an increasingly important driving force in the software and hardware industry. In this approach, models become the primary artifacts throughout the engineering lifecycle of computer-based systems. Requirements, behavior, functionality, construction and testing strategies of computer-based systems are all described in terms of (graphical) models. These models are not only used to simulate a system and reason about it, but also to allow all stakeholders to participate in the development process and to communicate with each other, to generate implementations (semi-)automatically, to test implementations, and to facilitate reuse. Once we have high-level models of designs, there are many nice things engineers can do with these models, allowing them to construct better software in less time. For instance, according to Orbons [HM06], the "Happy flow" model developed within the Boderc project (in which we par-

ticipated) enabled Océ to skip at least one complete physical machine-build iteration (saving many man-years of effort), because paper path designs could now be explored virtually. The construction of models typically requires specialized expertise, is time consuming and involves significant manual effort, implying that in practice often models are not available, or become outdated as the system evolves. In practice, 80% of software development involves old (legacy) code, for which only poor documentation is available. Manual construction of models of legacy components is typically very labor intensive and often not cost effective (if possible at all). A key problem therefore is to obtain models for existing software component [**Tre07**].

The solution that the ITALIA project will provide is technology to infer models automatically through observations and test, that is, through black box reverse engineering. We expect that our technology will be particularly effective for control oriented applications such as embedded controllers and network protocols. As outlined in the scientific part of this proposal, we expect to be able to deliver a tool that can routinely learn state machine models with up to 40 state variables and messages up to 10 parameters, provided that the operations on the data are simple (boolean operations, basic arithmetic, list processing,..) or known beforehand. Such a technology will have numerous potential applications. The ITALIA project will focus on applications of model learning technology within the area of testing, since we expect this to be the first area where it will becomes commercially interesting to apply and further develop this technology. More specifically, three types of applications will be pursued within the ITALIA project:

1. Regression testing. Experience shows that as software is fixed or enriched with new functionality, emergence of new and/or reemergence of old faults is quite common. *Regression testing* is any type of software testing that seeks to uncover software errors by partially retesting a modified program. Using our technology, one may first learn a model of the original program, and then use this model as input for state-of-the-art model based testing (MBT) technology to generate and execute tests on the modified software. User committee member Axini will provide case studies in this area, using its extensive network in the embedded systems area, which includes companies such as Philips and TomTom.

2. Test suites based on reference implementation. Our society has become completely dependent on the correct functioning of a variety of communication protocols which describe the operation of the internet, communication in cars and airplanes, handling of financial transactions, communication with smart cards, etc (TCP/IP, WAP, CAN, Bluetooth, OV-chip card, biometric passport, ..). When new protocols are being developed, for instance by the Internet Engineering Task Force (IETF), typically a *reference implementation* is constructed. This is an implementation that is to be be used as a definitive interpretation for the specification of the protocol. During the development of a conformance test suite, at least one relatively trusted implementation of each interface is necessary to discover errors or ambiguities in the specification, and to validate the correct functioning of the test suite. Using our technology, it becomes possible to automatically infer a state machine model of a reference implementation, and to use that model as input for an MBT tool to automatically infer a conformance test suite. Increasingly, hackers attack computer systems by exploiting vulnerabilities in protocol implementations. They send the wrong messages at the wrong time or apply a technique called *fuzzing* [GLM08] in which messages are mutated by flipping bits at random or moving fields of the messages around. The behavior of our learning algorithms is very similar to that of a hacker (although the objectives are opposite!): learning algorithms also provide any possible input in any possible state in order to construct a model. Learning models — in combination with subsequent analysis of these models using simulation or model checking — may help to establish that reference implementations are secure. In summary: application of our technology will lead to (a) the discovery of security vulnerabilities, (b) improved test suites, and (c) saving of time (and money) when reference implementations are extended. User committee members Collis and NLnet Labs will provide case studies in this area (initial case studies will concern secure transaction systems and DNSSEC protocol, respectively).

3. Legacy software is old software that continues to be used because it serves the users' needs, even though better technology is available. Typically the source code of legacy software is hard to read and poorly documented. If one needs to reimplement legacy software on a new platform with modern software technology, a behavioral model of the legacy software will be

useful, either as a basis for MBT of the new implementation, or to study interaction with newer, modeled components, or to generate the new software automatically. User committee members Océ-Technologies and Chess will provide case studies in this area, with focus on embedded control software (initial case studues will concern printers and wireless sensor networks).

Our goal is to reach the point where it becomes interesting for commercial parties to integrate our learning technology within their model based testing tools. The whole set-up of the project, in which the development of inference technology is directly driven by a variety of industrial case studies, has been designed to promote utilisation. As part of our work on the case studies proposed by Axini, we will fully integrate our learning tool with the Axini TestManager: using an adaptor provided by Axini, our learning tool will be able to communicate with the SUT and to construct an hypothesized model; this model will then be translated to a form that can be read by the Axini TestManager, allowing the TestManager to establish the validity of the model; if it turns out that the model is not valid, the counterexample provided by the TestManager will be translated to a form that can be read by our learning tool, thus inducing another cycle in the learning process. Since the Conclusion Test Platform of Collis is not model based, an equally tight integration with the Collis tools is not possible. However, using case studies and adaptors provided by Collis and Axini, we want to compare the effectiveness of the test suites of Collis and Axini with the test suites generated from our tools, in particular whether using our technology it is possible to find bugs that previously remained undetected (and vice versa). The University of Dortmund and the proposers intend to make the LearnLib tool and the extensions to it that will be developed within ITALIA available (under a general public license agreement) to any party that is interested to use it. Obvious candidates are our partners Axini and Collis, but we hope and expect that within four years our technology will reach a level of maturity that also other companies will be interested to integrate it in their testing tools. We intend to publish the basic algorithms for automata inference and test coverage in leading international conferences such as CAV, TACAS, MBT, ICST, ICTSS, ICGI and ICSE. In addition we intend to present our work at the regular meeting places of academia and industry, such as the Nederlandse Testdag, the Embedded Systems Day of Bits&Chips, and the ESI symposium. In this way, we intend to broaden the scope of industrial applications of our technology, and to actively encourage other companies to integrate our technology/algorithms within their tool environments. On the long run, the results of ITALIA may contribute to an update of standards concerned with the development of critical systems (see [BS93] for an overview) in which the use of formal methods is recommended or even required.

5.2 Potential users

The following companies/organizations have agreed to participate in the user committee. As indicated in the attached support letters, all these users will contribute case studies to the project. In addition, Axini and and Collis have committed an in kind contribution (see Section 8.5).

Company	**Contact**	**Telephone**	**e-Mail**
Axini B.V.	Machiel van der Bijl	06 16426332	`vdbijl@axini.com`
Collis B.V.	Henk van Dam	071 5813636	`Dam@collis.nl`
Chess	Marcel Verhoef	023 5149149	`Marcel.Verhoef@CHESS.NL`
NLnet Labs	Matthijs Mekking	020 8884551	`matthijs@NLnetLabs.nl`
Océ-Technologies B.V.	Lou Somers	077 3591917	`lou.somers@oce.com`

5.3 Past performance

The **ioco**-test theory of Tretmans has been implemented in various various model-based testing tools, including the academic tools TorX [**TB03**], JTorX [Bel10], TGV [JJ05], STG [CJRZ02], TestGen [HT99] Uppaal-Tron [HLM$^+$08], and the Agedis Tool set [HN04], and the commercial tool Axini Test Manager. The concepts defined in the PhD. thesis of Tretmans [**Tre92**] made it to the international standard *Formal Methods in Conformance Testing* [Int97].

Using the results of the STW projects *Côte de Resyste* and *Atomyste*, Machiel van der Bijl, AiO in Atomyste, started the spin-off company *Axini* in 2007 [Axi]. *Axini* markets the model-based test

tool *Axini Test Manager* and provides consultancy and support in model-based testing. These two STW projects also provided the know-how for consultancy and model-based testing of Interpay's payment box protocol for the Dutch highway road pricing system (contract between Interpay N.V. and the University of Twente; 2000–2001). More recently, a new model-based testing tool TorXakis developed using the results of the NWO project *Stress*, was applied for model-based testing of the new Dutch electronic passport, commissioned by Ministerie van Binnenlandse Zaken [**MPS+09**]. Currently, the technologies developed in Côte de Resyste, Atomyste and Stress are used in projects with Chess B.V. (model-based testing of a wireless sensor network; in the context of the EU FP7 project Quasimodo), and Océ (model-based testing of printer controller software; in the context of the KWR project FATs Domino). Tretmans is also employed at ESI, where he was and is involved in academic–industrial knowledge transfer in the projects *Tangram* (Model-Based Integration and Testing of Complex High-Tech Systems; with ASML) [**Tre07**] and *Poseidon* (System Evolvability and Reliability of Systems of Systems; with Thales).

Vaandrager has been and is involved in a large number of projects in which formal verification and model checking technology is applied to tackle practical problems from industrial partners, including the STW/PROGRESS project HAAST, the EU projects VHS, AMETIST and QUASIMODO, and the ESI projects BODERC and OCTOPUS. Academic-industrial knowledge transfer was/is very important in all these projects. As a tangible result of these projects, many improvements to control software and communication protocols were suggested, which helped to improve the quality of critical software systems. Examples are collaborations with Philips [**HSV94**, **BPV94**, **SV99**, **DGRV00**, **BSHV03**, **BGVZ10**], ASML [**HvdNV03**, **HvdNV06**], Chess [**SZHV09**, **HSV09**], Océ [**IKY+08**, **AHI+09**], NLnetLabs [**MWVS07**], and Cybernetix Recherche [**GV03**]. The paper [**HvdNV06**] is referred to in patent application ASML ref. P-1784.010. Vaandrager's group has been and is closely involved in the use and development of the timed automata model checker Uppaal, `www.uppaal.com`, see e.g. [**BDL+06**]. In part due to these efforts, the Uppaal toolset is now routinely used for industrial case studies and has thousands of users, both in academia and industry. Within the OCTOPUS project with Océ, we are currently involved in the construction of a toolset for model-driven design-space exploration for embedded systems [**BvG+10**].

6 Intellectual property

The proposed research is not constrained by any contract. Although not intended it could be that some of the new concepts developed will be patented.

7 Positioning of the project proposal

7.1. Uniqueness of the proposed project. In the Netherlands there is an active research community in the area of (model-based) testing. Software test experts from academia and industry meet each other regularly, for instance at the "Nederlandse Testdag", `http://www.testdag.nl` (Tretmans is in the steering committee), and at the Dutch Workshop on Formal Testing Techniques. Thus far, the Dutch testing community has paid little attention to model inference. The group of prof. Pieter Adriaans (UVA) has been working on grammatical inference from a linguistic perspective [AJ06] using the minimum description length (MDL) principle. The size of the automata that we are trying to learn is much larger. The group of prof. Wil van der Aalst (TUE) is working on generating automata from logs "process mining", which basically is a form of passive learning [ARV+10]. Recently, Sicco Verwer, a PhD student of prof. Cees Witteveen (TUD) completed a PhD thesis [Ver10] on passive learning of timed automata. In contrast to the work at TUE and TUD, we focus on active learning. We have established close collaboration with two internationally leading groups in the area of active learning of reactive systems: the group of prof. Bernhard Steffen at the University of Dortmund and the group of prof. Bengt Jonsson at the University of Uppsala. Collaboration with Dortmund centers on the use and further development of LearnLib, whereas collaboration with Uppsala emphasizes the further development of the abstraction techniques of [**AJU10**].

The ITALIA project is unique in bringing together research on automata learning with research on machine learning, model based testing, and computer-aided verification. The approach to use abstraction mappings (developed by the applicants in collaboration with the team of prof. Jonsson [**AJU10**]) is highly original and will enable the scaling of current learning tools that is required to learn models of realistic applications. The project is also unique due to the involvement of a number of industrial partners and research labs that are eager to contribute challenging case studies that will boost the utilisation of the results, and the involvement of Axini B.V., and the proposed integration of inference technology in the Axini Test Manager.

7.2. Embedding of the proposed project. The research will take place within the Model Based System Development (MBSD) group of the Institute for Computing and Information Sciences (ICIS) of the Radboud University Nijmegen. According to the 2010 research assessment of Computer Science research at 9 Dutch universities, ICIS is the best CS department in the Netherlands. Quality, relevance, and vitality & feasibility of the MBSD group were all rated as excellent. The evaluation committee explicitly mentions the combination of machine learning and verification, which we pursue in this proposal, as a promissing direction of research. Apart from the expertise within the MBSD group on model based testing and model checking, we also expect to benefit from the expertise of the Digital Security group of prof. Jacobs and the expertise on machine learning in the groups of dr. Lucas and prof. Heskes.

7.3. Request for support elsewhere. On September 15, 2010, we have submitted a related proposal to the Free Competition of NWO Physical Sciences. The NWO proposal focuses on the theoretical foundations of abstraction techniques for regular inference. The ITALIA proposal is much broader in scope since it also addresses coverage of model based testing techniques and integration with other learning approaches, and puts much more emphasis on case studies and utilisation. When both projects are accepted, they will mutually strengthen each other.

8 Financial planning

8.1. Personnel positions. We request funding for 2 full-time PhD positions for 4 years.

8.2. Consumables. In order to do the case studies, frequent visits to our industrial partners will be required both to learn about these systems ourselves and in order to do experiments on actual systems (SUTs). Based on 30 visits per year per PhD student and 10 visits per year per supervisor a €30, plus €1,400 for attending courses we arive at a total of €11,600.

8.3. Travel abroad. For each PhD student we request funding of €2,000 per year to visits 2 or 3 international conferences and/or summerschools. In addition, we want to schedule a number of visits of about 1 month each to collaborate with the teams of prof. Steffen and prof. Jonsson: 2 visits to Dortmund and two visits to Uppsala for PhD student 1, and 2 visits to Dortmund for PhD student 1. Based on €2,000 for 1 month to Dortmund, and €2,500 for one month to Uppala, this leads to a total travel budget of €29,000.

8.4. Investments. Not applicable.

8.5. Contribution from users. Two of our users have committed an in kind contribution to the project trough involvement of personel: Axini will contribute 100 hours/year = €46,400 (senior researcher) and Collis will contribute 70 hours/year = €32,480 (senior researcher).

8.6. Cost breakdown.

Total project costs	€482.648
Total contribution in cash	€0
Total contribution in kind	€78,880
Requested from STW (STW-bijdrage)	€403.768

8.7. Letters of support. Attached are letters of support from all members of the proposed user committee, from prof. Steffen (Uni. Dortmund) and from prof. Jonsson (Uni. Uppsala).

9 References

9.1 Selection of key publications research group

Some key publication of the group related to the proposal are [**AJU10**, **ASV10**, **AV10**, **LV95**, **LSV07**, **SVD01**, **HSV09**, **Wil07**, **Tre07**, **Tre08**].

9.2 List of publications cited

References

[**Aar09**] F. Aarts. *Inference and Abstraction of Communication Protocols.* Master thesis, Radboud University Nijmegen and Uppsala University, November 2009.

[ACV93] J. Alilovic-Curgus and S.T. Vuong. A metric based theory of test selection and coverage. In A. Danthine, G. Leduc, and P. Wolper, editors, *Protocol Specification, Testing, and Verification XIII*, pages 289–304. North-Holland, 1993.

[ACY95] R. Alur, C. Courcoubetis, and M. Yannakakis. Distinguishing tests for nondeterministic and probabilistic machines. In *Proceedings of the* 27^{th} *Annual ACM Symposium on Theory of Computing*, pages 363–372. ACM, 1995.

[**AHI+09**] I. AlAttili, F. Houben, G. Igna, S. Michels, F. Zhu, and F.W. Vaandrager. Adaptive scheduling of data paths using Uppaal Tiga. In S. Andova et.al, editor, *Proceedings First Workshop on Quantitative Formal Methods: Theory and Applications (QFM'09)*, volume 13 of *Electronic Proceedings in Theoretical Computer Science*, pages 1–12, 2009.

[AJ06] P.W. Adriaans and C. Jacobs. Using mdl for grammar induction. In Y. Sakakibara, S. Kobayashi, K. Sato, T. Nishino, and E. Tomita, editors, *Grammatical Inference: Algorithms and Applications, 8th International Colloquium, ICGI 2006, Tokyo, Japan, September 20-22, 2006, Proceedings*, volume 4201 of *Lecture Notes in Computer Science*, pages 293–306. Springer, 2006.

[**AJU10**] F. Aarts, B. Jonsson, and J. Uijen. Generating models of infinite-state communication protocols using regular inference with abstraction. In A. Petrenko, J.C. Maldonado, and A. Simao, editors, *22nd IFIP International Conference on Testing Software and Systems, Natal, Brazil, November 8-10, Proceedings.* IFIP, 2010.

[Ang87] D. Angluin. Learning regular sets from queries and counterexamples. *Inf. Comput.*, 75(2):87–106, 1987.

[ARV+10] W.M.P. van der Aalst, V. Rubin, H.M.W. Verbeek, B.F. van Dongen, E. Kindler, and C.W. Günther. Process mining: a two-step approach to balance between underfitting and overfitting. *Software and System Modeling*, 9(1):87–111, 2010.

[**ASV10**] F. Aarts, J. Schmaltz, and F.W. Vaandrager. Inference and abstraction of the biometric passport. In *Proceedings 4th International Symposium On Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 2010), 18-20 October 2010 - Amirandes, Heraclion, Crete (Track on Learning Techniques for Software Verification and Validation))*, 2010. To appear. Available via URL `http://www.fidesaarts.de/`.

[**AV10**] F. Aarts and F.W. Vaandrager. Learning i/o automata. In P. Gastin and F. Laroussinie, editors, *21st International Conference on Concurrency Theory (CONCUR), Paris, France, August 31st - September 3rd, 2010, Proceedings*, volume 6269 of *Lecture Notes in Computer Science*, pages 71–85. Springer, 2010.

[Axi] Axini. `http://www.axini.com`.

[BDG97] J.L. Balcázar, J. Díaz, and R. Gavaldà. Algorithms for learning finite automata from queries: A unified view. In *Advances in Algorithms, Languages, and Complexity*, pages 53–72, 1997.

[**BDL+06**] G. Behrmann, A. David, K. G. Larsen, J. Håkansson, P. Pettersson, W. Yi, and M. Hendriks. Uppaal 4.0. In *Third International Conference on the Quantitative Evaluation of SysTems (QEST 2006),* 11-14 September 2006, Riverside, CA, USA, pages 125–126. IEEE Computer Society, 2006.

[Bel10] A. Belinfante. JTorX: A Tool for On-Line Model-Driven Test Derivation and Execution . In J. Esparza and R. Majumdar, editors, *Tools and Algorithms for the Construction and Analysis of Systems – TACAS 2010*, volume 6015 of *Lecture Notes in Computer Science*, pages 266–270. Springer, 2010.

[BGJ+05] T. Berg, O. Grinchtein, B. Jonsson, M. Leucker, H. Raffelt, and B. Steffen. On the correspondence between conformance testing and regular inference. In M. Cerioli, editor, *Fundamental Approaches to Software Engineering, 8th International Conference, FASE 2005, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2005, Edinburgh, UK, April 4-8, 2005, Proceedings*, volume 3442 of *Lecture Notes in Computer Science*, pages 175–189. Springer, 2005.

[**BGVZ10**] J. Berendsen, B. Gebremichael, F.W. Vaandrager, and M. Zhang. Formal specification and analysis of zeroconf using Uppaal. *ACM Transactions on Embedded Computing Systems*, 2010. To appear.

[BHKL09] B. Bollig, P. Habermehl, C. Kern, and M. Leucker. Angluin-style learning of nfa. In Craig Boutilier, editor, *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pages 1004–1009, 2009.

[**BPV94**] D.J.B. Bosscher, I. Polak, and F.W. Vaandrager. Verification of an audio control protocol. In H. Langmaack, W.-P. de Roever, and J. Vytopil, editors, *Proceedings of the Third International School and Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT'94),* Lübeck, Germany, September 1994, volume 863 of *Lecture Notes in Computer Science*, pages 170–192. Springer-Verlag, 1994.

[Bri93] E. Brinksma. On the coverage of partial validations. In M. Nivat, C.M.I. Rattray, T. Rus, and G. Scollo, editors, *AMAST'93*, pages 247–254. BCS-FACS Workshops in Computing Series, Springer-Verlag, 1993.

[BS93] J.P. Bowen and V. Stavridou. Safety-critical systems, formal methods and standards. *IEEE Software Engineering Journal*, 8(4):189–209, 1993.

[**BSHV03**] H. Bohnenkamp, P. van der Stok, H. Hermanss, and F.W. Vaandrager. Cost-optimisation of the IPv4 zeroconf protocol. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN2003)*, pages 531–540, Los Alamitos, California, 2003. IEEE Computer Society.

[**BvG+10**] T. Basten, Benthum E. van, M. Geilen, M. Hendriks, F. Houben, G. Igna, F. Reckers, Smet S. de, L. Somers, E. Teeselink, N. Trcka, F. Vaandrager, J. Verriet, M. Voorhoeve, and Y. Yang. Model-driven design-space exploration for embedded systems: The octopus toolset. In *Proceedings 4th International Symposium On Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 2010), 18-20 October 2010 - Amirandes, Heraclion, Crete (Track on Formal Languages and Methods for Designing and Verifying Complex Engineering Systems)*, 2010. To appear.

[CGJ+03] E.M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-guided abstraction refinement for symbolic model checking. *J. ACM*, 50(5):752–794, 2003.

[CGN+05] C. Campbell, W. Grieskamp, L. Nachmanson, W. Schulte, N. Tillmann, and M. Veanes. Testing concurrent object-oriented systems with spec explorer. In J. Fitzgerald, I.J. Hayes, and A. Tarlecki, editors, *FM 2005: Formal Methods, International Symposium of Formal Methods Europe, Newcastle, UK, July 18-22, 2005, Proceedings*, volume 3582 of *Lecture Notes in Computer Science*, pages 542–547. Springer, 2005.

[Cho78] T.S. Chow. Testing software design modeled by finite-state machines. *IEEE Transactions on Software Engineering*, 4(3):178–187, 1978.

[CJRZ02] D. Clarke, T. Jéron, V. Rusu, and E. Zinovieva. Stg: A symbolic test generation tool. In J.-P. Katoen and P. Stevens, editors, *Tools and Algorithms for the Construction and Analysis of Systems – TACAS 2002*, volume 2280 of *Lecture Notes in Computer Science*, pages 151–173. Springer, 2002.

[CSH10] K. Claessen, N. Smallbone, and J. Hughes. QuickSpec: Guessing formal specifications using testing. In G. Fraser and A. Gargantini, editors, *Tests and Proofs – TAP 2010*, volume 6143 of *Lecture Notes in Computer Science*, pages 6–21. Springer Berlin / Heidelberg, 2010.

[dAH01] L. de Alfaro and T.A. Henzinger. Interface automata. In V. Gruhn, editor, *Proceedings of the Joint 8th European Software Engineering Conference and 9th ACM SIGSOFT Symposium on the Foundation of Software Engineering (ESEC/FSE-01)*, volume 26 of *Software Engineering Notes*, pages 109–120, New York, September 2001. ACM Press.

[Dal10] V. Dallmeier. *Mining and Checking Object Behavior*. PhD thesis, Saarland University, August 2010.

[**DGRV00**] M.C.A. Devillers, W.O.D. Griffioen, J.M.T Romijn, and F.W. Vaandrager. Verification of a leader election protocol: Formal methods applied to IEEE 1394. *Formal Methods in System Design*, 16(3):307–320, June 2000.

[EPG+07] M.D. Ernst, J.H. Perkins, P.J. Guo, S. McCamant, C. Pacheco, M.S. Tschantz, and C. Xiao. The Daikon system for dynamic detection of likely invariants. *Science of Computer Programming*, 69(1-3):35–45, 2007.

[**FGMT02**] L.M.G. Feijs, N. Goga, S. Mauw, and J. Tretmans. Test Selection, Trace Distance and Heuristics. In I. Schieferdecker, H. König, and A. Wolisz, editors, *Testing of Communicating Systems XIV*, pages 267–282. Kluwer Academic Publishers, 2002.

[**FTW06**] L. Frantzen, J. Tretmans, and T.A.C. Willemse. A symbolic framework for model-based testing. In K. Havelund, M. Núñez, G. Rosu, and B. Wolff, editors, *Formal Approaches to Software Testing and Runtime Verification, First Combined International Workshops, FATES 2006 and RV 2006, Seattle, WA, USA, August 15-16, 2006, Revised Selected Papers*, volume 4262 of *Lecture Notes in Computer Science*, pages 40–54. Springer, 2006.

[GJP06] O. Grinchtein, B. Jonsson, and P. Pettersson. Inference of event-recording automata using timed decision trees. In C. Baier and H. Hermanns, editors, *CONCUR 2006 - Concurrency Theory, 17th International Conference, Bonn, Germany, August 27-30, 2006, Proceedings*, volume 4137 of *Lecture Notes in Computer Science*, pages 435–449. Springer, 2006.

[GLM08] P. Godefroid, M.Y. Levin, and D.A. Molnar. Automated whitebox fuzz testing. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2008, San Diego, California, USA, 10th February - 13th February 2008*. The Internet Society, 2008.

[**GV03**] B. Gebremichael and F.W. Vaandrager. Control synthesis for a smart card personalization system using symbolic model checking. In *Proceedings First International Workshop on Formal Modeling and Analysis of Timed Systems (FORMATS 2003),* September 6-7 2003, Marseille, France, volume 2791 of *Lecture Notes in Computer Science*. Springer-Verlag, 2003.

[Hig10] C. de la Higuera. *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press, April 2010.

[HLM+08] A. Hessel, K.G. Larsen, M. Mikucionis, B. Nielsen, P. Pettersson, and A. Skou. Testing Real-Time Systems Using UPPAAL. In R.M. Hierons, J.P. Bowen, and M. Harman, editors, *Formal Methods and Testing*, volume 4949 of *Lecture Notes in Computer Science*, pages 77–117. Springer-Verlag, 2008.

[HM06] M. Heemels and G. Muller, editors. *Model-based design of high-tech systems*, Eindhoven, the Netherlands, March 2006. Embedded Systems Institute.

[HN04] A. Hartman and K. Nagin. The AGEDIS Tools for Model Based Testing. In *Int. Symposium on Software Testing and Analysis – ISSTA 2004*, pages 129–132. ACM Press, 2004.

[HNS03] H. Hungar, O. Niese, and B. Steffen. Domain-specific optimization in automata learning. In W.A. Hunt Jr. and F. Somenzi, editors, *Computer Aided Verification, 15th International Conference, CAV 2003, Boulder, CO, USA, July 8-12, 2003, Proceedings*, volume 2725 of *Lecture Notes in Computer Science*, pages 315–327. Springer, 2003.

[HSM10] F. Howar, B. Steffen, and M. Merten. From ZULU to RERS. In T. Margaria and B. Steffen, editors, *Leveraging Applications of Formal Methods, Verification, and Validation*, volume 6415 of *Lecture Notes in Computer Science*, pages 687–704. Springer, 2010.

[HSM11] F. Howar, B. Steffen, and M. Merten. Automata Learning with Automated Alphabet Abstraction Refinement. In *Verification, Model Checking, and Abstract Interpretation (VMCAI'11), January 23-25, 2011, Austin, Texas, USA*, 2011. To appear.

[**HSV94**] L. Helmink, M.P.A. Sellink, and F.W. Vaandrager. Proof-checking a data link protocol. In H. Barendregt and T. Nipkow, editors, *Proceedings International Workshop TYPES'93,* Nijmegen, The Netherlands, May 1993, volume 806 of *Lecture Notes in Computer Science*, pages 127–165. Springer-Verlag, 1994.

[**HSV09**] F. Heidarian, J. Schmaltz, and F.W. Vaandrager. Analysis of a clock synchronization protocol for wireless sensor networks. In A. Cavalcanti and D. Dams, editors, *Proceedings 16th International Symposium of Formal Methods (FM2009), Eindhoven, the Netherlands, November 2-6, 2009*, volume 5850 of *Lecture Notes in Computer Science*, pages 516–531. Springer, 2009.

[HT99] J. He and K.J. Turner. Protocol-Inspired Hardware Testing. In G. Csopaki, S. Dibuz, and K. Tarnay, editors, *Int. Workshop on Testing of Communicating Systems 12*, pages 131–147. Kluwer Academic Publishers, 1999.

[**HvdNV03**] M. Hendriks, N.J.M. van den Nieuwelaar, and F.W. Vaandrager. Recognizing finite repetitive scheduling patterns in manufacturing systems. In G. Kendall, E. Burke, and S. Petrovic, editors, *Proceedings of the 1st Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2003),* Nottingham, UK*, Volume I*, pages 291–319. The University of Nottingham, August 2003. ISBN 0-9545821-0-1.

[**HvdNV06**] M. Hendriks, N. J. M. van den Nieuwelaar, and F. W. Vaandrager. Model checker aided design of a controller for a wafer scanner. *Software Tools for Technology Transfer*, 8(6):633–647, 2006. Special Section on Quantitative Analysis of Real-time Embedded Systems.

[**IKY+08**] G. Igna, V. Kannan, Y. Yang, T. Basten, M. Geilen, F. Vaandrager, M. Voorhoeve, S. de Smet, and L. Somers. Formal modeling and scheduling of datapaths of digital document printers. In *Proceedings Sixth International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS 2008),* September 15-17, 2008, Saint-Malo, France, volume 5215 of *Lecture Notes in Computer Science*, pages 169–186. Springer Berlin / Heidelberg, 2008.

[Int97] International Telecommunication Union – Telecommunication Standardization Sector of ITU. *ITU-T Recommendation Z.500: Framework on Formal Methods in Conformance Testing.* Series Z: Programming Languages – Methods for Validation and Testing. ITU-T, Geneve, May 1997. Also: ISO/IEC JTC1/SC21 WG7 CD 13245-1.

[JJ05] C. Jard and T. Jéron. TGV: Theory, Principles and Algorithms: A Tool for the Automatic Synthesis of Conformance Test Cases for Non-Deterministic Reactive Systems. *Software Tools for Technology Transfer*, 7(4):297–315, 2005.

[**KATP03**] P. Koopman, A. Alimarine, J. Tretmans, and R. Plasmeijer. Gast: Generic Automated Software Testing. In R. Peña, editor, *IFL 2002 – Implementation of Functional Programming Languages*, volume 2670 of *Lecture Notes in Computer Science*, pages 84–100. Springer-Verlag, 2003.

[Kel76] R.M. Keller. Formal verification of parallel programs. *Communications of the ACM*, 19(7):371–384, 1976.

[KNSP09] G. Kovács, G.A. Németh, M. Subramaniam, and Z. Pap. Optimal string edit distance based test suite reduction for SDL specifications. In R. Reed, A. Bilgic, and R. Gotzhein, editors, *SDL 2009: Design for Motes and Mobiles*, volume 5719 of *Lecture Notes in Computer Science*, pages 82–97. Springer, 2009.

[KPGSG09] T. Kanstrén, E. Piel, A. Gonzalez-Sanchez, and H.-G. Gross. Observation-Based Modeling for Testing and Verifying Highly Dependable Systems – A Practitioner's Approach. In A Wagner, editor, *Workshop on Design of Dependable Critical Systems at Safecomp 2009*, Hamburg, Germany, September 2009. 18 pages.

[Leu06] M. Leucker. Learning meets verification. In F.S. de Boer, M. M. Bonsangue, S. Graf, and W.P. de Roever, editors, *Formal Methods for Components and Objects, 5th International Symposium, FMCO 2006, Amsterdam, The Netherlands, November 7-10, 2006, Revised Lectures*, volume 4709 of *Lecture Notes in Computer Science*, pages 127–151. Springer, 2006.

[LGS+95] C. Loiseaux, S. Graf, J. Sifakis, A. Boujjani, and S. Bensalem. Property preserving abstractions for the verification of concurrent systems. *Formal Methods in System Design*, 6(1):11–44, 1995.

[LMP08] D. Lorenzoli, L. Mariani, and M. Pezzè. Automatic generation of software behavioral models. In *ICSE '08: Proceedings of the 30th international conference on Software engineering*, pages 501–510, New York, NY, USA, 2008. ACM.

[**LSV07**] L.Cheung, M. Stoelinga, and F.W. Vaandrager. A testing scenario for probabilistic processes. *J. ACM*, 54(6), 2007.

[LT87] N.A. Lynch and M.R. Tuttle. Hierarchical correctness proofs for distributed algorithms. In *Proceedings of the* 6^{th} *Annual ACM Symposium on Principles of Distributed Computing*, pages 137–151, August 1987. A full version is available as MIT Technical Report MIT/LCS/TR-387.

[**LV95**] N.A. Lynch and F.W. Vaandrager. Forward and backward simulations, I: Untimed systems. *Information and Computation*, 121(2):214–233, September 1995.

[LY96] D. Lee and M. Yannakakis. Principles and methods of testing finite state machines — a survey. *Proceedings of the IEEE*, 84(8):1090–1123, 1996.

[Mei10] K. Meinke. CGE: A sequential learning algorithm for Mealy automata. In J.M. Sempere and P. García, editors, *Grammatical Inference: Theoretical Results and Applications, 10th International Colloquium, ICGI 2010, Valencia, Spain, September 13-16, 2010. Proceedings*, volume 6339 of *Lecture Notes in Computer Science*, pages 148–162. Springer, 2010.

[MNRS04] T. Margaria, O. Niese, H. Raffelt, and B. Steffen. Efficient test-based model generation for legacy reactive systems. In *HLDVT '04: Proceedings of the High-Level Design Validation and Test Workshop, 2004. Ninth IEEE International*, pages 95–100, Washington, DC, USA, 2004. IEEE Computer Society.

[Moo56] E.F. Moore. Gedanken-experiments on sequential machines. In *Automata Studies*, volume 34 of *Annals of Mathematics Studies*, pages 129–153. Princeton University Press, 1956.

[MP05] L. Mariani and M. Pezzè. Behaviour Capture and Test: Automated Analysis of Component Integration. In 10^{th} *IEEE Int. Conf. on Engineering of Complex Computer Systems – ICECCS'05*, pages 292–301. IEEE Computer Society, 2005.

[**MPS$^+$09**] W. Mostowski, E. Poll, J. Schmaltz, J. Tretmans, and R. Wichers Schreur. Model-Based Testing of Electronic Passports. In M. Alpuente, B. Cook, and C. Joubert, editors, *Formal Methods for Industrial Critical Systems – FMICS 2009*, volume 5825 of *Lecture Notes in Computer Science*, pages 207–209. Springer-Verlag, 2009.

[**MWVS07**] M. Mekking, W. Wijngaards, F.W. Vaandrager, and T. Schouten. Formalizing shim6, a proposed internet standard in uppaal. In *VVSS 2007 (verification and validation of softwaresystems) symposium*, Eindhoven, March 2007. University of Technology.

[NVS$^+$04] L. Nachmanson, M. Veanes, W. Schulte, N. Tillmann, and W. Grieskamp. Optimal strategies for testing nondeterministic systems. In G.S. Avrunin and G. Rothermel, editors, *Proceedings of the ACM/SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2004, Boston, Massachusetts, USA, July 11-14, 2004*, pages 55–64. ACM, 2004.

[OHM10] Point One, HTAS, and M2I. Visiedocument hightech systems & materials, May 2010.

[PVY02] D. Peled, M.Y. Vardi, and M. Yannakakis. Black Box Checking. *Journal of Automata, Languages, and Combinatorics*, 7(2):225–246, 2002.

[RMSM09] H. Raffelt, M. Merten, B. Steffen, and T. Margaria. Dynamic testing via automata learning. *STTT*, 11(4):307–324, 2009.

[RS89] R.L. Rivest and R.E. Schapire. Inference of finite automata using homing sequences (extended abstract). In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing, 15-17 May 1989, Seattle, Washington, USA*, pages 411–420. ACM, 1989.

[RSB05] H. Raffelt, B. Steffen, and T. Berg. Learnlib: a library for automata learning and experimentation. In *FMICS '05: Proceedings of the 10th international workshop on Formal methods for industrial critical systems*, pages 62–71, New York, NY, USA, 2005. ACM Press.

[RSBM09] H. Raffelt, B. Steffen, T. Berg, and T. Margaria. Learnlib: a framework for extrapolating behavioral models. *STTT*, 11(5):393–407, 2009.

[ST09] M.I.A. Stoelinga and M. Timmer. Interpreting a successful testing process: Risk and actual coverage. *Joint IEEE/IFIP Symp. on Theoretical Aspects of Software Engineering – TASE'09*, pages 251–258, 2009.

[**SV99**] M.I.A. Stoelinga and F.W. Vaandrager. Root contention in IEEE 1394. Technical Report CSI-R9905, Computing Science Institute, University of Nijmegen, March 1999.

[**SVD01**] J. Springintveld, F.W. Vaandrager, and P.R. D'Argenio. Testing timed automata. *Theor. Comput. Sci.*, 254(1-2):225–257, 2001.

[**SZHV09**] M. Schuts, F. Zhu, F. Heidarian, and F.W. Vaandrager. Modelling clock synchronization in the Chess gMAC WSN protocol. In S. Andova et.al, editor, *Proceedings Workshop on Quantitative Formal Methods: Theory and Applications (QFM'09)*, volume 13 of *Electronic Proceedings in Theoretical Computer Science*, pages 41–54, 2009.

[**TB03**] J. Tretmans and E. Brinksma. TorX : Automated Model Based Testing. In A. Hartman and K. Dussa-Zieger, editors, *First European Conference on Model-Driven Software Engineering*. Imbuss, Möhrendorf, Germany, December 11-12 2003. 13 pages.

[**Tre92**] J. Tretmans. *A Formal Approach to Conformance Testing*. PhD thesis, University of Twente, Enschede, The Netherlands, 1992.

[**Tre96**] J. Tretmans. Test generation with inputs, outputs, and repetitive quiescence. *Software–Concepts and Tools*, 17:103–120, 1996.

[**Tre07**] J. Tretmans, editor. *Tangram: Model-Based Integration and Testing of Complex High-Tech Systems*, Eindhoven, The Netherlands, 2007. Embedded Systems Institute.

[**Tre08**] J. Tretmans. Model based testing with labelled transition systems. In R.M. Hierons, J.P. Bowen, and M. Harman, editors, *Formal Methods and Testing, An Outcome of the FORTEST Network, Revised Selected Papers*, volume 4949 of *Lecture Notes in Computer Science*, pages 1–38. Springer, 2008.

[Ver10] S. Verwer. *Efficient Identification of Timed Automata — Theory and Practice*. PhD thesis, Delft University of Technology, March 2010.

[Vil96] J.M. Vilar. Query learning of subsequential transducers. In L. Miclet and C. de la Higuera, editors, *Grammatical Inference: Learning Syntax from Sentences, 3rd International Colloquium, ICGI-96, Montpellier, France, September 25-27, 1996, Proceedings*, volume 1147 of *Lecture Notes in Computer Science*, pages 72–83. Springer, 1996.

[VRC06] M. Veanes, P. Roy, and C. Campbell. Online testing with reinforcement learning. In K. Havelund, M. Núñez, G. Rosu, and B. Wolff, editors, *Formal Approaches to Software Testing and Runtime Verification, First Combined International Workshops, FATES 2006 and RV 2006, Seattle, WA, USA, August 15-16, 2006, Revised Selected Papers*, volume 4262 of *Lecture Notes in Computer Science*, pages 240–253. Springer, 2006.

[VVP10] R. te Velde, J. Veltkamp, and M. Plomp. De softwaresector in Nederland — survey 2010. Publicatienummer 2010.002-1018, Dialogic, Utrecht, 2010.

[**Wil07**] T.A.C. Willemse. Heuristics for ioco-based test-based modelling. In L. Brim, B.R. Haverkort, M. Leucker, and J. van de Pol, editors, *Formal Methods: Applications and Technology, 11th International Workshop, FMICS 2006 and 5th International Workshop PDMC 2006, Bonn, Germany, August 26-27, and August 31, 2006, Revised Selected Papers*, volume 4346 of *Lecture Notes in Computer Science*, pages 132–147. Springer, 2007.

[Yok94] T. Yokomori. Learning non-deterministic finite automata from queries and counterexamples. In *Machine Intelligence 13*, pages 169–189, 1994.