Project no.: ICT-FP7-STREP-214755

Project full title: Quantitative System Properties in Model-Driven Design

Project Acronym: QUASIMODO

Deliverable no.: D5.10

Title of Deliverable: Final report: case studies and tool integration

Contractual Date of Delivery to the CEC:	**Month 36**
Actual Date of Delivery to the CEC:	**Month 40 (30. April 2011)**
Organisation name of lead contractor for this deliverable:	**P04 RWTH Aachen**
Author(s):	**Hernán Baró Graf, Henrik Bohnenkamp, Holger Hermanns, Joost-Pieter Katoen, Jan Tretmans, Marcel Verhoef, Jiangsheng Xingj**
Participants(s):	**P01, P02, P04, P05, P07, P08, P10**
Work package contributing to the deliverable:	**WP5**
Nature:	**R**
Version:	**1.0**
Total number of pages:	**55**
Start date of project:	1 Jan. 2008 **Duration:** 36 month

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)

Dissemination Level

PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Abstract:

This deliverable brings together the final results of all case studies of the Quasimodo project, and describes the tools developed and integrated by Quasimodo in the context of their application to the case studies.

Keyword list: Case Studies, Industrial Strength, Modelling, Verification, Testing.

Contents

Abbreviations

AAU: Aalborg University, DK (P01)

ESI: Embedded Systems Institute, NL (P02)

ESI/RU: Radboud University Nijmegen, NL (under the auspices of P02)

ESI/UT: Universiteit Twente, NL (under the auspices of P02)

RWTH: RWTH Aachen University, D (P04)

SU: Saarland University, D (P05)

Terma: TERMA (P07)

CHESS: CHESS (P08)

HYDAC: Hydac (P10)

1 Introduction

Case studies have been the driving momentum behind the Quasimodo project. We started off with four case studies delivered by our industrial partners, but we broadened the spectrum of application cases considerably during the project. The case studies served as a very natural testbed and great motivation for improving our tools, integrating them and applying them. In the sequel, we report on the following distinguished activities:

Chess Myrianed WSN

- model-checking
- model-based testing
- simulation of interference
- probabilistic energy analysis

Chessway: the design of safe real-time systems

Hydac Accumulator Charge Controller

- controller synthesis
- model-based testing

Terma Herschel/Planck

- schedulability analysis
- model-based testing

Model-Based Testing of Electronic Passport

Wireless Bike Brakes

ASML: Modelling in POOSL and UPPAAL

OCE: Data Path Verification

Zeroconf: Specification and Analysis using Uppaal

GSM-R: its impact on railway capacity

Further smaller Model-Based Testing case studies

The individual case study reports vary in their degree of detail. This is because for several of them reports are already included in earlier deliverables of Quasimodo WP5, or because details are available in scientific publications mentioned.

2 The CHESS WSN case: Myrianed MAC Protocols

Quasimodo-Partner CHESS has proposed several variants of MAC protocols for the Myrianed Wireless Sensor Node. Tasks are to find suitable mechanisms for TDMA scheduling and Clock (re)synchronisation. Several quality metrics have been defined to evaluate the suitability of solutions [37]:

- Resource consumption
  - Energy
  - Computational complexity
  - Storage
  - Bandwidth
- Latency: average time required to deliver a message to (all) its destinations
- Scalability: up to tens of thousands of nodes
- Robustness: tolerance to node and communication failure
- Mobility: nodes do not necessarily have a fixed location
- Adaptability: changing communication patterns and network density
- correctness.

In the QUASIMODO project, the Myrianed case study has been tackled in four directions:

1. Formal analysis of the Myrianed protocols, in particular the clock synchronisation mechanisms, using models formulated as Networks of Timed Automata, and analysed using UPPAAL. The results of this study are presented in Section 2.1.
2. Quantitative analysis by extensive discrete-event simulation of quantitative models formulated in the modelling language MoDeST, executed in the performance evaluation environment Möbius. The results of this approach are presented in Section 2.2.
3. Testing of the Myrianed core using three different Online-Testing tools: JTorX, torxakis and UPPAAL Tron. This is described in Section 2.3.
4. A practical case study to show the suitability of the Myrianed nodes for real-time applications. This is described in Section 7.

2.1 The Myrianed Clock Synchronisation mechanism model checked

In Deliverable D5.7 (Section 3.1) we have reported on a very deep analysis of clock synchronisation algorithms with tools from the UPPAAL family. The analysis revealed inconsistencies in the algorithms and devised improvements.

The results are not repeated here, to avoid duplication.

2.2 Energy vs. Message Propagation Speed

Participants: Haidi Yue, Henrik Bohnenkamp, Joost-Pieter Katoen (RWTH), Frits van der Wateren, Marcel Verhoef, Bert Bos (CHESS).

Context In the following, we describe the efforts of RWTH and CHESS in gaining insight into the behaviour of the Myrianed protocols by stochastic discrete-event simulation. Overall aim was to assess the energy efficiency of Myrianed MAC protocols developed by CHESS, and by putting them into context to classical wireless MAC protocols.

The case study has three distinct parts, which are addressed in the following.

Contribution The case study has three distinct parts, which are addressed in the following.

Analysis of gMAC. The very first version of the Myrianed MAC protocol, gMAC, has been described in Deliverable 5.2 [37]. The protocol has been modelled in MoDeST, as described in Deliverable 5.5 [13]. The findings where published in [86], and are summarised in the following.

gMAC is a TDMA protocol, i.e. time is divided into fixed-length frames, which in turn are subdivided into fixed-length slots. The radio is only switched on in the *active slot period* for receiving or sending, which is usually very short compared to the frame length: usually, the ratio is in the order of 10^{-2}. Energy consumption of a WSN node using a TDMA protocol is very well predictable over time, and our measurements have confirmed that energy is drained at a near constant rate as time progresses. In TDMA, in one frame usually only one of the active slots is chosen for sending. All other active slots are used for receiving. In stationary WSN networks an attempt is made to *schedule* the send slots such that neighbouring nodes do not send at the same time and interfere with each other (i.e. to avoid *collisions*). For the Myrianed WSN node, this would be futile, however, since the nodes have been designed especially with mobile applications in mind. Scheduling would be as ineffective as the network topology changes. In a mobile application, collisions are inevitable.

The gMAC protocol does therefore not rely on scheduling, but has mechanisms built in that allow detection of collisions. A node that detects a collision with a neighbouring node will then change its send slot, which otherwise remains unchanged.

Aim of this study is to assess, first, to find out the effectiveness of the protocol in terms of collision detection, and second, the time the protocol needs to propagate a message load through the network. Since energy is consumed linearly with time, the faster the message propagation, the more energy-efficient the protocol.

The propagation speed varies quite a lot with the number of collisions that occur during transmission, and the number of collisions can be influenced by protocol parameters, first and foremost, the number of active slots in a frame. The more active slots, the lower the probability of a collision, but the higher the energy consumption.

We have modelled gMAC in MoDeST and conducted several simulation experiments with Möbius, where we estimated, first, the effectiveness of the collision detection mechanism, and second, a rough measure for the energy consumption vs. the number of active slots.

The results show that collisions can be *avoided* only under great loss of energy, i.e. with a great number of active slots. However, gMAC does recognise quite many collisions, and a change of the send slot, once a collision is detected, does improve message propagation times, i.e. energy efficiency. With the simulation results it was possible to obtain an optimal, i.e. most energy-efficient number of active slots to disseminate all messages, in static network as well as with mobile nodes.

Despite the fact that the results obtained with MoDeST/Möbius seem to indicate that gMAC is indeed a good protocol to improve communication, the results are not prone to criticism. Measurements showed that gMAC is in reality not as good as initially hoped, and the reason for that is probably that the protocol was designed with too simple an idea in mind on how nodes would interact with each other.

In the following section, this problem is further explained and investigated.

Radio Models. gMAC was designed with a specific radio model in mind, the unit disk graph model (UDG). Apparently, UDG is not realistic enough to give insight into the performance and effectiveness of WSN nodes subjected to the harsh conditions that prevail in mobile networks.

The aim of this study are to evaluate the impact of the radio model used for simulations on the resulting data. For this, we carried out a study [46] in which the results of two MAC protocol variants were compared, using two different radio models: the first protocol, gMAC, the second one, slotted ALOHA [70]; the first radio model, UDG, the second one, the signal-to-interference-plus-noise-ratio (SINR) model as first described by Gupta/Kumar [33].

In the classical slotted Aloha protocol (csA), also a TDMA protocol, no mechanism to identify or avoid collisions is present. Instead, a node chooses each frame anew a send slot randomly from set of the active slots. Slotted Aloha is fully probabilistic. Our initial expectation at the beginning of our study was that slotted Aloha would show worst case performance, would therefore provide a lower bound on the achievable performance of a MAC protocol, and give an indication on ho much better gMAC would fare.

The second, SINR radio model [33] works with two types of quantities in a wireless network. First, the signal strength of a node; second, the distance of nodes from each other. Both types of quantities are used to describe the *signal strength* of a node i at the position of a node j. Let p_i be the sending power of node i and x_i its position. Then the relative signal strength of a message from node i at node j ($j \neq i$) is described by $r_i(x_j) := p_i d(x_i, x_j)^{-\alpha}$, where $d(x_i, x_j)$ is the distance between x_i and x_j and α the *path loss exponent*, which determines the power loss over distance. Depending on the environment, it is usually assumed that α has a value between 2 and 5. $r_i(x_j) = 0$ if i is not sending.

The SINR model assumes that a signal can only be received if its strength is significantly higher than the strength of all signals of all sending nodes combined. Formally, this means that Node j will receive a signal from node k if $r_k(x_j) > \beta(\sum_i r_i(x_j) + \nu(x_j))$, where β determines the minimal share of the whole signal that k has to contribute to be received by j. $\nu(x_j)$ is the background noise at node j, and describes abstractly the signals from any other incompatible radio source. The value of β must be between 0.5 and 1.

The experiments carried out in the study are defined by the cross product of the two protocols $\{\text{gMAC}, \text{csA}\}$ and the two radio models $\{\text{UDG}, \text{SINR}\}$. The simulations were carried out with

different network topologies: nodes uniformly distributed, nodes in a Gaussian cluster, and nodes in two Gaussian clusters.

The results can be summarised as follows.

1. gMAC/UDG proves, as in the previous study, effective in reducing the number of collisions. csA/UDG, on the other hand, shows a higher number of collisions, as was expected. gMAC, in general, appears to be superior to csA.

2. The results with the SINR model are in general more pessimistic, i.e., both protocols perform much worse, when compared with the results with the UDG model. Surprisingly, it also turned out that the difference between gMAC/SINR and csA/SINR is nearly gone, gMAC performs only marginally better than csA, and there are cases when csA actually performs better than gMAC in terms of message propagation (this in more noisy or less dense networks).

The results of this study show that the choice of the radio model greatly influences the results.

Analysis of distributed slotted Aloha. The ineffectiveness of gMAC has been established independently by CHESS by measurements. As a consequence, CHESS has abandoned gMAC and developed an alternative approach to the Myrianed MAC protocol: the *distributed slotted Aloha* (dsA) protocol. This protocol is based on csA, with the difference that the listening period of the frame is now scheduled over the active slot period, and that the number of active slots can be varied, depending on the number of nodes perceived as direct neighbours. As in csA, a send slot is chosen randomly each frame from the set of active slots.

In order to assess the behaviour of dsA, we conducted another study using MoDeST and Möbius to model and analyse the protocol. More specifically, the behaviour of dsA was compared to classical slotted Aloha (csA), using the SINR model. The experiments were again carried out to obtain message propagation speed vs. energy consumption. In [85], we show that dsA, while in essence slower than csA, is still more energy efficient. The reason for this is that the number of slots in which the radio is switched on for listening is constant in dsA, independent of how many slots are active.

The dsA protocol assumes that all nodes send with the same power. The SINR model suggests that constant and identical signal strength may result in strong interference in dense areas of the network, which could cause nodes farther away to become disconnected, simply because the noise is too high. To overcome this unfavourable situation, we suggested an extension of dsA, where nodes can adapt their send power dynamically, based on the number of neighbours. The aim of this is not to save power in the nodes by reducing send power (this effect would be negligible since most power is spent on receiving), but by reducing interference and thus improving the message propagation speed. Sending power is thus increased by an integer factor n when size of the neighbourhood list reaches a low-water mark, and is reduced by factor $1/n$ when it reaches a high-water-mark (for $n = 4$, the resulting power levels are similar to the capabilities of the radio used for the CHESS WSN node). We adapted the MoDeST models to incorporate this dynamic power management, and carried out experiments to look for an improvement.

The results of these experiments show the following. In a network topology where nodes are randomly, but uniformly distributed, no effect can be observed. We explain this with the fact that in such a network there are not really areas denser than others.

In a Gaussian cluster, results are different. It turns out that the choice of parameter n is very important: for $n = 4$, no improvement in message propagation speed can be observed, compared to dsA without power management. For $n = 2$, however, an improvement is very clearly visible: energy consumption and message propagation time are about 30% smaller with dynamic power management than for dsA without it.

Perspective Whereas the analysis of the CHESS WSN protocols has provided answers to interesting questions, it has also opened up new ones. While the UDG radio model appears to be unsuitable to give a realistic picture of mobile WSN, the question whether SINR is a suitable alternative has yet to be answered. Comparison with measurement data provided by CHESS proved to be inconclusive. The problem is that the SINR model does incorporate signal strength and distance to decide the reception of messages, assumes perfect sphere-shaped propagation of radio waves, and ignores so fundamental things as antenna characteristics and reflections. Especially the latter might be the reason that measurements of a WSN deployed in railway coaches (i.e. metal boxes with many reflecting surfaces) can not be explained by the SINR model.

However, assuming that SINR is a valid model for interference in outer space in a vacuum, for simulation purposes it seems to be possible to extend it: antenna characteristics can be incorporated by assuming the signal power of a radio being a function of the direction of the reception point. Reflections, while more difficult to handle, could under circumstances be incorporated using ray tracing. If fundamental research would corroborate the validity of these ideas, SINR could become a valuable ingredient in realistic simulation of WSN networks.

2.3 Testing the Myrianed Protocol: JTorX, torxakis, UPPAAL TRON

Participants: Frits van de Wateren, Marcel Verhoef (CHESS),
Jan Tretmans (ESI),
Axel Belinfante (ESI/UT),
Feng Zhu, Julien Schmaltz (ESI/RU).

Context In this section we describe the Quasimodo efforts related to conformance testing of implementations of wireless sensor network nodes through model-based testing.

Conformance Testing. A Wireless Sensor Network (WSN) consist of many, communicating nodes. For successful communication between nodes it is necessary that (*i*) there is a well-designed and precisely defined protocol for communication between the nodes, and (*ii*) the behaviour of each node complies with this protocol definition. For our CHESS Myrianed WSN, the defined protocol is the Myrianed gossip Medium Access Protocol *gMAC*. This protocol was described in Deliverable 5.2. Sections 2.1 and 2.2 of this deliverable describe how this protocol was analysed for various properties.

In this section we consider the second point using the technique of *model-based testing*. This means that model-based testing is applied to perform a classical protocol conformance test [43] of the gMAC protocol layer to check whether the gMAC implementation of an individual node in isolation behaves in compliance with the protocol rules defined in the gMAC protocol specification.

Such a conformance test is important for checking the correctness of a single, implemented node. This importance increases if in later stages of the CHESS WSN project nodes in a single network will be supplied by different, third-party manufacturers based on the same gMAC protocol specification.

The use of model-based testing means that first a model, i.e., an abstract description of the required behaviour of the protocol according to the gMAC specification, is constructed. Secondly, a model-based test tool is used to automatically generate test cases from this model, which are subsequently also automatically executed on the gMAC protocol stack of a WSN node.

Contribution *Approach*. A straightforward approach to model-based testing of an SUT (System Under Test), which in this case is a single WSN node, would consist of (*i*) making a model from available documentation; (*ii*) developing a test environment in which (automatically generated) test cases can be executed, on-line, on a WSN node; (*iii*) feeding the model to a selected model-based testing tool, which automatically generates and executes test cases; and (*iv*) analysing the test outcomes for compliance with the model behaviour.

Such a straightforward approach to model-based testing, however, turned out to be too naive for the WSN node and did not work. First, constructing a model from available documentation turned out to be impossible: the documentation is incomplete, imprecise, ambiguous, and volatile. The design and development of the WSN turned out to be mainly ‘guru-driven’: a few,

very clever engineers designed and developed it, and they know how it works, but the documentation does not necessarily reflect all their latest insights. This implies that making a model is driven by talking with these gurus, trying to construct a model from their explanations, and subsequently trying to get their explanations confirmed by doing model-based testing experiments on the WSN node. A discrepancy between actual outcomes and expected ones does not necessarily point to a fault in the SUT, but may be due to errors, misunderstanding, incompleteness, or invalidity of the model. In case of such discrepancies between the model and the SUT we went back to the gurus trying to get more and better explanations for these discrepancies, improved the model, and re-tested the SUT. Thus, from meetings and explanations, intertwined with test experiments on the SUT, we gradually and iteratively 'learned' the behaviour of the WSN node, making modifications and additions to the model in each iteration. This process is very clarifying for the testers as well as for the guru-developers who learn more about their own system.

The second challenge is developing a test environment in which (automatically generated) test cases can be executed, on-line, on a WSN node. A WSN node is a special device, with special hardware, whose behaviour is very time-critical. This means that interacting with a WSN node requires special hardware that must also operate in very time-critical way, and must be completely time-synchronised with the (on-line) test execution tool. Such hardware is expensive, and developing a real-time testing environment that perfectly synchronises with the SUT clocks is very cumbersome. An alternative is to test the gMAC software in a simulated environment with simulated time, i.e., time is simulated by the increasing value of a variable. Since the gMAC software is implemented as a module in the programming language 'C' it can run on any computer. For our testing of the gMAC layer we ran the software on a normal PC with input and output redirected to a standard socket interface. To accomplish synchronisation between SUT and tester the clock was implemented in software as a special input to the SUT, so that the tester can control the clock of the SUT by explicitly giving clock ticks to the SUT.

Then, in the third step, feeding the model into a model-based testing tool for generation and execution of test cases is indeed completely automatic. Yet, when analysing the test outcomes for compliance with the model behaviour, it has to be taken into account that in this case model-based testing is not only checking an SUT with respect to a model, but also checking the model itself. Model-based testing in practice serves as a technique to detect discrepancies between the SUT and the model, but without making any judgement about which one is wrong. Only subsequent analysis and diagnosis can show whether the model shall be adapted, or the SUT shall be repaired. What we see is a process of concurrent improvement of both the SUT and the model by iteratively comparing them using tests: the SUT is improved using tests generated from the model, and the model is refined using observations made during these tests. The benefit of model-based testing is that this comparison is fully automated.

When in the beginning no precise model could be constructed due to lack of complete and precise documentation, the way to start was to develop a very liberal, non-deterministic model, i.e., a model that allows (almost) any behaviour. By observing the responses of the SUT and analysing the test logs this model was refined, after which the SUT was tested again with this refined model, which, in turn, was refined based on observations, test logs, and discussions with the guru-developers. The result is an 'agile' model-based testing process in which both the model and the SUT are refined and improved in cyclic steps. Figure 1 gives one of the resulting Timed

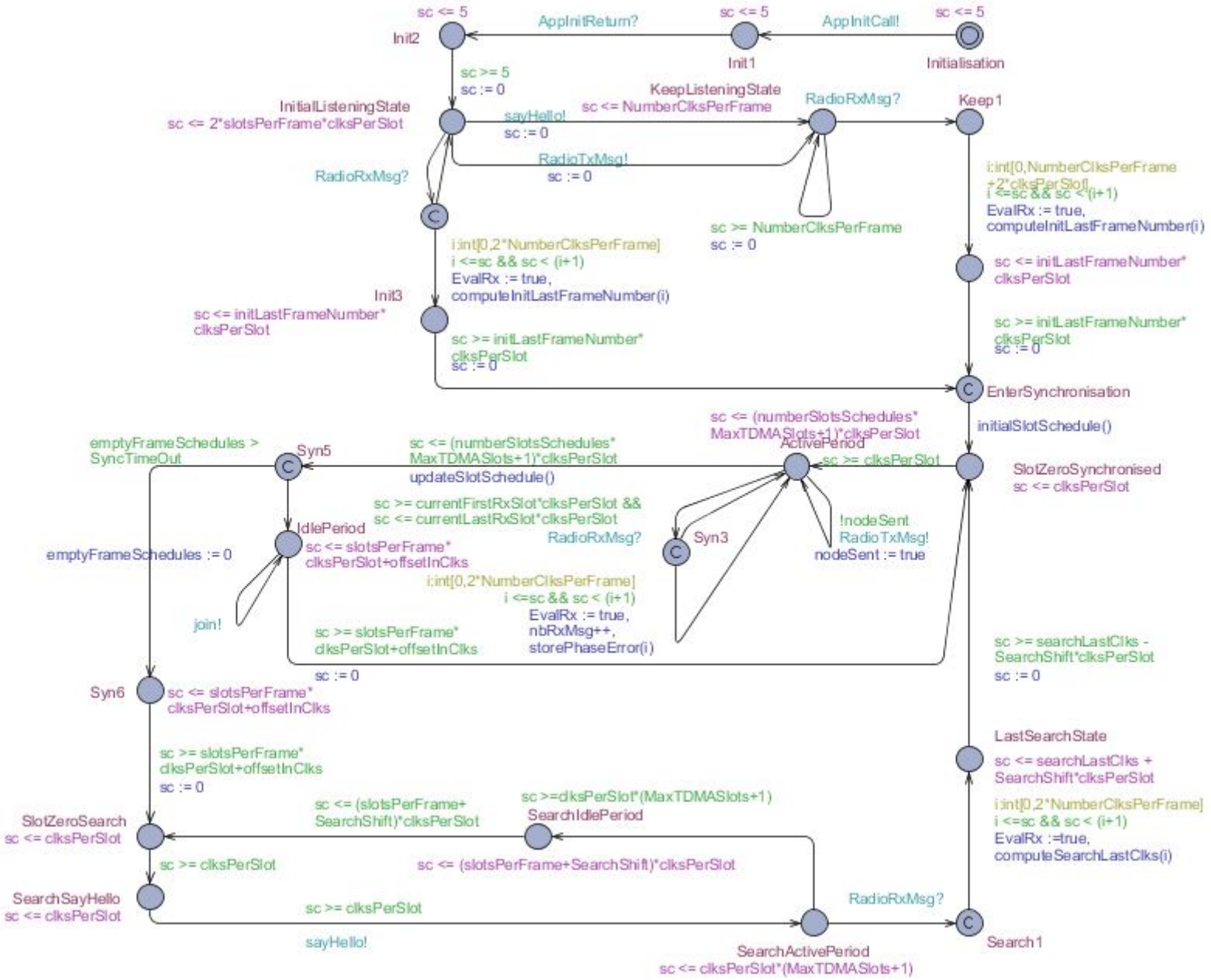


Figure 1: One of the Timed Automata models of the WSN node.

Tools. For model-based testing three model-based testing tools were used: UPPAAL-TRON [38], JTORX [10], and TORXAKIS [5]. These tools were all developed by Quasimodo partners, they were improved during the project, they have their theoretical underpinning in the **ioco**-testing theory [75], or a timed variant of it [84], and they perform model-based testing on-line (on-the-fly), i.e., they execute the test case while generating it. Model-based testing, including **ioco**, timed testing, and on-line testing, is elaborated in [58]; model-based testing of the WSN node with UPPAAL-TRON is described in more detail in [87, 80].

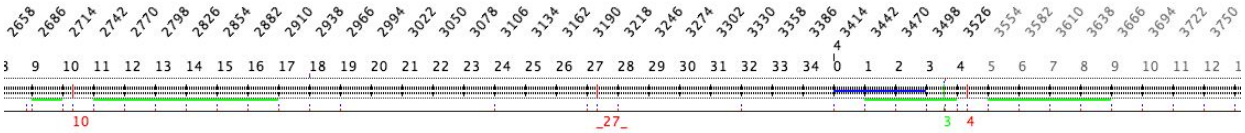


Figure 2: Visualisation of the behaviour of a WSN node from clock tick 2658 to 3750, which correspond to parts of frame 3 and 4. In frame 3 we can see slots 8 until 34, in frame 4 we see slots 0–12. The node sends messages in slots 10 and 27 of frame 3, and in slot 4 of frame 4, and it receives a message in slot 3 of frame 4. In slots 9–16 of frame 3, and in 1–8 of frame 4 the radio receiver is active.

The three tools were used with the same SUT and the same test environment, in simulated time. The tool TORXAKIS cannot deal with real-time. Consequently, all clock-events have to be modelled explicitly as actions in the model, which leads to a cumbersome, though straightforward model. UPPAAL-TRON takes Timed Automata models as input, such as the model in Figure 1, which allow for a more succinct modelling of timing constraints. Also JTORX has a front-end from TORX [12] that can deal with Timed Automata models. A strip-chart-style animation developed using Processing (`www.processing.org`) visualises the timing behaviour of the SUT, and facilitates its analysis; see Figure 2.

Discussion. Several models have been developed, starting with models that are very liberal, i.e., they hardly pose any restrictions on the behaviour of the SUT. Subsequently these models were refined, based on analysis of the actual behaviour of the SUT when tested with these models, and on discussions with the developers. Model-based testing turned out to be feasible, and it increased our understanding of the behaviour of the Myrianed gMAC protocol, in particular (also) in specific situations such as start-up of a node, loss of synchronisation, and resynchronisation. This led to increasingly more precise models. Consequently, model-based testing was not used for assessing the conformance of the SUT, but for improving understanding.

In our first modelling efforts for the WSN node we were inspired by the models which were developed for verification. It turned out, however, that modelling for verification is different from modelling for model-based testing. Whereas for verification a particular behavioural aspect is chosen which is then modelled in great detail, including internal (white-box) mechanisms, modelling for testing must include all external behaviour including aspects which where abstracted away in the verification models, such as start-up and re-synchronisation procedures. On the other hand, for testing abstractions have to made, i.e., non-determinism is introduced, for internal mechanisms that are not controllable from external interfaces. Thus, in the end our models were completely different from the ones used for verification.

Many and long tests were run with all three tools on the SUT, which consisted of the original ‘C’-code compiled and executed on a standard PC in a test environment that redirected inputs, outputs, and clock ticks through a socket connection to the model-based test tool. Due to the use of simulated time the progress of time was much slower than ‘real’ real-time, thus making long-lasting test runs. Whereas synchronisation in simulated time between TORXAKIS and the SUT was easy and straightforward because all clock ticks were represented as explicit actions in

the model, this turned out to be more tricky for UPPAAL-TRON. A special adapter, i.e. glue-code, between UPPAAL-TRON and the SUT was developed with a synchronisation protocol so that the timing delays of UPPAAL-TRON were correctly mapped to the clock ticks of the SUT, and vice versa, i.e., the clock ticks of SUT must be correctly related to the timing constraints of the Timed Automata model. The difficulties encountered when developing this adapter have led to a proposal to change the test interfaces of UPPAAL-TRON, which are currently elaborated by the UPPAAL-TRON development team; see [87] for more details.

All three tools have advantages and disadvantages. TORXAKIS cannot directly deal with real-time – clock ticks must be explicitly added to the model – but it can deal with data in models: manipulations on all kind of data, also for counting clock ticks, are easily performed. JTORX has a nice user interface with logging and replay facilities, it can deal with Timed Automata, but not so easily with data. UPPAAL-TRON is very powerful in real-time, a bit less in data, but in its current version still lacks an insightful user interface. Ideally, we would like to combine time handling and the user modelling interface of UPPAAL-TRON (UPPAAL), the user test interface of JTORX, and data handling of TORXAKIS. Very first steps in this direction were made in Quasimodo; see Deliverable 5.9.

The behaviour of a WSN node is very non-deterministic and time-critical, e.g., the node can determine autonomously in which slot it will send, and reception of a message can lead to completely different subsequent behaviour based only on the time when it was received. All three tools can deal with non-determinism without problems. Real-time testing is an important feature of UPPAAL-TRON, JTORX can deal with real-time models, but for TORXAKIS is has to be explicitly encoded as actions in the model. Non-determinism and real-time are discriminating features of our model-based testing tools. Current commercial model-based testing tools, e..g., [78], are not able to deal with testing of systems like WSN, because of these typical embedded software aspects such as real-time and non-determinism caused by concurrency, abstraction, underspecification, and partial specification.

Perspective Further developments in model-based testing for embedded systems should aim at real, industrial use. For this, a more mature model-based testing tool is required that combines features of UPPAAL-TRON, TORXAKIS, and JTORX, as described above, and that integrates with other tools in model-driven development. Moreover, the 'agile' method of obtaining models via observations made during tests shall be extended, so that models can be 'learnt' from SUTs.

With respect to testing of the WSN node, it would be interesting to move towards 'real' real-time testing, i.e., testing the gMAC layer on its target hardware in real time, or, as a first step, on its target hardware with a speed-controlled clock. Currently, CHESS is working in this direction. Once a model is available this can be used to provide a conformance test service for assessing compliance of third-party Myrianed nodes.

3 The Chessway case: the design of safe real-time embedded systems in UPPAAL

Participants: Bert Bos, Teun van Kuppeveld (CHESS)
Marcel Verhoef (CHESS), Jiansheng Xing (ESI/UT).

Context In [14], we have worked on a case study inspired on an example from industrial practice, the development of a self-balancing scooter. It has been carried out to evaluate the use of timed automata, whereby the design of the discrete control software, including fault handling, was modelled using Uppaal.

System specifications are generally written as text documents, supported by figures to show the desired system behaviour. It appears that many behavioural questions remain unasked and lead to modifications during test and integration phase. What we demonstrate is that modelling such requirements unveils issues that were not foreseen in the specifications, but are important for the user behaviour and to clarify the structure of the solution. Note that many time projects initially focus on the main and desired system behaviour and while adding more and more detail into the system description, incorporate system behaviour related to start-up, shutdown and reaction to unsafe situations is added later and informally. However Start-up, shutdown, and safety reactions have a high impact on the user perception of the system, as we will illustrate in our example. We will also show that preparing a state model of the system behaviour helps to structure and complete the system behavioural descriptions, and could be done as part of the system level design. This work focuses on system level design, in particular on the interaction between user and system. We intend to model the system behaviour for both nominal and for non-nominal situations.

Contribution We took the self-balancing scooter as a system to experiment with, as this was used before in a case study about heterogeneous design. The result of that study was a working self-balancing scooter such that the real-time control for forward and backward movement worked properly. This is the system behaviour as seen from the point of view of control engineering. However it is, without proper switch-on/switch-off functions, left/right steering or any form of safety handling, which shows as system behaviour to the user. We modelled this system behaviour in Uppaal in order to define and to verify it.

In the Chess case study with the self balancing scooter we learnt the following:

1. Simply drawing up an Uppaal model as part of the system design already helps to complete the behavioural description at system level or at least shows hidden complexity.

2. Running the simulations and formally verifying the model shows specification errors very early in the development process and avoids finding these mistakes during test and integration. The model forces the developer to think about the system behaviour and about the software structure. A complex s model is fairly quickly changed to find a simpler one. The modelling exercise and explaining the model to others forced us to clearly define the states for the discrete control and safety monitor.

3. The model helps to specify the system behaviour, in particular switch on / switch off and safety handling and recovery. These parts of system behaviour are often postponed to the design and implementation phase and by that time not well designed due to time constraints. Also these aspects are then designed at engineering level, while they are crucial for the user experience. Structural changes to simplify the software structure are avoided.

4. The setup is also such that it can cope with more than one error the time, or react properly on an additional error while handling the current ones. This is manly due to the independent detection of errors, proceed to run the controller after all error have disappeared, and due to the somewhat rigid, simple strategy of reacting to all errors in the same manner.

5. The system level design of the electronics, such as communication between the FPGA boards was partly determined by the information needed in the Uppaal model, i.e. the Uppaal model leaned us which information was minimally needed to make the distributed control possible.

6. The modeller needs a high level of abstract thinking to imagine the system behaviour as timed automata.

Perspective The case study taught us that the specification could be made more precise due to the development of a Uppaal model of the system behaviour and that the resulting implementation worked *first time right* according to what was specified in the model. The case study showed that a supposedly simple system unveils its complexity already at this level of abstraction and the model was needed to simplify this structure.

4 The Hydac case: Oil under pressure

HYDAC is a company that produces all kinds of components for oil-hydraulic systems. Among them we find sophisticated pump/pressure systems with accumulators and controllers. These are installed at the customer side to provide pressurised oil to applications such as molding machines. The standard controller used in this setting is a very simple two-point on/off controller, but better options are being developed by HYDAC to improve energy efficiency and to reduce wear. Their major development, ACC (accumulator charge controller) is in the focus of this case study. Originally described in Deliverable 5.2, has been subject to several analysis, verification, and testing efforts in the Quasimodo project.

4.1 Robust controller synthesis

In Deliverable D5.7 (Section 2) we have reported on a very successful systematic way to develop models and to use a chain of Quasimodo and other automatic tools for the synthesis, verification and simulation of a provably correct and near optimal controller for this real industrial equipment.

The results are not repeated here, to avoid duplication.

4.2 Model-based Testing on top of a Simulink model

This work is reported in [31].

Participants: Alexander Graf-Brill, Holger Hermanns (SU)
Kai Mittermüller (Hydac),
Jan Tretmans (ESI).

Context Quasimodo supports HYDAC in their software development by devising new techniques and tools for model-driven design, analysis, and especially testing. The aforementioned accumulator-charge controller serves as a case study to perform model based testing.

Contribution As an outcome of earlier Quasimodo work on this case (Deliverable 5.2), there is an implementation of the ACC and a suitable environment in MATLAB/Simulink [1] at hand. Several components are implemented in Stateflow, a control logic tool used to model state charts and flow diagrams within a Simulink model. This model will serve as the implementation under test (IUT), and this implementation is tested against a specification, which is based on the formal analysis work done for this case (see preceding section).

In the MATLAB Simulink IUT model, there are corresponding components for all parts of an oil-hydraulic system (apart from the reservoir, which can be ignored) including an ACC implementation in Stateflow.

The specifics of the case make it possible to split the specification into two orthogonal parts. One of them is a small IO-LTS that specifies the boundaries of a correct operation with respect to pressure values, This part is much in line with a two-point on-off controller. The other part represents the consumer-side workload with respect to pressure demand. The latter specification

is based on a set of assumptions made w.r.t. the consumer's behaviour. One of them imposes a cyclic pressure demand, since the ACC controller only has to work correctly for such consumers. Other assumptions relate to realistic scenarios. These assumptions are formulated as constraints over a real-valued function space, and heuristics for finding satisfying functions are put in place.

The engine that combines and executes these two parts of the specification – and thus the conformance tester – is implemented in Java. It communicates with the Simulink IUT via a TCP/IP connection, which is integrated into the IUT via a Stateflow diagram component that wraps the C-functions needed on the Simulink side to read signals from the TCP/IP channel. The application is configured by providing a configuration file in xml-format.

Indeed, with this approach it became very convenient to derive tests and execute tests in an automatic manner. Many test-runs turn out to violate stated requirements. One violated requirement concerns the assumption that the cyclic behaviour of the pump always starts with the pump being idle. This is violated by the IUT, and this must be considered a real design flaw. Other requirement violations are less directly linked to actual design flaws, because they may also be rooted in inaccuracies in the heuristics that generate the workload.

Perspective In conclusion, the testing activities on the Hydac case clearly show the benefits of model-based automatic testing. Bugs were found, and confidence was gained in the system as well as in the design process. The integration of our conformance tester into Matlab/Simulink via a Stateflow component is now very well understood and serves as a blueprint for integration with the mainstream testing tools in Quasimodo.

5 The TERMA case: Herschel/Planck satellite software architecture

The Herschel/Planck mission consists of two satellites. Herschel and Planck have different scientific objectives and thus the sensor and actuator configurations differ, but both satellites share the same computational architecture. The common architecture consists of a single processor, real-time operating system (RTEMS), basic software layer (BSW) and application software (ASW).

5.1 Schedulability Analysis of Herschel/Planck

Participants: Jacob Illum, Kim G. Larsen, Marius Mikucionis, Brian Nielsen, Arne Skou (AAU), Steen Palm (TERMA).

Context The Herschel/Planck mission consists of two satellites. Herschel and Planck have different scientific objectives and thus the sensor and actuator configurations differ, but both satellites share the same computational architecture. The common architecture consists of a single processor, real-time operating system (RTEMS), basic software layer (BSW) and application software (ASW).

Terma A/S has performed classical worst case response time analysis by analysing [71] and [72] resulting in [62]. The analysis is based on classical scheduling framework [18]. The goal of this work was to show that ASW tasks and BSW services are schedulable on a single processor and no deadline is violated. The framework uses preemptive fixed priority scheduler and a mixture of priority ceiling and priority inheritance protocols for resource sharing and extended deadlines (beyond period). One of the results of [62] is that the system is *not* schedulable on Herschel in event processing configuration, however it is argued that such situation has never been observed in testing and hence the result is too pessimistic.

The goal of the Quasimodo contribution [42] is to apply a model-based approach allowing schedulability analysis to be carried out as model checking. With this approach a more precise analysis is possible (and hence more optimistic but still realistic results) by making more assumptions explicit in the model with better control over task and resource modelling than the classical scheduling framework can offer.

The contribution [53] develops a methodology to solve the scheduling problem using the UPPAAL model checker with extensive application of the new feature of stop-watches (a natural modelling of preemption, more accurate estimates of worst case blocking-times and worst-case response times). The methodology is an extension of both the Times tool [6] (supporting only highest locker protocol [26]) and the UPPAAL scheduling framework [24] (support for multi-processor scheduling but no support for shared non-CPU resources). Our model-based analysis is much more optimistic than the classical (conservative) scheduling method because it offers more expressive modelling formalism and allows specifying and analysing the system behaviour more precisely.

Contribution We have shown how UPPAAL model-checker can be applied for schedulability analysis of a system with single CPU, fixed priorities preemptive scheduler, mixture of periodic tasks and tasks with dependencies, mixed resource sharing protocols and voluntary CPU suspension. The development proceeded in several phases:

Modelling. A complete and detailed UPPAAL TA model of tasks was created by following the classical methodology and extracting data from [62]. The resulting task model includes detailed execution patterns of individual tasks, when they request and release CPU and other resources. The model used stop-watches and was designed so that schedulability question can be answered by checking that the none of the deadlines can be violated, and the worst case response times can be obtained by estimating the supremum of a dedicated clock (response time) value.

Early results. The task execution timings were treated as deterministic and early results were obtained by limiting the search to 12 cycles as the verification memory consumption was a limiting factor.

Tackling memory consumption. An artificial progress measure of the modelled system behaviour was invented based on hypothetical hyper-period in order to apply sweep-line method [22]. The sweep-line method allowed to reuse the memory and verify a complete system model behaviour in two minutes without cycle limitations. Verification memory and time were measured for various progress measure limits and we concluded that memory consumption is reduced dramatically if it correlates with hyper-period divisors, also it is enough to have only a few distinct progress levels to achieve a reasonable precision in CPU utilisation estimates. The results are published in [56].

Relaxing determinism in execution times. The deterministic execution times are hardly realistic even on old and relatively predictable platforms as a task algorithm complexity may depend on concrete input data, thus the model was augmented with a concept of the best case execution time (BCET). As the concrete data for BCET was not available (the response time can be anywhere between zero and WCET in [62]), we set out to study times relative to WCET and determine the scalability of the verification technique in terms of the execution time non-determinism. The first experiments showed that even though the difference bound matrix (DBM) structures are fast to operate on, they were not compact as a storage for non-deterministic model state spaces, thus we switched to clock difference diagram (CDD) structures. We showed that the system is still schedulable when BCET differs from WCET by up to 10% and is probably not schedulable if the difference is greater than 14%. The non-firmness of the latter result stems from the fact that the analysis is based on stop-watch over-approximation and the found counterexamples may be spurious and not realisable. The new response time estimates (Table 1) and verification resources (Table 2) are included in [55].

The direct modelling was interesting and useful to Terma. It offered a complementary view to schedulability analysis to confirm the assumptions underlying their classical analysis (complicated due to the use of two resource sharing protocols and task dependencies).

The analysis under assumptions of deterministic worst-case execution time also provided valuable results and insights gained from visualising task execution valuable. However, due to scheduling anomalies, the worst case response time is theoretically not guaranteed to occur when a task is executing for its WCET.

We are therefore trying to relax this assumption by including BCET and more detailed task information.

The detailed task information regarding best and worst case execution time, internal task structure, and detailed synchronisation dependencies is not currently (generally) available in industrial practice for schedulability analysis. In part this is because this information is not needed for classical scheduling analysis; and hence not collected, and in part because this information is more difficult to obtain.

Analysis with execution time of 0 to WCET and any resource locking order is in principle easy to model, but do not scale to large systems (like this case with 32 tasks) due to state-space-explosion. The result may prove to be less conservative than schedulability analysis – further experience is needed.

Given more detailed information our work shows that a more detailed analysis is now a possibility, and will be technologically feasible for industrial systems in the near future. Our work also shows that this can provide more insights into how the system may behave, that are complementary to classical schedulability analysis.

Perspective Issues and directions to be considered as a future follow up:

- Is the model fair with respect to the actual system? We will evaluate this by also applying the methodology to new missions under development.
- Sporadic tasks are modelled as periodic tasks. The sporadic aspect can be modelled as non-deterministic task release, however such naive treatment results in vastly larger state space to be explored.
- The concept of the best case execution time (BCET) might not be universally applicable. For example, it might be more realistic to model the task alternating non-deterministically between resource locking when exact resource locking patterns are unknown. In this case only execution time sums and counts are known (measured from simulation), but not individual instances when the resources are locked.
- Statistical model checking could be used to estimate the probability distribution of the response times for each task and a risk model could be devised as a feedback to the user to focus on refining the more risky tasks. For example, for hard real-time case, in addition to margin analysis (slack time between response time and deadline) UPPAAL could provide a probability distribution of various margins; for a soft real-time case a probability of stepping over the deadline can be estimated.

Table 1: Task parameters and worst case response time estimates by response time analysis (Terma) and UPPAAL (0%, 5% and 10% of execution time non-determinism).

		Specification			WCRT			
ID	Task	Period	WCET	Deadline	Terma	0%	5%	10%
1	RTEMS_RTC	10.000	0.013	1.000	0.050	0.013	0.013	0.013
2	AswSync_SyncPulseIsr	250.000	0.070	1.000	0.120	0.083	0.083	0.083
3	Hk_SamplerIsr	125.000	0.070	1.000	0.120	0.070	0.070	0.070
4	SwCyc_CycStartIsr	250.000	0.200	1.000	0.320	0.103	0.103	0.103
5	SwCyc_CycEndIsr	250.000	0.100	1.000	0.220	0.113	0.113	0.113
6	Rt1553_Isr	15.625	0.070	1.000	0.290	0.173	0.173	0.173
7	Bc1553_Isr	20.000	0.070	1.000	0.360	0.243	0.243	0.243
8	Spw_Isr	39.000	0.070	2.000	0.430	0.313	0.313	0.313
9	Obdh_Isr	250.000	0.070	2.000	0.500	0.383	0.383	0.383
10	RtSdb_P_1	15.625	0.150	15.625	4.330	0.533	0.533	0.533
11	RtSdb_P_2	125.000	0.400	15.625	4.870	0.933	0.933	0.933
12	RtSdb_P_3	250.000	0.170	15.625	5.110	1.103	1.103	1.103
14	**FdirEvents**	250.000	5.000	230.220	7.180	5.553	5.553	5.553
15	**NominalEvents_1**	250.000	0.720	230.220	7.900	6.273	6.273	6.273
16	**MainCycle**	250.000	0.400	230.220	8.370	6.273	6.273	6.273
17	HkSampler_P_2	125.000	0.500	62.500	11.960	5.380	7.350	8.153
18	HkSampler_P_1	250.000	6.000	62.500	18.460	11.615	13.653	14.153
19	Acb_P	250.000	6.000	50.000	24.680	6.473	6.473	6.473
20	IoCyc_P	250.000	3.000	50.000	27.820	9.473	9.473	9.473
21	**PrimaryF**	250.000	34.050	**59.600**	**65.47**	54.115	56.382	58.586
22	**RCSControlF**	250.000	4.070	239.600	76.040	53.994	56.943	58.095
23	Obt_P	1000.000	1.100	100.000	74.720	2.503	2.513	2.523
24	Hk_P	250.000	2.750	250.000	6.800	4.953	4.963	4.973
25	StsMon_P	250.000	3.300	125.000	85.050	17.863	27.935	28.086
26	TmGen_P	250.000	4.860	250.000	77.650	9.813	9.823	9.833
27	Sgm_P	250.000	4.020	250.000	18.680	14.796	14.880	14.973
28	TcRouter_P	250.000	0.500	250.000	19.310	11.896	11.906	14.442
29	Cmd_P	250.000	14.000	250.000	114.920	94.346	99.607	101.563
30	**NominalEvents_2**	250.000	1.780	230.220	102.760	65.177	69.612	72.235
31	**SecondaryF_1**	250.000	20.960	189.600	141.550	110.666	114.921	122.140
32	**SecondaryF_2**	250.000	39.690	230.220	204.050	154.556	162.177	165.103
33	Bkgnd_P	250.000	0.200	250.000	154.090	15.046	139.712	147.160

Table 2: UPPAAL verification resource usage: verification limit is in a number of 250ms periods (∞ means unlimited), number of states is in millions, memory in megabytes, time in seconds.

limit	0%			5%			10%			14%		
	states	mem	time	states	mem	time	states	mem	time, s	states	mem	time
1	0.001	51.2	1.47	0.5	83.0	903.1	1.5	124.1	4962.8	3.3	186.9	23986.5
2	0.003	53.7	2.45	0.8	96.8	1619.9	2.4	139.7	7755.2	5.3	198.7	33299.2
4	0.005	54.5	4.62	1.5	97.2	2881.8	4.4	138.3	13720.0	9.2	274.6	51176.6
8	0.010	54.7	8.48	2.8	97.8	5325.1	9.1	156.5	31122.3	18.2	364.6	102932.4
16	0.020	55.3	16.11	5.4	112.0	9952.0	17.8	176.0	60124.5	35.4	520.4	158816.7
∞	0.196	58.8	159.64	52.7	553.9	97507.4	181.9	1682.2	530604.9	error may be reachable		

5.2 Testing of Herschel/Planck

Participants: Kim G. Larsen, Marius Mikucionis, Brian Nielsen (AAU), Steen Palm, Jan Storbank Pedersen (TERMA).

Context Terma A/S has created a test suite for the Herschel/Planck mission based on customer requirements specification and performed tests on proprietary TSIM simulator. The requirement specification contains a detailed description of the system and communication protocols at the functional level.

The goal of the Quasimodo contribution is to apply model-based testing techniques where the requirements can be formulated as a formal model and the tests derived and executed automatically. Such approach could lead to an automation and scaling of testing process, longer, more elaborate and precise tests, testing combinations of multiple requirements at the same time, and eventually prove as an efficient technique at finding faults.

UPPAAL TRON is gaining popularity at testing real-time requirements based on timed automata models.

Terma have identified a candidate set of requirements from [63] to be formalised and prepared a TCP/IP socket interface to TSIM simulator of Herschel/Planck software stack to be tested using online testing paradigm offered by UPPAAL TRON. The selected requirements correspond to software components responsible for communication link between satellites and Earth via telemetry commands.

The requirements [63] are modelled as a network of UPPAAL timed automata where the concrete requirements are reflected directly in the model and annotated with "SWR-XXX" comments. Listing 1 show the global declarations and functions used by the modelled processes and Figure 3 shows timed automata models for tele-command handling, database loading and fire functions. UPPAAL TRON adapter is being developed.

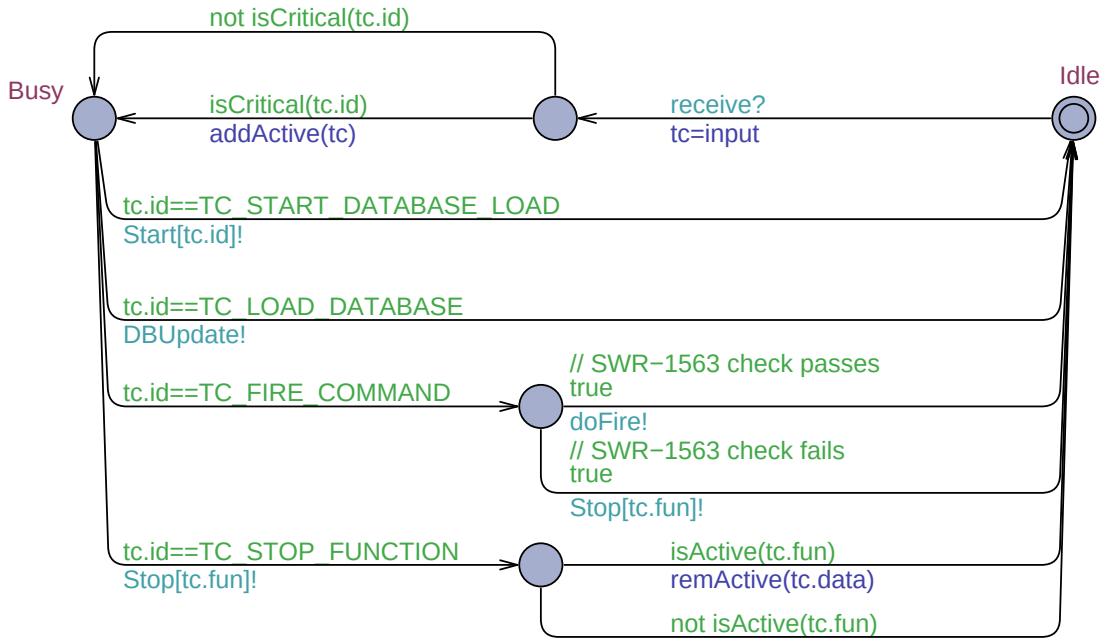
Contribution Problems being solved currently include:

- Programing the tele-command and telemetry data passing in the UPPAAL TRON adapter.
- Time synchronisation between TSIM simulator and UPPAAL TRON.

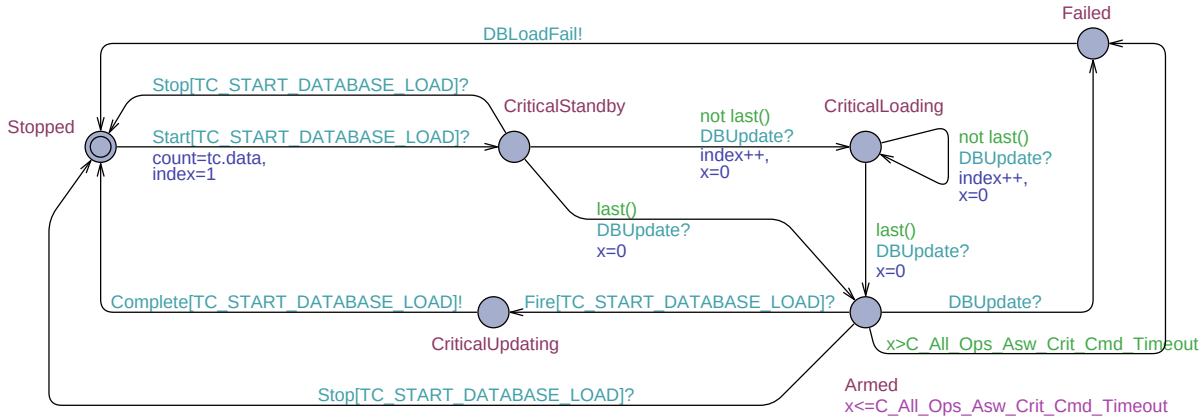
Listing 1: Shared and global declarations.

```
typedef int [0,73] ACC_ASW_TC_ID; // from H−P−4−TASW−IF−0002, Issue 3 G, page 59

const ACC_ASW_TC_ID // from H−P−4−TASW−IF−0002, Issue 3 G, page 59
    TC_LOAD_DATABASE = 1, // just the selected commands from Steen's email.
    TC_START_DATABASE_LOAD = 3,
    TC_FIRE_COMMAND = 7,
    TC_STOP_FUNCTION = 42,
    TC_REPORT_FUNCTION_STATUS = 43,
    TC_DISTRIBUTE_CPDU_COMMANDS = 44;

typedef int [1,2] Function_t ;

const int C_All_Ops_Asw_Crit_Cmd_Timeout = 1000; // SWR−607
```

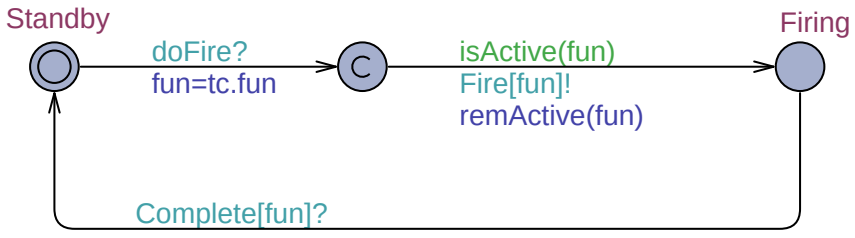
```
typedef struct {
    ACC_ASW_TC_ID id;
    int data; // this will be interpreted based on the concrete command
    ACC_ASW_TC_ID fun; // reference to the function
} TC_t;

TC_t TC_EMPTY = { 0, 0, 0}; // needed in SWR-1505

// some internal communication between processes:
chan Start [ACC_ASW_TC_ID], DBUpdate, DBLoadFail, doFire;
chan Stop[ACC_ASW_TC_ID], Fire[ACC_ASW_TC_ID], Complete[ACC_ASW_TC_ID], FireComplete;

/**
 * SWR-602: a dedicated buffer for active critical telecommands.
 * SWR-1505
 */
TC_t active [10];
int [0,11] activeLength;

// needed in SWR-602:
void addActive(const TC_t &tc) { active [activeLength++] = tc; }

// needed in SWR-1505:
void remActive(const ACC_ASW_TC_ID id) {
    int i=0;
    // locate the critical function:
    for (i=0; i<activeLength && active[i].id != id; i++);
    // remove the critical function and shift all others:
    activeLength--;
    for (true ; i<activeLength; i++)
        active[i] = active[i+1];
    active[activeLength] = TC_EMPTY;
}

// needed in SWR-1507:
bool isActive(const ACC_ASW_TC_ID id) {
    int i;
    // locate the critical function:
    for (i=0; i<activeLength; i++)
        if (active[i].id == id) return true;
    return false;
}
```



(a) Tele-command handler.



(b) Database load function.



(c) Fire function.

Figure 3: UPPAAL TA models of requirements for tele-command handling, database loading and fire functions.

6 Model-Based Testing of Electronic Passports

Participants: Wojciech Mostowski, Erik Poll, Julien Schmaltz, Jan Tretmans, Ronny Wichers Schreur (ESI/RU).

Context Electronic passports, or e-passports for short, contain a contactless smartcard which stores digitally-signed data. Access to e-passports involves several protocols, which are specified in the standards of the International Civil Aviation Organisation (ICAO) [40]. The second generation of e-passports introduced in the EU in the summer of 2009 contains sensitive biometric data, namely fingerprints. To protect this information, the EU mandates the use of a strong security mechanism called Extended Access Control protocol (EAC) [17].

The challenge of this work, published as [57], was to rigorously test the second generation of Dutch e-passports for conformance of their access protocol implementations using model-based testing.

Contribution For model-based testing, we first developed formal models of the e-passport access protocols. The official standards give long and detailed descriptions of the individual protocols. Understanding these descriptions and the combination and possible interaction of the various protocols was difficult. We started with conceptual models in the form of finite state machines drawn on a white-board and then gradually transformed these state diagrams into symbolic labelled transition systems, which are input for the model-based testing tool TORXAKIS.

TORXAKIS is based on the model-based testing tool TORX [76] extended with symbolic test generation capabilities to deal with quantitative information from large data domains [27]. The underlying theory is the *ioco*-test theory for labelled-transition systems [75] extended for symbolic labelled transition systems [27]. TORXAKIS performs random walks through a model, it sends commands to the e-passport chip and verifies that the responses conform to the model. TORXAKIS is implemented in Haskell.[1] For lower-level communication with the e-passport chip we used a card reader with the open source passport-terminal software that we helped develop in the JMRTD project.[2]

Having a model and a model-based testing tool we tested actual second-generation e-passports. The tests were run fully automatically: overnight tests were able to perform over $1{,}000{,}000$ protocol steps per test on an e-passport. This gave sufficient confidence that the access protocols were correctly implemented.

The most difficult part of the testing process was understanding the official specifications and constructing a formal model for them. Finite state diagrams turn out to be a very effective and perspicuous way to specify the combination of passport protocols. Indeed, it amazes us that the official specifications do not use finite state diagrams anywhere. Once we had that model, and the test infrastructure (connection of the model-based testing tool to the card reader, etc.) the actual testing only took less than a week. During this period we also adapted and played with different models. By slightly changing the model, new tests are derived automatically with

[1] `http://www.haskell.org`
[2] `http://jmrtd.org`

a simple key stroke. By refining and tweaking the model we could quickly find out how any underspecification or unclarities in the specifications had been resolved in the implementation that we tested.

Perspective The e-passports were successfully tested with model-based testing, and much longer and more thorough testing was performed than would have been possible with traditional testing.

Yet, e-passports are relatively simple request-response systems. They use data parameters which makes that they require symbolic labelled transition systems and TORXAKIS, instead of plain labelled transition systems and TORX, but they are not real-time, nor do they show nondeterministic behaviour or concurrency. The next step will be to perform model-based testing with systems that add nondeterminism, concurrency, real-time, and more complexity. We intend to test the gMAC protocol layer of the Chess Wireless Sensor Network node, a Quasimodo case study. This gMAC protocol has real-time behaviour and exhibits concurrent and nondeterministic behaviour.

7 Bike Braking WSNs

This work is published in *IEEE WoWMoM 2011* [30].

Participants: Hernán Baró Graf, Holger Hermanns, Juhi Kulshrestha, Jens Peter, Anjo Vahldiek, Aravind Vasudevan (SU).

Context Classical abstractions used in software engineering leave out "nonfunctional" aspects, such as cost, efficiency and robustness. In particular in the field of embedded software, there is a growing awareness that these abstractions no longer suffice to arrive at dependable designs [2, 54]. Embedded software is subject to complex and permanent interactions with its, mostly physical, environment via sensors and actuators.

The future will likely bring an increase in wireless technology also in the context of safety-critical control applications. Wireless communication is known to be inherently unreliable and often is characterised by relatively high message loss rates. When hard real time requirements are to be met despite wireless communication, it becomes even more difficult to come up with safety guarantees. The central problem is that consecutive failures in message transfer may affect the correct functioning.

There are two specific application conditions under which this problem may be overcome: First, it is possible – if the application allows it – to run the control loop on a pace that is slow enough such that the probability of a prohibitive number of consecutive message losses is clearly negligible. Second, if the systems allows for a fail-safe state, it is possible to force the design into that state whenever the number of consecutive losses exceeds some justifiable bound. If not well configured, this may lead to a system that is barely operational, but it is a safe design after all.

But what to do if neither of the two conditions are met? If there is no fail-safe state and if the controller must react within a very limited time window? A responsible designer will likely react very reluctantly to the idea of solving with wireless technology a control problem that is hard real-time, safety-critical and does not offer a fail-safe state to fall back to.

In this work we are looking at a very tiny control problem of precisely that sort. It is safety-critical, has hard real-time requirements and does not have an obvious fail-safe state. We are looking at the brakes of an ordinary bicycle and are investigating what happens when the mechanical connection is replaced by a wireless sensor and actuator pair. We report here about the modelling, verification, design and construction of such a *wireless bike brake*.

This project was originally conceived as the *mad bike project*, and there is some craziness in this idea after all. However we think that it is a very good case to study the principal possibilities and limitations of wireless control without going to excessive infrastructure costs. Indeed, our investigations allow us to discriminate between different options to solve this and similar problems with different dependability guarantees.

The dependability guarantees we can give are quantitative – or probabilistic – guarantees. We guarantee that the probability of the system to not react within a safe time window is low, where the precise number depends on the assumptions made about timing and individual message loss

probabilities. We measure these individual loss probabilities on the real design and use these measurements as input to our model-based analysis. In fact, we consider both a setup with message loss probabilities in the order of 10^{-5} as well as a more challenging scenario with losses occurring for more than 50% of the messages sent.

The bike brake system has very strict real time timing requirements. The time between the rider applying the brake by pressing the handle to the braking force actually being applied, has to be short enough to ensure the safety of the rider. The time for applying the brake includes the time for the force sensor to notice the difference in the force being applied, conveying it to the sender, the sender transmitting these values wirelessly to the receiver and its informing the actuator to apply the braking force. The timings of all these steps in the braking process are also limited by the hardware being used.

Contribution For the purpose of studying quantitative dependability, we resort to the Modest language, since it supports the specification of real-time as well as probabilities. In fact, we restrict to the subset that corresponds to probabilistic timed automata in our specifications. The Modest specifications are submitted to the *mcpta* analysis engine, see Deliverable 5.9. This model checker enables us to arrive at the above guarantees. It can also derive average response time bounds and other measures. It does so by using the PRISM model checker [47] as a back end, encoding time as a variable automatically. This is likely the first concrete use case of this kind, mainly because PTA model checkers debuted only very recently [77, 36, 60]. The Modest tool set also includes a simulation engine *modes*, which we use for validation purposes, namely to link between verification results and empirical measurements.

Adding redundancy into a system is a classical concept to improve the overall reliability. To experiment with this idea we introduce a generic $replicator$ node, that acts as a $receiver$ and a $sender$ at the same time. At a glance it is obvious that any type of redundancy will improve the results, but it should be considered that adding a node induces more communication, which in a TDMA setting is naturally accommodated by extending the number of slots per frame accordingly. This means that the more nodes we introduce as senders the longer the frames become. If we want comparable results and we know that we need a reaction within 150 ms we must keep a fixed number of slots. For practical reasons we established 12 slots in 150 ms and we divide the frames accordingly to the number of transmitting nodes.

To study the effect of replication, we add in parallel a set of nodes as $replicators$, extending the model with 1, 2 and 4 $replicators$. The change to the model is minimal.

$Rep\backslash t$	150 ms	1 sec	10 sec
0	0.0175963	0.2796740	0.9732774
1	0.0221659	0.3091858	0.9844734
2	0.0304205	0.3709793	0.9935737
4	0.0639192	0.5259302	0.9996357

Table 3: Maximum probabilities of crashing within t

Table 3 reports on a series of verification runs for the above requirement that are summarised

in the table below. In this setup, we run the model checker for different time constraints t: 150 ms, 1 s and 10 s. We configure the model using 0, 1, 2 and 4 replicators, Rep, to compare reliability (thus in 150 ms we have 6, 4, 3 and 2 frames respectively). We also need to vary the corresponding maximum consecutive losses (in frames), i.e. MAXLOST $= 6, 4, 3$ *or* 2. From the experiments we adjust the probability of a single message lost p to 51%, which corresponds to the average obtained from the experiment logs, for the chosen setup. The results were produced within 1 to 5 minutes, depending on t, in a dual-core notebook with 3 GB of RAM.

Perspective This work has focussed on the design, modelling, verification, simulation, construction, and deployment of a real case, a prototypical bike with wireless brakes. For the safety of the rider, we determine that is imperative that the brake shoe reacts within no more than 250 ms to a command issued by the brake handle. For the current prototype, the delay by the mechanical and conversion components is about 100 ms, which leaves some 150 ms for a successful communication between the wireless partners. Measurements show that an unfortunate configuration in DSA mode can lead to message loss probabilities around 50%. According to our model checking results, this implies that a bare communication delay of 150 ms cannot be guaranteed in 1 out of 50 brake attempts. Using a replicator network to add redundancy to the communication is revealed to be counterproductive by the model checker, and by experiments, if one takes the increased round timing into account. This insight is non-obvious, and is obtained by state-of-the-art PTA model checking and simulation and later confirmed with several experiments. We suppose it is of general interest to designers of wireless dependable systems to be able to study whether simple replication mechanisms can improve the safety guarantees.

Finally, the key to arrive at a safe design is to drastically reduce the individual message loss probabilities. For the MyriaNed system, this is achieved – maybe not surprising – by avoiding randomness is slot assignment, using the fixed slot allocation scheme FSA. The model checker enables us to readily prove that this twist results in a design with very high reliability guarantees, far beyond the "five-nines" yardstick 99.999%. In a nutshell, our model checking studies clearly hint at the potential tradeoffs when designing such systems.

On the experimental side we would like to work on a second prototype, that will incorporate several improvements to improve on the reaction time guarantees and to amplify rider convenience:

- a hydraulic disc brake with a more direct apparatus to apply and adjust the brake force, combined with an anti-lock braking system (ABS).
- a force sensor in the brake handle with a force feedback system.

8 Model-based Testing of a Software Bus at Neopost

Participants: Mariëlle Stoelinga, Axel Belinfante (ESI/UT),
Marten Sijtema (Sytematic Software, NL),
Lawrence Marinelli (Neopost, Texas, USA).

Context In [69], we report on the actual industrial use of model-based methods during the development of a software bus. The central claim made by the field of model-based testing is that, while it requires an initial investment to develop rigorous models and perform rigorous testing, these pay off in the long run in terms of better, and more maintainable code. We investigate this claim the development of a software bus.

Contribution At Neopost Inc., we developed the server component of a software bus, called the *XBus*, using formal methods during the design, validation and testing phase: We modelled our design of the XBus in the process algebra mCRL2, validated the design using the mCRL2-simulator, and fully automatically tested our implementation with the model-based test tool JTorX. This resulted in a well-tested software bus with a maintainable architecture. Writing the model, simulating it, and testing the implementation with JTorX only took 17% of the total development time. Moreover, the errors found with model-based testing would have been hard to find with conventional test methods. Thus, we show that formal engineering can be feasible, beneficial and cost-effective.

Developing the XBus. We have deployed model-based testing and the model-based test tool JTorX during the development of the XBus at Neopost Inc. Neopost is one of the largest companies in the world producing supplies and services for the mailing and shipping industry, like franking and mail inserting machines, and the XBus is a software bus that supports communication between mailing devices and software clients. The XBus allows clients to send XML-formatted messages to each other (the X in XBus stands for XML), and also implements a service-discovery mechanism. That is, clients can advertise their provided services and query and subscribe to services provided by others. We have developed the XBus using the classical V-model [65], using formal methods during the design and testing phase. The total running time of this project was 14 weeks. An important step in the design phase was the creation of a behavioural model of the XBus, written in the process algebra mCRL2 [32, 4]. This model pins down the interaction between the XBus and its environment in a mathematically precise way. Performing this modelling activity greatly increased the understanding of the XBus protocol, which made the implementation phase a lot easier. After implementing the protocol, we tested the implementation against the mCRL2 model, using JTorX. JTorX [10, 3] is a model-based testing tool is capable of automatic test generation, execution and evaluation and is (partly) developed during the Quasimodo-project. During the design phase, we already catered for model-based testing, and designed for testability: we took care that at the model boundaries, we could observe meaningful messages. Moreover, we made sure that the boundaries in the mCRL2 model matched the boundaries in the architecture. Also, to use model-driven test technology required us to write an adapter. This is a piece of software that translates the protocol messages from the

mCRL2 model into physical messages in the implementation. Again, our design for testability greatly facilitated the development of the adapter.

Our findings. We ran JTorX against the implementation and the mCRL2 model (once configured, JTorX runs completely automatically) and found five subtle bugs that were not discovered using unit testing, since these involved the order in which protocol messages should occur. After repairing these, we ran JTorX several times for more than 24 hours, without finding any more errors. Since writing the model, simulating it, and testing the implementation with JTorX only took 17% of the total development time (counting only human working time), we conclude that the formal engineering approach has been very successful: with limited overhead, we have created a reliable software bus with a maintainable architecture. Therefore, we clearly show that formal engineering is not only beneficial for large, complex and/or safety-critical systems, but also for more modest projects.

Bugs Found Using JTorX. One of the most interesting part of testing is finding bugs. In this case, not only because it allows improving the software, but also because finding bugs can be seen as an indication that model based testing is actually helping us. We found 5 bugs of which we think that they are hard to find without a tool like JTorX. JTorX discovered these bugs within a few seconds. Also, JTorX's tracing facility proved to be particularly useful for debugging: when a failure occurs, JTorX records the sequence of actions leading to the error. This trace has proven instrumental in pointing down the source of the failure.

Findings and Lessons Learned. So how long did it take to create the artefacts for model-based testing, namely the model, the test interface and the adapter? Programming and simulating the model took 2 weeks, or 80 hours. The test interface was created in a few hours, since it was designed to be loosely coupled to the engine. It was a matter of a few dozens lines of code. The adapter was created in two days, or 16 hours. Thus, given the total project time of 14 weeks, creating the artefacts needed for model-based testing thus about 17% of our time. Writing a model takes a significant amount of time, but also forces the developer to think about the system behaviour thoroughly. Moreover, we found it extremely helpful to use simulation to step through the protocol, before implementing anything. Making and simulating a model gives a deep understanding of the system, in an early stage of development, from which the architectural design profits. Writing an adapter can sometimes be a large project, but in this case it was relatively straightforward. This can be attributed to having an architectural design that closely resembles the formal model, and having a one-to-one mapping between the actual XBus messages and their model representation.

Perspective We conclude that model-based testing using JTorX was a success: with a relatively limited effort, we found five subtle bugs. We needed 17% of the time to develop the artefacts needed for model-based testing, and given the errors found, we consider that time well spent. Moreover, for future versions of the XBus, JTorX can be used for automatic regression tests: by adapting the mCRL2 model to new functionality, one can detect automatically if new bugs are

introduced. We also conclude that making the formal model together with the architectural design had a positive effect on the quality of the design. Moreover, the resulting close resemblance between model and design simplified the construction of the adapter.

9 Verification of Printer Datapaths Using Timed Automata

Participants: Georgeta Igna, Frits Vaandrager (ESI/RU).

Context Modern embedded systems are characterised by distributed implementation platforms that include a heterogeneous mix of several processors, one or more buses for communication, and a variety of sensing and actuating devices. They have to operate in dynamic and interactive environments, and need to carry out a mix of data-intensive computational tasks and event-processing control tasks. Not only functional correctness is important, but also quantitative properties related to timeliness, quality-of-service, resource usage and energy consumption.

The complexity of embedded systems and their development trajectories is thus increasing rapidly. At the same time, development trajectories are expected to deliver products that are inexpensive and performing, while meeting stringent time-to-market constraints. The complexity of the designs and the constraints imposed on the development trajectory dictate a systematic, model-driven design approach that leverages reuse and is supported by tooling whenever possible. In multiprocessor systems with many data-intensive tasks, a bus may be among the most critical resources, and severely degrade the timing predictability.

The problem is that allocation of bandwidth to one (high-priority) task may lead to a reduction of the bandwidth of other tasks, and thereby effectively slow down these tasks. If we do not want this to occur, for instance in the case of safety critical systems, then we may use e.g. a time division multiple access (TDMA) strategy on the buses in order to give each task a guaranteed bandwidth. However, for most systems such a solution is too expensive. According to Williams et al. [82], for the foreseeable future off-chip memory bandwidth will often be the constraining resource in system performance of multicore computers. Clearly, WCET analysis for such systems is a major research challenge.

Contribution Existing performance analysis techniques are not able to accurately predict WCETs for systems with this type of highly dynamic resource behaviour. Simulation of detailed models certainly provides insight, but fails to provide WCETs in settings with uncertain job arrival times, dynamic and interactive environments and/or uncertain processing times. In this paper, we show how the dynamic behaviour of a memory bus and a USB in a realistic printer application can be faithfully modelled using timed automata.

In addition, we show how to compute WCETs (latencies) for the application using the model checker Uppaal [8, 51, 9]. The case study that we describe here originates from the Octopus [59] project. Octopus is a cooperation between Océ Technologies, the Embedded Systems Institute and several academic research groups in the Netherlands. Its objective is the development of new methods and techniques to support model-driven design space exploration for embedded systems. Some preliminary work from the Octopus project was reported in [41]. There, we considered a simplified version of an Océ printer architecture. Using this architecture, we studied the differences among three modelling formalisms and supporting tools used in the project: Uppaal [8, 51, 9], Colored Petri Nets [45, 44] and Synchronous Dataflow Graphs [29, 68]. In this paper, we present a detailed model of a realistic printer design which, in particular, includes a

description of the scheduling rules used by the Océ printer controller. We analyse, using Uppaal, the worst case latency of scan jobs with uncertain arrival times in a setting where the printer is concurrently processing an infinite stream of print jobs. The purpose of this paper is to show that the Uppaal model checker can handle the complexity of dynamic memory bus behaviour in a realistic model of a complex industrial application. To the best of our knowledge, no other analysis technique/tool, except maybe the hybrid method of [49], is currently able to do a performance analysis for this type of systems (involving a dynamic memory bus and uncertain arrival times). Existing techniques for WCET analysis of distributed embedded systems, such as Modular Performance Analysis [20, 74], SymTA/S [34] and MAST [35] are not applicable since they lead to overly conservative analysis results. In [49], a hybrid method is proposed for analysing embedded real-time systems that integrates modular performance analysis and timed automata. It would be interesting to use our detailed Uppaal models of the memory bus and USB as part of this hybrid method.

We have analysed an Océ printing machine and two of its datapaths. We computed the worst latency of one datapath which has uncertain arrival time and the other datapath is infinitely often used. Our results show a strong dependency between the two datapaths.

Perspective As usual with model checking, long running time was a key issue within our case study. In order to be able to do the model checking (within reasonable time), we had to slightly scale down some of the parameters in the model. Still, the current version of Uppaal is close to the point where it can handle the complexity of industrial designs. One technical issue that we faced is that although essentially the behavior of the model is fully deterministic when all the scheduling rules are added, the resulting Uppaal model is not (and suffers from state space explosion) due to interleaving of internal actions of the various resources. We resolved this by using the channel and process priorities from Uppaal, but a better solution would be to extend Uppaal with support for confluence detection and/or partial order reduction.

We computed the worst latency by repeatedly checking an invariant property. Using a binary search we managed to find the exact value of certain parameters. However, this type of parametric analysis requires a lot of time and it would be most helpful to mechanise it using Uppaal, possibly using multiple processors to parallelize computations. A lesson that we have learnt is that it is extremely difficult to maintain correctness of the model in a setting where the object of modelling has such a high complexity. There was not a single document describing the design. In fact there was not a single person who was able to answer all our questions: the knowledge was spread over a large design team. For the engineers it is difficult to understand the intricacies of our Uppaal model. The syntax of Uppaal is not sufficiently expressive to describe the design in such a way that a small change in the design corresponds to a small change in the model. Due to these difficulties, the Octopus project has decided to develop a high level language for describing the designs, together with a translation to Uppaal: on one hand this will make it much easier to communicate with the engineers, and on the other hand it will reduce the chances of introducing errors in the Uppaal model.

10 Formal Specification and Analysis of Zeroconf Using Uppaal

Participants: Jasper Berendsen, Biniam Gebremichael,
Frits W. Vaandrager (ESI/RU),
Miaomiao Zhang (Tongji University,Shanghai, China).

Context Our society increasingly depends on the correct functioning of modern communication technology. Most prominent are (mobile) phones and the Internet, but there are also networks in modern cars, trains, and aeroplanes, and the new generation of consumer electronics allows all sorts of devices to communicate with each other. The most important and most often used protocols that describe the operation of these networks are standardised. Examples of this are the Internet protocol (TCP/IP), FireWire/iLink (IEEE 1394), HAVi, WAP, CAN, and BlueTooth. Due to a combination of factors, the complexity of these protocol standards is often very high: rapid changes in the capabilities of the underlying hardware, the fact that often many (industrial) parties are involved in standardisation, each with its own interests, and market demands to extend the functionality of the protocols. Since these standards serve as a guide to implementers from many different companies, with different backgrounds, it is vital that standards only allow for one clear interpretation, are complete, and ensure the required functionality for each implementation. For most protocol standards, this is clearly not the case. In fact, it is surprising that protocols which are of such immense importance to our society are typically written in informal language, with frequent ambiguities, omissions, and inconsistencies. They also fail to state what properties are expected of a network running a protocol, and what it means for an implementation to conform to a standard.

By now there is ample evidence that formal (mathematical) techniques and tools may help to improve the quality of protocol standards. Numerous publications describe the formal modelling and analysis of critical parts of protocols, and via these case studies many previously undetected bugs have been detected (see, e.g., Clarke et al. [23], Bruns and Staskauskas [16], Devillers et al. [25], Langevelde et al. [50], Stoelinga [67], Holzmann [39], Chkliaev et al. [21], and Vaandrager and Groot [79]). In most cases, these studies were carried out after the completion of the standards, and involved guessing to fill in holes and resolve ambiguities. An exception is the work by Romijn [64], who aimed at applying formal methods already during the standard development process. That effort has resulted, for instance, in the discovery and correction of many errors, omissions, and inconsistencies, as well as the addition of correctness properties, in the IEEE 1394.1 FireWire Net Update standard.

In order to avoid holes and ambiguities in standards, the obvious way to go is to describe critical parts using formal specification languages, similar to the way in which diagrams are used to specify the electrical circuits and mechanical parts. There have been joint attempts of academia and industry to arrive at formal description languages for protocols. The most notable attempts at this have been the LOTOS and SDL standardisation efforts. However, to the best of our knowledge, these languages have thus far not been used in the authoritative part of protocol standards. Some protocol standards have extended finite-state machines (EFSMs) inside, but

these are mostly illustrative, not completely formal, and sometimes contain mistakes. Bruns and Staskauskas [16] used (a well-defined subset of) C to describe the SONET Automatic Protection Switching (APS) protocol and reported that developers found their C description easy to understand and superior to that which appeared in the APS standard. However, the lack of abstraction mechanisms is an obvious drawback of C.

The relationships between an (abstract) formal model of a protocol and the corresponding informal standard are typically obscure. As pointed out in Brinksma and Mader [15], page 1: Formal Specification and Analysis of Zeroconf Using Uppaal: "Current research seems to take the construction of verification models more or less for granted, although their development typically requires a coordinated integration of the experience, intuition and creativity of verification and domain experts. There is a great need for systematic methods for the construction of verification models to move on, and leave the current stage that can be characterised as that of model hacking. The ad-hoc construction of verification models obscures the relationship between models and the systems that they represent, and undermines the reliability and relevance of the verification results that are obtained." As a step toward the development of a systematic method, we report in this article on the systematic construction of a verification model of a recent protocol standard.

More specifically, we describe the use of Uppaal to model and analyse critical parts of Zeroconf, a protocol for dynamic configuration of IPv4 link-local addresses. Our goal has been to construct a model that (a) is easy to understand by engineers, (b) comes as close as possible to the informal text (for each transition in the model there is a corresponding piece of text in the standard), and (c) may serve as a basis for formal verification.

Contribution There are many situations in which one would like to use the Internet Protocol for local communication, for instance, in the setting of in-home digital networks or to establish communication between laptops. For these type of applications, it is desirable to have a plug-and-play network in which new hosts automatically configure an IPv4 address, without using external configuration servers, like DHCP and DNS, or requiring users to set up each computer by hand. The Zeroconf protocol has been proposed to achieve exactly this. It describes how a host may automatically configure an interface with an IPv4 address within the 169.254/16 prefix that is valid for communication with other devices connected to the same physical (or logical) link. The most widely adopted Zeroconf implementation is Bonjour from Apple Computer, but several other implementations are available.

The contribution of this article is, first of all, a formal model of (a critical part of) Zeroconf, a protocol with clear practical relevance, that is easy to understand, faithful to the RFC, and with an extensive discussion of the relationship between the model and the RFC. Our modeling efforts revealed several errors (or at least ambiguities) in the RFC that no one else spotted before. We present two proofs of the mutual exclusion property for Zeroconf for an arbitrary number of hosts and IP addresses: a manual, operational proof, and a proof that combines model checking with the application of a new abstraction relation that is compositional with respect to committed locations. The model checking problem has been solved using Uppaal and the abstractions have been checked by hand.

Perspective Our goal has been to construct a model of Zeroconf that (a) is easy to understand by engineers, (b) comes as close as possible to RFC 3927, and (c) may serve as a basis for formal verification.

In this study, we have modelled and analysed a fragment of Zeroconf in a restrictive setting without faulty nodes, merging of subnetworks, etc. In order to deal with dynamically changing network topologies, a more sophisticated use of abstractions will be required, for instance along the lines of Bauer [7]. An obvious challenge is to mechanize all these abstractions using either (an extension of) UPPAAL-TIGA (Cassez et al. [19]) or a general-purpose theorem prover. The timing behaviour of Zeroconf becomes really interesting when studied within a setting in which also the probabilistic behaviour is modeled. The performance analysis of Zeroconf reported in Bohnenkamp et al. [11] and Kwiatkowska et al. [48] has been carried out for an abstract probabilistic model of Zeroconf. A challenging question is whether these results also hold for a (probabilistic extension) of our more realistic model.

11 Model Based Testing for ASML Case Study

Participants: Jiangsheng Xingj (ESI/UT), Bart Theelen (ESI),
Jeroen Voeten, Jan Tretmans (ESI),
Rom Langerak, Jaco van de Pol (ESI/UT).

Context To deal with the increasingly design complexity within ever-shortening times of software/hardware systems. System level design methodologies have been widely used to assist in choosing well-founded design decisions. POOSL (Parallel Object-Oriented Specification Language) is a powerful general purpose system level modelling language which has been used in many academic and industrial case studies. In research on design space exploration of motion control systems, POOSL was used to construct models for evaluating system performance under different configurations. The considered motion control algorithms are characterised by periodic execution and distributed in multiple processors, which are interconnected by RapidIO (Rapid Input/Output) packet switches. POOSL analysis gives estimation results for worst-case packet latencies and average-case packet latencies, which are essential performance criteria for motion control systems.

However, most motion control systems are time-critical and safety-critical. Worst-case packet latencies are strict timing constraints. Exact worst-case packet latencies are to be determined, which is out of the capability of the POOSL approach. Motivated by this requirement, we applied model checking techniques by transforming the original POOAL model into an UPPAAL model. We investigated the main concepts and elements of the POOSL language and summed up several patterns for the transformation from POOSL to UPPAAL. With these transformation patterns, we obtained an UPPAAL model from the original POOSL model. With this UPPAAL model, we verified some functional behaviours such as deadlock freedom as well as worst-case packet latencies. Moreover, we showed that the analysis of average-case packet latencies can also be accomplished with assistance of the UPPAAL simulator.

Given the advantages of the model checking approach using UPPAAL, another question emerges: is the transformation correct? Or in other words, does the UPPAAL model have the same behaviour as the original POOSL model? For this purpose, we propose to use MBT (model based testing) tool UPPAAL Tron (Tron for short hereinafter). The main purpose of the MBT approach is that: checking if the behaviour of the SUT (System Under Test) conforms to the behaviour described in the system specification. By defining the POOSL model as SUT and the UPPAAL model as the system specification, we can check if the behavior of the transformed UPPAAL model conforms to the behaviour specified in the POOSL model. Another motivation is that the above setting has not been investigated before for Tron. In this case study, the SUT is a simulator which has its own virtual clock. The time synchronisation between the SUT and Tron complicates the communication and a special protocol is needed. This kind of setting is new but it may exist in many application domains. So it is a generic setting worthy of investigation.

Contribution Tron can check whether the timed runs of the SUT are specified in the system model (similar to timed trace inclusion) and no illegal (unexpected, unspecified) timed behaviour

is observed. Time is considered continuous, input/output events can happen at any real valued moment in time, but deadlines are constrained by integers. The specification is an UPPAAL timed automata network partitioned into a model of the system and a model of the system's environment assumptions. The model can be non-deterministic, allowing reasonable freedom for system implementations, modelling possible/tolerable time drifts, soft time deadlines. Test primitives are generated directly from the model, executed and the system responses checked at the same time, online (on-the-fly) while connected to the SUT, thus avoiding huge intermediate test suites. In our setting, the UPPAAL model serves as the system specification and the POOSL model serves as the SUT. Each model is divided into two parts: a system part and a system environment part. Tron interprets the system environment part of the UPPAAL model and generates inputs to the POOSL model, the POOSL model progress according to the input and generates output to Tron. Tron then check conformance by comparing the output from the POOSL model with the corresponding results of the system specification.

Tron interprets the system environment part of the UPPAAL model and then sends input to the Adapter; the Adapter translates the abstract input into concrete input and relays it to the SUT. The SUT conducts this concrete input action and then sends output (if exists) back to the Adapter. The adapter translates the concrete output into abstract output and then relays it to Tron. Tron then checks if this output is valid. If so, Tron will continue the testing with the next input. If not, Tron will record and report the failure event and exit testing.

SocketAdapter has been provided for the communication between Tron and the SUT. However, current implementation only considers the scenario when the SUT and the Adapter are in the same process space (Adapter can call the function of SUT directly). So the communication is constrained between Tron and the Adapter (adapt the information between Tron and the SUT). For the ASML case study, SUT (the POOSL model) cannot be in the same process space with the Adapter. Some kind of communication means must be provided. Socket is the most convenient way for that purpose and a protocol is needed as mentioned before. As the protocol mainly concerns the time synchronisation between the two models, we first introduce how time is handled in Tron.

The Virtual Time Framework in Tron . Tron employs a virtual time framework. The purpose of the framework is to provide "lab" conditions for testing software where the value of a global reference clock is controlled and detached from physical time. Such framework allows testing time delays specified in software in ideal conditions where the time spent on computation and communication is treated as zero. If the computation and/or communication time is known and needed to be taken into account, then such delays can be replaced by "timed-wait" calls and an abstraction of control software can be tested under ideal conditions. The virtual time framework assumes the following protocol:

- There is a single instance of a global clock in the entire test setup.
- All participating threads follow these rules:
  - Register its presence at the global clock.
  - All time-related system calls are redirected to the global clock.

  - Each thread should perform some computation and eventually call to wait for some condition to occur.
  - The thread computation time is assumed to be negligible and only the waiting times are significant.

- The global clock serves the threads in the following way:
  - If there is at least one thread computing and not waiting, then the clock value stays constant.
  - If all threads are waiting for condition to occur, then the global clock finds the smallest clock increment which would trigger a timeout at least for one thread, increments the clock and notifies appropriate threads.

In sum, there are three clocks involved:

1. the global time clock in the UPPAAL model
2. virtual clock (the global reference clock in the virtual time framework which is interchangeable with the real-world/host clock)
3. the clock used by SUT.

Tron does some scaling and offsetting when translating between 1 and 2 (e.g. testing starts with model clock = 0, while the virtual or a host clock can have any initial value; then 1 model time unit stands to some fixed amount of real/virtual time). However, the scaling and offsetting are transparent for users and we do not care for it. So the main problem and our focus is the synchronisation of 2 and 3. In the next section, we'll explain how virtual clock and the clock used by SUT can be synchronised.

The Protocol between SUT and Tron.
The main idea of the protocol is:

<-- Tron sends input actions which are immediately executed by POOSL (let's say: a@0, where '0' is the relative delay). For this input (refers to the injection of a packet), we need to specify command type (packet injection), packet index (if possible), source end point, destination end point, priority (optional). We only consider equal-sized packets and thus size parameter is not needed.

<-- Tron can also ask for outputs for a particular time period: request-for outputs up to t. For this input (which corresponds to a labelled edge following a timed delay and then an output channel in the UPPAAL model), we need to specify command type (timed delay), t (number of time units for delay), outputs to be monitored (none, one or more; none means monitoring all outputs, one or more means monitoring outputs at one or more specific end points). Outputs monitoring selection can also be implemented as a separate command.

--> if POOSL has an output at some time $t' <= t$ then it sends that output: x@t', and then Tron can continue. For this output, we need to specify the command type (packet received), packet index (if possible), destination end point, model time (assume sending output instantly).

--> if POOSL does not have any output in $timeframe <= t$, then POOSL responds with 'no output up to t'. Theoretically, this corresponds somehow to a timed quiescence. For this output, we need to specify the command type (no output), model time.

MBT Experiment Results. The environment part of the UPPAAL model interacts with the system part with two channels: the input channel mi and an output channel mo. The system part is almost the same as we have discussed in our previous paper (we just remove the packet injection part and add two corresponding channels mi? and mo!, omitted for simplicity).

In our setting, the input channel normally represents an input packet into the system. The output channel represents a received packet which has arrived at its destination. However, for this setting, we can only check the worst-case latency property for a specific packet. We have also extended the environment model such that a non-deterministically selected packet is released and its worst-case packet latency property is checked w.r.t to its implementation (the POOSL model).

For the above two settings, we have run the corresponding MBT experiments. The results show that the POOSL model conforms to the UPPAAL model w.r.t. the worst-case latency property. The results also indicate that the protocol works for our settings and it might be extended to apply to other scenarios. The experiments are easy to implement. However, the underlying MBT framework is hard to implement and it took us much time for debugging and polishing the protocol. Although Tron is a powerful MBT tool, it requires professional knowledge of Tron and underlying system model. More work is needed to make Tron easy to use and extend its applicability.

Perspective In this case study, we investigate the behaviour conformance between the POOSL model and its transformed UPPAAL model. First, a protocol is designed for the communication between Tron and SUT (the POOSL model). The main difficulties exist in the synchronisation of time in these two models. Initial experiment results show that the transformed UPPAAL model conforms to the POOSL model w.r.t. the worst-case packet latencies. As our future work, we would like to polish the protocol such that more complicate behaviours can be specified. Then we'll extend the system environment model and investigate in more detail the behavior conformance of the two models (such as packets serving orderings).

12 The impact of GSM-R on railway capacity

Participants: D. N. Jansen (ESI/RU),
S. G. Klabes, E. Wendler, (RWTH).

Context The operation of railway systems strongly depends on the underlying train control system. ERTMS (EuropeanRail Traffic Management System) is a project launched by the European Union in order to increase the interoperability of the national railway systems in Europe. One of the two main components of ERTMS is GSM-R, a wireless communication standard based on GSM. In this paper, we focus on the stochastic nature of GSM-R communication failures and their possible impact on railway capacity. Firstly, we will compare the results of our newly introduced model to the results obtained with a standard blocking time model, applied by a couple of European railway infrastructure managers.

After this validation of our model, we will then use the stochastic approach in order to evaluate the impact of GSM-R communication on the railway operation and railway capacity with ETCS level 3. We can show that ETCS level 3 indeed leads to a capacity increase in our setting. So, while a single GSM-R message may be more error-prone than traditional communication, the framework of ETCS can cope well with this imperfection.

Contribution Railway systems know a long history of train protection and control, as to reduce the risk of train accidents. Many systems include some automated communication between train and track side equipment. Several different, mostly national systems emerged [73]. The different train control systems are still a major obstacle for crossborder rail traffic. Today, trains for crossborder traffic need to be equipped with all train control systems installed on the tracks that the train utilises during its journey. The European Rail Transport Management System (ERTMS) shall lead to a harmonisation of the European train control systems. It is one of the backbone projects to achieve higher interoperability between the different train control systems used in European countries in the hope to increase the share of rail transport on the overall transport in Europe.

ERTMS consists of two standards: the European Train Control System ETCS and the Global System for Mobile communication for Railway applications GSM-R [83]. Today ETCS is foreseen to have three levels (1, 2 and 3). Level 1 defines a standard for discontinuous train control with standardised hardware. Level 2 replaces traditional line side signals by transmission of movement authorities via GSM-R communication. It still operates on an infrastructure that is segmented into fixed blocks. Wendler [81] and Geiß [28] show that adapting the size of these fixed blocks increases the capacity of a line. ETCS level 3 further requires that trains report to the train control centre via GSM-R which infrastructure they have safely left. This provides the possibility to create a virtual block around a train. If GSM-R communication fails, the train control centre cannot reassign the infrastructure to another train and line capacity decreases.

As with many new developments, it is still difficult to estimate the impact of ERTMS. Will it achieve at least the same level of safety as traditional train control? Will it allow at least the same performance (speed, headway)?

Perspective In order to estimate the impact of GSM-R communication on the line capacity we introduced a stochastic communication model.We extended this model in order to emulate track and train behaviour and conducted simulations of a railway system with a simple track layout by means of two modelling approaches, and compared the results.We found that the results of this newly introduced stochastic model are comparable to those obtained by applying a sophisticated modelling tool. As the tool RUT is being used in practice, we are confident that the models we produced are also close to reality.

The simulation of ETCS level 3 indicates that GSM-R communication failures do not have a severe impact on the capacity of a line. Thus, the deterministic modelling approach of model A (which does not take into account GSM-R communication errors) may be appropriate. These results can even be extended for railway systems equipped with GSM-R communication as it occurs in ETCS level 2, as long as the braking phase on the track determines the minimum headway time; this holds for traditional, same-size blocks and for the case where blocks are smaller near stations (but not if one also enlarges blocks maximally on open track).

We could verify the appropriateness of the deterministic modelling approach with regards to the GSM-R communication by comparing it to a stochastic modelling approach. We propose, as future work, to evaluate similarly other aspects of railway operation which are represented deterministically for simplicity: are those simplifications also close enough to the (stochastic) real behaviour?

13 Further Model-Based Testing Case Studies

Participants: Petur Olsen (AAU), Johan Uijen, Carsten Rütz, Julien Schmaltz (ESI/RU), Jan Tretmans (ESI), Frits Vaandrager (ESI/RU).

Context Using the results from Quasimodo on model-based testing (MBT), we performed a couple of other (smaller) model-based testing case studies:

1. testing key states of automated trust anchor updating (RFC 5011) in Autotrust [66];
2. testing a printer controller at Océ: control part;
3. testing a printer controller at Océ: data part [61].

13.1 Testing automated trust anchor updating in Autotrust

Contribution The usability of Timed Model-Based Testing with UPPAAL-TRON has been investigated in a case study: conformance testing of the implementation Autotrust with Automatic Trust Anchor Updating, a protocol to help securing DNS.

The conclusion of this case study is that Timed Model-Based Testing with UPPAAL-TRON is indeed usable for testing timing requirements in such a protocol. Therefore, model-based testing can be considered as a promising technique that is applicable to timed testing of protocol implementations. Compared to manual testing, the technique adds the ease of automated test generation, which increases test coverage in many cases.

Further research concerning this case study is necessary with respect to inclusion of more concurrency in testing key states by separating key events from immediate calls to Autotrust. This would increase interleaving of key events from different keys. Furthermore, testing could be extended to the whole RFC, including more detailed tests. Moreover, an experiment on latency caused by the developed adapter might reveal test coverage problems. Analysing timing coverage of tests a posteriori could show if varying ranges of input choices with respect to timing were covered with the executed tests. More generally, investigating timing coverage and latency of UPPAAL-TRON should be part of further research.

Models for the timing aspects of the protocol have been developed specifically for model-based testing. Reuse of already existing models of the protocol, constructed for model-checking, was shown to be infeasible. Discovering rules, techniques, or heuristics for developing models that can be (re)used for both model-based testing and model-checking could make model-checking as well as (timed) model-based testing even more usable and profitable.

Finally, it was shown that the implementation Autotrust behaves according to the tested part of its specification.

13.2 Testing a printer controller: control part

Contribution Océ is a company producing high-end, professional printers. The controller in these printers basically has two tasks: (*i*) handling the printing and waiting queues; and (*ii*) processing an input job description and sending the corresponding commands to the printing hardware. The part of the controller handling the printing and waiting queues can be seen as a reactive system, which continually monitors job descriptions, sends them through the job processor to the printing hardware, and allows the user to perform actions on these queues via a user interface, for instance to cancel a job. The second task will be further described in Section 13.3.

The handler for printing and waiting queues was tested using state-based models. Modelling the complete controller was too much, so a small but still complex part of the software was selected for the model. A number of different approaches were tried, but it appeared that making parallel executing models, each covering a small part of the SUT gave best results.

For test generation from the models several tools were applied: JTORX, GAST, Conformiq, TORXAKIS, and directly programming the model in the language PYTHON. Test execution was performed by connecting these tools to Océ's test execution framework. It turned out that both dealing with data and with nondeterminism were important which in this combination are only supported by GAST and TORXAKIS. Since the Océ's test execution environment uses PYTHON also a direct implementation of the model in PYTHON was developed. This last one turned out to be best accepted by Océ engineers.

Benefits. A few new software problems were pointed out using MBT, the connection between MBT and the existing test execution framework proved to be feasible, academic tools like GAST and TORXAKIS were shown to operate in an industrial environment, and some shortcomings of commercial tools were pointed out.

Shortcomings. We learnt that making models of real industrial systems is not trivial and needs more support. With respect to the MBT tools, GAST is able to deal with the size of systems at Océ, but lacks structuring of models. TORXAKIS can in principle deal with the Océ problems, but for longer tests it has scalability problems (state-space explosion).

Dealing with concurrency (parallelism) and non-determinism of behaviour is crucial for the Océ case and gives TORXAKIS and GAST a clear benefit over commercial tools such as Conformiq. The latter failed because it cannot deal with concurrency and nondeterminism, as discussion with its developers confirmed. JTORX has shortcomings when it comes to modelling data aspects. All, JTORX, GAST, and TORXAKIS have severe shortcomings in usability: writing down the models in their respective input languages is much too cumbersome. That's why also the PYTHON solution was tried.

13.3 Testing a printer controller: data part

Contribution The data part, i.e., the part of the controller which processes input job descriptions and sends commands to the hardware, does not operate reactively. It accepts one job description at a time, and produces outputs for that job. Such a system can be seen in its abstract form as a simple, stateless function, accepting a set of input parameter values and returning a

set of output parameter values. Input parameters are specific settings for a print job (number of pages, stapling, duplex/simplex, etc.), and the output is the description, in terms of output parameters of the actually printed job. The dependencies between inputs and outputs are not trivial, and as the number of input parameters is over 100 and the number of output parameters is more than 40, the size of the system makes testing a difficult task.

The controllers were modelled as a set of boolean formulas. Each formula is called a clause and is an expression, in the form of an implication, for an expected output value. To manage complexities of the models, we employed a trick for handling dependencies, by using some output values from the system under test to verify other output values. To avoid circular dependencies, the clauses were arranged in a hierarchy, where each clause depended on the outputs of its children. This modelling trick enabled us to model and test complex systems, using relatively simple models.

Pairwise testing was used for test case generation. This manages the number of test cases for complex systems.

The model-based testing approach has proved promising in improving the testing process and the quality of test cases. It resulted in increased maintainability and gives better understanding of test cases and their produced output. Using pairwise testing resulted in measurable coverage, with a test set smaller than the manually created test set.

The case study shows the importance of dealing efficiently with data transformations, and testing transformational systems with test suites that do not increase exponentially with the size of their input domains.

References

[1] MATLAB. `http://www.mathworks.com/`.

[2] Special issue on embedded systems. IEEE Computer Science, 2000.

[3] JTorX webpage, August 2009. http://fmt.ewi.utwente.nl/tools/jtorx/.

[4] mCRL2 toolkit webpage, September 2009. http://www.mcrl2.org/.

[5] María Alpuente, Byron Cook, and Christophe Joubert, editors. *Formal Methods for Industrial Critical Systems, 14th International Workshop, FMICS 2009, Eindhoven, The Netherlands, November 2-3, 2009. Proceedings*, volume 5825 of *Lecture Notes in Computer Science*. Springer, 2009.

[6] Tobias Amnell, Elena Fersman, Leonid Mokrushin, Paul Pettersson, and Wang Yi. TIMES – a tool for modelling and implementation of embedded systems. In *TACAS '02: Proceedings of the 8th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 460–464, London, UK, 2002. Springer-Verlag.

[7] J. Bauer. *Analysis of Communication Topologies by Partner Abstraction*. PhD thesis, Universität des Saarlandes, Saarbrücken, Germany, 2006.

[8] 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.

[9] G. Behrmann, A. David, and K.G. Larsen. A tutorial on Uppaal. In M. Bernardo and F. Corradini, editors, *Formal Methods for the Design of Real-Time Systems, International School on Formal Methods for the Design of Computer, Communication and Software Systems, SFM-RT 2004, Bertinoro, Italy, September 13-18, 2004, Revised Lectures*, volume 3185 of *Lecture Notes in Computer Science*, pages 200–236. Springer, 2004.

[10] 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, 16th International Conference (TACAS 2010)*, volume 6015 of *LNCS*, pages 266–270. Springer, March 2010.

[11] 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.

[12] Henrik Bohnenkamp and Axel Belinfante. Timed testing with torx. In John Fitzgerald, Ian Hayes, and Andrzej Tarlecki, editors, *FM 2005: Formal Methods*, volume 3582 of

Lecture Notes in Computer Science, pages 596–596. Springer Berlin / Heidelberg, 2005. 10.1007/11526841-13.

[13] Henrik Bohnenkamp, Joost-Pieter Katoen, Kai Mittermüller, Holger Hermanns, Julien Schmaltz, Faranak Heydarian, Frank Cassez, and Haidi Yue. Deliverable 5.5: Case studies: models. Deliverable, ICT-FP7-STREP-214755 / QUASIMODO, 2009.

[14] Bos, van Kuppeveld, Verhoef, and Xing. Design of safe real-time embedded systems in uppaal, 2010.

[15] E. Brinksma and A. Mader. On verification modelling of embedded systems. Technical Report TR-CTIT-04-03, Centre for Telematics and Information Technology, Univ. of Twente, The Netherlands, January 2004.

[16] G. Bruns and M.G. Staskauskas. Applying formal methods to a protocol standard and its implementations. In *Proceedings International Symposium on Software Engineering for Parallel and Distributed Systems (PDSE 1998),* 20-21 April, 1998, Kyoto, Japan, pages 198–205. IEEE Computer Society, 1998.

[17] BSI. Advanced security mechanisms for machine readable travel documents - extended access control (EAC) - version 1.11. Technical Report TR-03110, German Federal Office for Information Security (BSI), Bonn, Germany, 2008.

[18] Alan Burns. Preemptive priority based scheduling: An appropriate engineering approach. In *Principles of Real-Time Systems*, pages 225–248. Prentice Hall, 1994.

[19] F. Cassez, A. David, E. Fleury, K.G. Larsen, and D. Lime. Efficient on-the-fly algorithms for the analysis of timed games. In Martín Abadi and Luca de Alfaro, editors, *CONCUR 2005 - Concurrency Theory, 16th International Conference, CONCUR 2005, San Francisco, CA, USA, August 23-26, 2005, Proceedings*, volume 3653 of *Lecture Notes in Computer Science*, pages 66–80. Springer, 2005.

[20] Samarjit Chakraborty, Simon Künzli, and Lothar Thiele. A general framework for analysing system properties in platform-based embedded system designs. In *In DATE*, 2003.

[21] D. Chkliaev, J. Hooman, and E. de Vink. Verification and improvement of the sliding window protocol. In *Proceedings TACAS'03*, pages 113–127. Lecture Notes in Computer Science 2619, Springer-Verlag, 2003.

[22] Søren Christensen, Lars Kristensen, and Thomas Mailund. A Sweep-Line method for state space exploration. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 450–464. 2001.

[23] E. M. Clarke, O. Grumberg, H. Hiraishi, S. Jha, D. E. Long, K. L. McMillan, and L. A. Ness. Verification of the Futurebus+ cache coherence protocol. In *Proc. CHDL*, pages 15–30, 1993.

[24] Alexandre David, Jacob Illum, Kim G. Larsen, and Arne Skou. *Model-Based Design for Embedded Systems*, chapter Model-Based Framework for Schedulability Analysis Using UPPAAL 4.1, pages 93–119. CRC Press, 2010.

[25] 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.

[26] Elena Fersman. *A generic approach to schedulability analysis of real-time systems*. Acta Universitatis Upsaliensis, 2003.

[27] L. Frantzen, J. Tretmans, and T.A.C. Willemse. Test generation based on symbolic specifications. In *Proceedings of the 4th International Workshop on Formal Approaches to Software Testing (FATES '04)*, volume 3395 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2005.

[28] Günter Geiß. CIR-ELKE-Pilot Offenburg–Basel. *Signal + Draht*, 94(1+2):39–41, 2002.

[29] A.H. Ghamarian, M.C.W. Geilen, S. Stuijk, T. Basten, A.J.M. Moonen, M.J.G. Bekooij, B.D. Theelen, and M.R. Mousavi. Throughput analysis of synchronous data flow graphs. In *Application of Concurrency to System Design, 6th International Conference, ACSD 2006*, pages 25–34. IEEE CS Press, June 2006.

[30] Hernán Baró Graf, Holger Hermanns, Juhi Kulshrestha, Jens Peter, Anjo Vahldiek, and Aravind Vasudevan. A verified dependable wireless safety critical hard Real-Time design. In *12th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM) (IEEE WoWMoM 2011)*, Lucca, Italy, June 2011.

[31] Alexander Graf-Brill. Testing the hydac case. Bachelor thesis, Saarland University, 2011.

[32] Jan Friso Groote et al. The mCRL2 toolset. In *Proc. International Workshop on Advanced Software Development Tools and Techniques (WASDeTT 2008)*, 2008.

[33] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Trans. Inf. Theory*, 46(2), March 2000.

[34] Arne Hamann, Marek Jersak, Kai Richter, and Rolf Ernst. Design space exploration and system optimization with symta/s– symbolic timing analysis for systems. In *IEEE Real-Time Systems Symposium*, pages 469–478, 2004.

[35] M. González Harbour, J. J. Gutiérrez García, J. C. Palencia Gutiérrez, and J. M. Drake Moyano. Mast: Modeling and analysis suite for real time applications. In *In 13th Euromicro Conference on Real-Time Systems*, page 125, 2001.

[36] Arnd Hartmanns and Holger Hermanns. A modest approach to checking probabilistic timed automata. In *QEST*. IEEE Computer Society, September 2009.

[37] Holger Hermanns, Poul Hougaard, Teun van Kuppeveld, Kai Sven Mittermüller, Jan Storbank Pedersen, Marcel Verhoef, and Ivo van Vessem. Deliverable 5.2: Preliminary description of case studies. Deliverable, ICT-FP7-STREP-214755 / QUASIMODO, 2008.

[38] Robert M. Hierons, Jonathan P. Bowen, and Mark Harman, editors. *Formal Methods and Testing, An Outcome of the FORTEST Network, Revised Selected Papers*, volume 4949 of *Lecture Notes in Computer Science*. Springer, 2008.

[39] G.J. Holzmann. *The SPIN Model Checker: Primer and Reference Manual*. Addison Wesley, 2004.

[40] ICAO. Doc 9303 - machine readable travel documents - part 1-2. Technical report, International Civil Aviation Organization, 2006. Sixth edition.

[41] 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.

[42] Jacob Illum, Kim G. Larsen, Marius Mikucionis, and Steen Palm. Model-based approach for schedulability analysis. Intern Report.

[43] ISO, Geneve. *Information Technology, Open Systems Interconnection, Conformance Testing Methodology and Framework. International Standard IS-9646*, 1991. Also CTITT X.290–X.294.

[44] K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use*. EATCS Monographs on Theoretical Computer Science. Springer, 1992.

[45] Kurt Jensen, Lars Michael Kristensen, and Lisa Wells. Coloured petri nets and cpn tools for modelling and validation of concurrent systems. *Int. J. Softw. Tools Technol. Transf.*, 9:213–254, May 2007.

[46] Malte Kampschulte. Evaluation of radio models for the analysis of a gossiping mac protocol. Bachelor thesis, RWTH Aachen University, 2010.

[47] M. Kwiatkowska, G. Norman, and D. Parker. Prism: Probabilistic model checking for performance and reliability analysis. *ACM SIGMETRICS Performance Evaluation Review*, 36(4):40–45, 2009.

[48] M. Kwiatkowska, G. Norman, D. Parker, and J. Sproston. Performance analysis of probabilistic timed automata using digital clocks. In K. Larsen and P. Niebert, editors, *Proc. Formal Modeling and Analysis of Timed Systems (FORMATS'03)*, volume 2791 of *LNCS*, pages 105–120. Springer-Verlag, 2003.

[49] Kai Lampka, Simon Perathoner, and Lothar Thiele. Analytic real-time analysis and timed automata: a hybrid method for analyzing embedded real-time systems. In *Proceedings of the seventh ACM international conference on Embedded software*, EMSOFT '09, pages 107–116, New York, NY, USA, 2009. ACM.

[50] I. van Langevelde, J.M.T. Romijn, and N. Goga. Founding FireWire bridges through Promela prototyping. In 8^{th} *International Workshop on Formal Methods for Parallel Programming: Theory and Applications (FMPPTA)*. IEEE Computer Society Press, April 2003.

[51] K. G. Larsen, P. Pettersson, and W. Yi. UPPAAL in a Nutshell. *Int. Journal on Software Tools for Technology Transfer*, 1(1–2):134–152, October 1997.

[52] K.G. Larsen, B. Nielsen, and J. Tretmans, editors. *Quasimodo*. Springer Verlag, 2011. to appear.

[53] Kim G. Larsen, Marius Mikučionis, Brian Nielsen, Steen Palm, and Jacob I. Rasmussen. Model-based approach for schedulability analysis. Quasimodo Deliverable D5.7b. Confidential Document.

[54] Edward A. Lee. Embedded software. *Advances in Computers*, 56:56–97, 2002.

[55] Marius Mikucionis, Kim G. Larsen, and Brian Nielsen. *Handbook on Quantitative Model-Driven Development for Embedded Systems*, chapter Schedulability Analysis using Uppaal: The Herschel/Planck Software Case. submitted.

[56] Marius Mikučionis, Kim Guldstrand Larsen, Jacob Illum Rasmussen, Brian Nielsen, Arne Skou, Steen Ulrik Palm, Jan Storbank Pedersen, and Poul Hougaard. Schedulability analysis using uppaal: Herschel-planck case study. In Tiziana Margaria, editor, *ISoLA 2010 – 4th International Symposium On Leveraging Applications of Formal Methods, Verification and Validation*, volume Lecture Notes in Computer Science. Springer, October 2010.

[57] W. Mostowski, E. Poll, J. Schmaltz, J. Tretmans, and R. Wichers Schreur. Model-based testing of electronic passports. In *Proceedings of the 14th International Workshop on Formal Methods for Industrial Critical Systems (FMICS '09)*, volume 5825 of *Lecture Notes in Computer Science*, pages 207–209. Springer, 2009.

[58] B. Nielsen and J. Tretmans. *Model Based Testing*. In Larsen et al. [52], 2011. to appear.

[59] Homepage octopus project, May 2011. http://www.esi.nl/short/octopus.

[60] Joël Ouaknine and Frits W. Vaandrager, editors. *Formal Modeling and Analysis of Timed Systems, 7th International Conference, FORMATS 2009, Budapest, Hungary, September 14-16, 2009. Proceedings*, volume 5813 of *Lecture Notes in Computer Science*. Springer, 2009.

[61] J. Foederer P. Olsen and and J. Tretmans. Model-based approach for schedulability analysis. Submitted, 2011.

[62] Steen Palm. Herschel-Planck ACC ASW: sizing, timing and schedulability analysis. Technical report, Terma A/S, 2006.

[63] Jan Storbank Pedersen. Herschel/planck acc asw interface control document. Technical report, Terma A/S, June 2008.

[64] J.M.T. Romijn. Improving the quality of protocol standards: Correcting IEEE 1394.1 FireWire net update. *Nieuwsbrief van de Nederlandse Vereniging voor Theoretische Informatica*, 8:23–30, 2004. Available at `http://www.win.tue.nl/oas/index.html?iqps/`.

[65] Paul E. Rook. Controlling software projects. *IEE Software Engineering Journal*, 1(1):7–16, January 1986.

[66] C. Rütz and J. Schmaltz. An experience report on an industrial case-study about timed model-based testing with uppaal-tron. In *Proceedings of the* 7^{th} *Int. Workshop on Advances in Model-Based Testing (A-MOST'11), IEEE CS*, 2011.

[67] M. Stoelinga. Fun with FireWire: A comparative study of formal verification methods applied to the IEEE 1394 root contention protocol. *Formal Aspects of Computing Journal*, 14(3):328–337, 2003.

[68] S. Stuijk, M.C.W. Geilen, and T. Basten. SDF3: SDF For Free. In *Application of Concurrency to System Design, 6th International Conference, ACSD 2006*, pages 276–278. IEEE CS Press, June 2006.

[69] M. Sytema, M.I.A. Stoelinga, A.F.E. Belinfante, and L. Marinelli. Experiences with formal engineering: Model-based specication, implementation and testing of a software bus at neopost. In *Proceedings of the 16th International Workshop on Formal Methods for Industrial Critical Systems (FMICS'11)*, LNCS, 2011.

[70] Andrew S. Tanenbaum. *Computer Networks*. Prentice Hall International, Inc., 1989. Second Edition.

[71] Terma A/S. Herschel-Planck ACMS ACC ASW requirements specification. Technical report, Terma A/S, Issue 4/0.

[72] Terma A/S. Software timing and sizing budgets. Technical report, Terma A/S, Issue 9.

[73] Gregor Theeg and Béla Vincze. European train protection systems compared. *Signal + Draht*, 99(7+8):35–40, 2007.

[74] Lothar Thiele, Iuliana Bacivarov, Wolfgang Haid, and Kai Huang. Mapping applications to tiled multiprocessor embedded systems. In *Proceedings of the Seventh International Conference on Application of Concurrency to System Design*, pages 29–40, Washington, DC, USA, 2007. IEEE Computer Society.

[75] J. Tretmans. Model based testing with labelled transition systems. In *Formal Methods and Testing*, volume 4949 of *Lecture Notes in Computer Science*, pages 1–38. Springer, 2008.

[76] J. Tretmans and H. Brinksma. TorX: Automated model based testing. In *Proceedings of the 1st European Conference on Model-Driven Software Engineering*. Imbuss, Möhrendorf, Germany, 2003.

[77] Uppaal pro - uppaal for probabilistic timed automata, May 2011. http://www.cs.aau.dk/ arild/uppaal-probabilistic/.

[78] Mark Utting and Bruno Legeard. *Practical Model-Based Testing - A Tools Approach*. Morgan Kaufmann, 2007.

[79] F.W. Vaandrager and A.L. de Groot. Analysis of a biphase mark protocol with Uppaal and PVS. *Formal Aspects of Computing Journal*, 18(4):433–458, December 2006.

[80] M. Verhoef, A. Belinfante, F. Zhu, and J. Tretmans. *Model Based Testing of a WSN node*. In Larsen et al. [52], 2011. to appear.

[81] Ekkehard Wendler. Weiterentwicklung der Sperrzeitentreppe für moderne Signalsysteme. *Signal + Draht*, 87(7+8):268–273, 1995.

[82] Samuel Williams, Andrew Waterman, and David Patterson. Roofline: an insightful visual performance model for multicore architectures. *Commun. ACM*, 52:65–76, April 2009.

[83] Peter Winter. ETCS: the european train control system: Requirements and reality. *RTR*, 51(2+3):29–37, 2002.

[84] Wayne Wolf, editor. *EMSOFT 2005, September 18-22, 2005, Jersey City, NJ, USA, 5th ACM International Conference On Embedded Software, Proceedings*. ACM, 2005.

[85] Haidi Yue, Henrik Bohnenkamp, Malte Kampschulte, and Joost-Pieter Katoen. Analysing and improving energy efficiency of distributed slotted aloha. In *The 11th International Conference on Next Generation Wired/Wireless Advanced Networking (NEW2AN)*, LNCS. Springer-Verlag, 2011.

[86] Haidi Yue, Henrik Bohnenkamp, and Joost-Pieter Katoen. Analyzing energy consumption in a gossiping MAC protocol. In *Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance (MMB/DFT)*, volume 5987 of *LNCS*, pages 107–119. Springer-Verlag, 2010.

[87] Feng Zhu. Testing timed systems in simulated time with uppaal-tron: An industrial case study. Bachelor thesis, Institute for Computing and Information Sciences, Radboud University, 2010.