# Towards an Automated Reverse Engineering of Design Models from Trace Recordings

Andreas Sailer
Laboratory of Safe and Secure Systems (LaS$^3$)
Ostbayerische Technische Hochschule (OTH) Regensburg
Seybothstraße 2, 93053 Regensburg, Germany
andreas.sailer@oth-regensburg.de

**Abstract:** Recording traces from an embedded system is a common procedure in the process of developing real-time systems. While this technique is generally used for debugging and performance analysis, the recordings contain much more information. These pieces of information however can be useful to generate or maintain a system model in an automated manner. Therefore, the goal is to develop an approach that creates a model of a real-time system, especially of its observed temporal behaviour, from events logged in a trace recording. This research aims to extract, analyse and deduce information about the system under observation from a limited view of the proceedings in the system. Also as a consequence of this intention a detailed specification on events necessary for specific use cases emerges. To achieve all this, an iterative approach consisting of three consecutive steps is proposed. At first the necessary algorithms will be developed on a model-based foundation utilising a discrete event simulation. In a next step, the implementation will be aligned to take also specifics of the execution on real hardware into account. Finally, the approach can be evaluated on hardware with examples from real life.

## 1 Motivation

Contrary to popular belief, the development of software for embedded and non-embedded systems are fundamentally different [GLT03]. Processes and technologies have to address additional requirements such as timing constraints. This means the correctness of such a system depends not only on the functional correctness but also on its temporal accuracy. Consequently, traditional debuggers cannot be used to observe the runtime behaviour of software since they require halting a system's execution, which would result in a significant alteration of the timing behaviour. Instead, tracing is used during the development of embedded software systems. 'Concretely, trace recording implies detection and storage of relevant events during runtime, for later off-line analysis' [KWK10, p.1].
Nowadays, execution traces are used in most parts of the industry for dynamic timing analysis [WEE$^+$08]. From the recorded events in a trace, the event-to-event execution time is calculated. That way the minimal and maximal observed execution times are determined, which help to estimate the actual execution time bounds. Metrics that are calculated this way are for example the Execution Time (ET) of functions [SSD$^+$13], and the Response Time (RT) [QHE12] or the mean Normalised Blocking Time (mNBT) [AMS$^+$13] of tasks.

However, much more information can be extracted from execution traces. This is due to the fact that events from different abstraction levels of a system can be recorded [Hus07]. On the one hand a trace can contain events from the system level, where system events such as task switches or the creation and termination of tasks can be registered as well as calls to functions. On the other hand also proceedings on a process level can be monitored. Those are for example events because of accesses to resources such as data signals or semaphore queues. Therefore, the main goal of this research is to define an approach how a temporally accurate design model of a real-time system can be automatically created based on trace recordings.
Such a generated model has many advantages over a manually maintained one and can be used for multiple purposes. In the simplest way it can tremendously help to understand the behaviour of a software system in general [HLL05]. Although model-driven software development (MDSD) is standard in the embedded industry, it is also common to modify the generated code afterwards by hand. Thus, an approach, which can easily keep a model up-to-date, would be a huge benefit not only for simple documentation purposes during development but also for the verification of software models [HA05] or the maintenance of legacy systems [TVD12]. But the currently probably most important reason for such a generated model is simulation-based timing analysis [KKM11]. With the tendency towards a commitment to multi-core technology, software that has run on a single core for years has now to be allocated on different cores and optimised regarding certain metrics. Thus, the use of a discrete event simulation helps to handle the resulting complexity by imitating the key characteristics of a system on the basis of a model. That way it is possible to estimate the effect of design decisions without the need for physical access to the system.
Although trace recordings can give a valuable in-depth look into a system's execution behaviour, collecting all this data is expensive. In the area of embedded systems, where only very limited resources are available, this might result in an unintended alteration of the system behaviour [Gai86] or an unmanageably large growth in data traffic [RS13]. Due to the fact that not every event contributes to the model extraction in the same way, an additional motivation is to develop a detailed specification on all the events necessary for specific use cases. That way a work flow is targeted, which can make a precise statement about the needed quality of a trace recording in order to achieve specific goals.
The remainder of this paper is organised in the following way: In the next section, we discuss the presented problem in more detail. After that, an overview of related research and state-of-the-art is given. In Section 4, we present the approach to implement the aforementioned research goals. We conclude with an outline of the current state of research and the achieved results so far.

## 2 Problem

As mentioned above, the main goal of this research is to define an approach that creates a design model of an embedded system, which is coherent and consistent in timing, based on the information contained in execution traces. Another prerequisite is that this model

has to be compliant to AUTOSAR [aut13], the Automotive Open System Architecture, in order to guarantee applicability in the industry for this research.
The mentioned model must therefore cover elements from hardware to software as well as operating system characteristics. The former has to include an abstract description of the used processing unit with the number of cores and their frequencies.
Main focus of the research however will be on the software model. This must describe how the system is composed in the matter of tasks, functions and used resources. Each of these components have again a number of characteristics that have to be determined. For tasks these are, for example, the priority, preemptability, maximum number of concurrent activations, period and offset. In case of functions, especially the exact modelling of accesses to resources provided by the operating system, such as data signals or semaphores, is of importance. That way the dependency between functions in the system is comprehensible, which is crucial for deploying software on multiple processing cores. Finally, the operating system has to be described by determining the used scheduling algorithm and modelling its characteristics.
Based on these requirements this work is organised around the following four research problems:

1. **Method for model extraction**
   First of all, a method for automatically building an AUTOSAR compliant model based on the observed behaviour recorded in a trace has to be developed. Such a general applicable method itself already states a huge innovation for the industry due to the fact that currently only very limited research has been done in the area of reverse engineering for embedded systems [KKM11].
   The method can be developed based on different technical approaches. On the one hand it can be implemented purely algorithmically, where all known is explicitly programmed. On the other hand knowledge can also be learned inductively.
   Therefore, a method has to be developed, which not only solves this problem best but which also performs best regarding accompanied challenges such as creating the most confidence in the resulting model or providing the best robustness regarding abnormalities in the trace.

2. **A heuristic for tracing**
   The information contained in a trace depends on many facts such as, for example, the length of time recorded in real time. This means that recording for a longer time period can result in more details about the system. Similarly, the levels of abstractions considered as well as the granularity of the system proceedings observed are crucial. Hence, the quality of a trace can vary greatly. However, the accuracy of the model is again dependent on the quality of the measured data.
   Unfortunately, recording every detail of the proceedings in a system all the time is also not possible. This is due to the fact that every measurement affects the timing behaviour of the system in a little, known as the so called probe effect [Gai86]. Thus, extensive recording can lead to erroneous runtime behaviour of a system. Furthermore, tracing has to cope with the limited system resources and connectivity available for embedded systems [TVD12]. Those lead to the fact that not enough trace data can be stored and transferred to obtain all arising events.

For those reasons, we think a heuristic for tracing is necessary, which tackles the question how to get the most usable information into a trace recording for building a coherent and consistent model regarding a specific use case.

3. **Model refinement**
   Another fact regarding tracing is that each recording only contains information about the system's behaviour that has actually been observed. So each time just a fraction of the complete system behaviour is received [HA05]. The quality of the targeted model however is dependent on those observations. If there is no recording of a certain component of the system, the approach and consequently the model cannot get knowledge about this. Yet, the more detailed and complete the created model is, the more valuable and significant it gets for further purposes.
   Hence, additional thoughts have to be given on how new, informative recordings can be explored in order to gradually refine the quality of an existing model.

4. **Quality of the approach**
   Finally, the quality of the approach has to be assessed in a convincing and confident way. One of the goals of the reverse engineering is to get a model that can be used for timing simulation and further analysis. As a consequence, the resulting model is only of value, if and only if a clear statement about its quality can be made. Quality in this case means, to measure to what extent the model behaviour conforms to that of the actual system. When considering the targeted field of application for such an automatic reverse engineering approach, for example the automotive industry, where functionality is bought from suppliers in form of black boxes, it gets obvious that trace recordings are the only available information sources about the actual system. Hence, in order to get such a quality measure, a precise way to compare a model with trace recordings has to be defined.

## 3 Related Work

In general 'surprisingly little research in reverse engineering targets embedded systems' [KKM11, p.8]. In this work a broad overview on the research activities in the area of software reverse engineering for embedded systems is given.
One of the latest publications about architecture recovery mentioned there is [MW10]. Marburger and Westfechtel present an approach to analyse the structure as well as the behaviour of a system. For that, a state machine is extracted from the available source code. Our research however seeks applicability in the automotive industry. There the availability of source code cannot be ensured, since it is common to purchase functionality from external suppliers, which then have to be considered as black boxes.
Huselius [HA05, AHNW06, HAHP06, Hus07] provides probably the most related work. In the context of his Ph.D. thesis [Hus07] he researched reverse engineering of legacy real-time systems in order to develop an automated approach based on execution-time recording. For this, he creates a probabilistic state-machine model expressed in the ART-ML language [HA05]. Nevertheless, his work provides some shortcomings, which are

crucial for our purpose to use the model for simulation-based timing analysis. ART-ML, for example, provides a very high-level view on the system. This means resource accesses are associated with tasks instead of functions. To be able to allocate software to different cores and to optimise it regarding certain metrics however, the knowledge of functions and their execution order is significant. He moreover makes certain assumptions for his research, such as the knowledge of the priority or the triggering method of the tasks, which are not acceptable for our intended applicability.
One of the latest research activities in this area comes from Ciccozzi [CSCS13]. There, a round-trip approach for component-based embedded systems is proposed. The goal is to provide an automated support for deployment decisions at modelling level. It however focusses only on the determination of extra-functional properties and doesn't consider architectural or behavioural characteristics for back-propagation to the design model.
In addition to the aforementioned related works, which cover existing ambitions in the area of embedded systems, relevant approaches can also be found in the area of process mining. The goal there is to extract information from event logs in order to capture the underlying business process. An overview on the main issues around this topic can be found in [VdAW04]. As the name already implies, process mining focusses on the application of data mining techniques in order to achieve the desired goal. Attention however should be paid to the fact, that models for real-time systems have to satisfy much stricter requirements then business processes.
The confidence in the quality of the resulting model plays a crucial role. Therefore, also a vast variety of literature on model validation can be found. Balci proposes in [Bal90] different techniques for model validation in simulation studies. This work is amended by Sargent [Sar04]. Law presents in [Law09] general techniques for validating a simulation model. In the end, all of the above list just a set of statistical techniques, which can be used for general validation. Huselius [HKHP07, Hus07] developed a metric, which is used to evaluate the quality of models from embedded real-time systems by analysing the response time of the available tasks. Kraft discusses in [Kra10] a possibility for the validation of simulation models by comparing simulation traces with traces from the modelled system. Although different steps are proposed in this work, no particular properties are presented, but just a vague approach how to find those properties. Furthermore, most steps proposed target obviously a manual validation and are thus not feasible for an automatic approach.

## 4 Approach

In order to achieve the projected goals, this research is divided into three consecutive phases. However, the first and second phase are meant to be executed iteratively. For example in cases, where the evaluation at the end of a phase reveals insufficient quality, an iteration of the phase or even the previous one is necessary to improve the algorithms. In the following subsections, the individual phases are presented in detail.

### 1. Simulation-based Development & Validation Phase

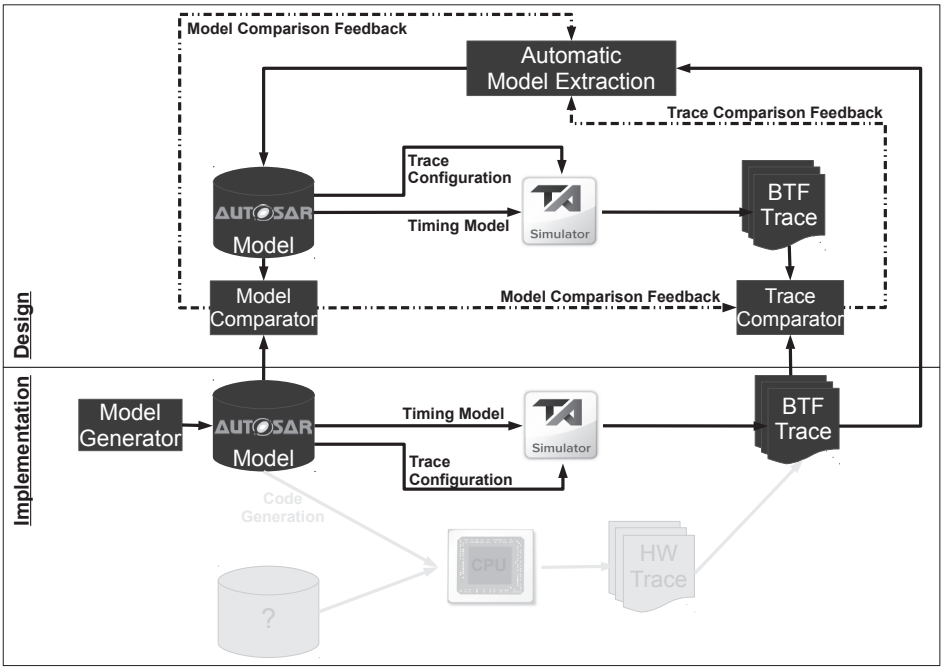


Figure 1: Simulation-based Development & Validation Phase

In the first phase as depicted in Figure 1, all necessary development is done utilizing a discrete event simulation. For this, the simulation tool from the Timing Architects Toolsuite [TA] is used. The advantage thereby is that trace recordings can be generated much faster and more comfortable than working with real hardware. The proposed approach starts with an implementation model, which is pictured in the lower part of the figure. Based on this AUTOSAR compliant model, the simulator creates trace recordings corresponding the configured granularity and length. That way the resulting information contained within a single trace file can be set electively. This functionality is designated to investigate the problem of specifying a heuristic for tracing.

The generated trace recordings serve then as input for the automatic model extraction. As the name already implies, this automatic model extraction builds then again an AUTOSAR compliant model based on the information contained in trace recordings. Thus, the quality of the resulting model can easily be determined by comparing it with the implementation model. That way the development of a method for model extraction can be done model-based, which makes it possible to make a clear statement about the quality of the resulting model and as a consequence also about the quality of the method itself.

However, later during the operation of the automatic reverse engineering approach, there might not be an implementation model available to compare to. As a consequence, a pos-

sibility to assess the created model with regards to trace recordings from the origin source has to be available. This is done by simulating the gained model and afterwards comparing the resulting trace with other trace recordings from the original source. This procedure makes it also possible to test the potential of supervised learning techniques in the automatic model extraction as well as in the trace comparison.
The development proceeds then until the automatic model extraction continually produces models of the desired quality. This is evaluated by using variations of major architectural system designs for the implementation model.

### 2. Operational Adoption & Validation Phase

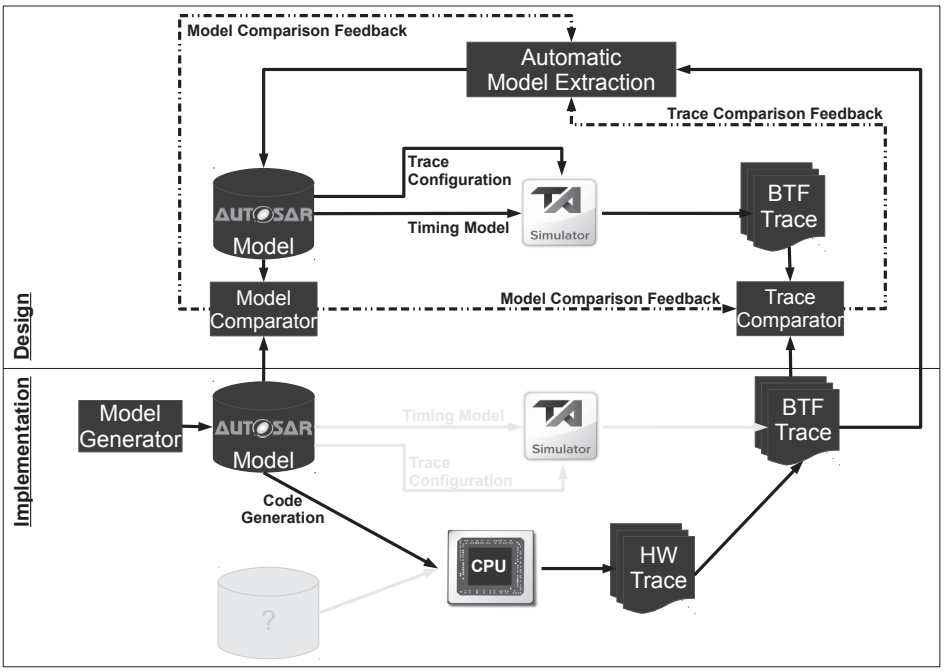


Figure 2: Operational Adoption & Validation Phase

In the second phase, the next step towards an operational use of the reverse engineering approach is taken. For that, the method for model extraction developed up to that point is validated using hardware traces instead of trace recordings generated by the discrete event simulation.
Starting point for this are once again the exact kind of models as used before with the simulation. Those however are now executed on real hardware by using code generation. The resulting hardware traces are then transformed into the trace format of the simulation, called BTF Trace Format. This process is just a simple mapping of elements from a binary trace format, which is used to compress the data within a hardware trace, to a more usable

trace format.
From here on, as depicted in Figure 2, the rest of the work flow remains the same. That way, the differences between the simulation and execution can be determined and the automatic model extraction adapted accordingly.
Since this approach is still model-based, the quality of the resulting model can be determined by comparing it with the implementation model. That way the automatic model extraction can again be improved until it continually produces models of the desired quality.

## 3. Operational Phase

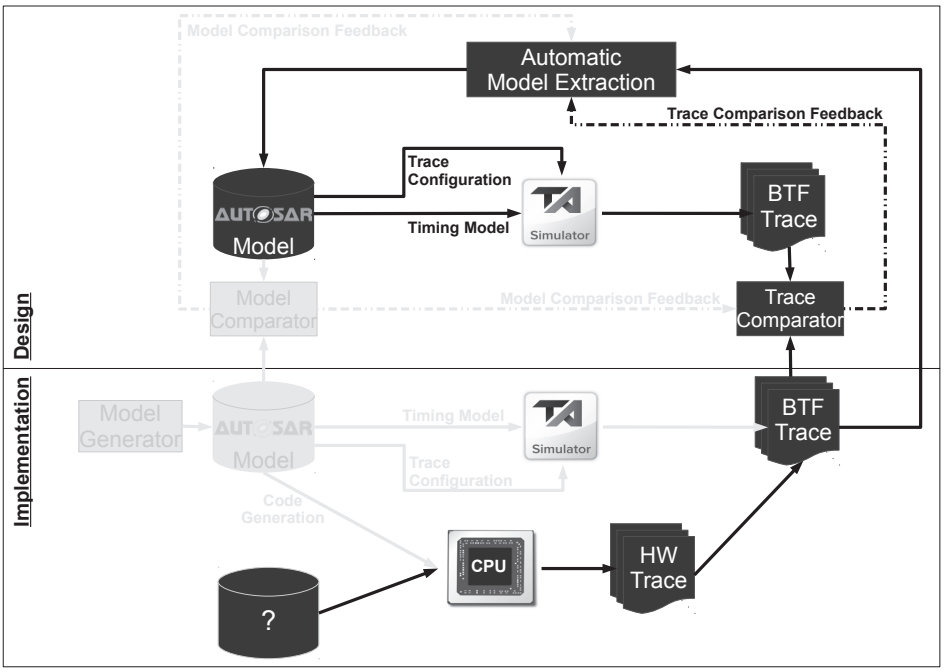


Figure 3: Operational Phase

Last but not least, the developed reverse engineering approach is ready for operation. While the starting point of the previous phases was each time an implementation model, this time it is not the case. Instead, the system has to be considered as a black box for different reasons. On the one hand, for example, the system might have not been developed model-based and thus there was never a model available. On the other hand the source code generated from a model might also have been altered manually afterwards without synchronizing those changes with the model.
As a consequence, all the information available about the system comes from hardware traces and there is no possibility to assess the resulting model with regards to an imple-

mentation model. However, as depicted in Figure 3, the work flow of the reverse engineering approach remains the same. So, since this approach already produced models of a desired quality before throughout the previous two phases, the user can be confident in the resulting model this time, too.
In addition to this fact, a statement about the quality of the resulting model is made by comparing it with other trace recordings from the original source. This is also done equivalently to the previous two phases by simulating the gained model and afterwards comparing the resulting trace with the other trace recordings from the origin source. That way the needed confidence from the user in the resulting model should be achieved.

## 5 Preliminary Results & Future Work

The preliminary results obtained so far demonstrate not only the feasibility of the proposed approach but also the necessity of research in this area. The following paragraphs give an overview on the current achievements and developments.
Currently, the transition between the first and second phase is in process. In a first step the work focused on extracting individual properties of a model one by one from trace recordings. For example, the used scheduling algorithm is determined from a trace by using constraint programming. At first, significant changes in the state of a task, such as task preemptions, are detected within the trace. Based on those a corresponding constraint describing the expected behaviour of a scheduling algorithm is created. In case of a fixed-priority scheduler, for each task preemption a constraint expressing that the priority of the preempting task is higher than that of the preempted one would be added to the constraint system of the represented scheduling algorithm. Finally, if the constraint system has a solution, this means that there was no fact militating against the scheduling algorithm under investigation.
Another example of determining a property of the system behaviour is the specification of the dynamic runtime of functions. For that, the individual runtimes of a function are calculated from the trace at first. Those values are then used for a goodness-of-fit test to examine, how well they fit a statistical distribution. Further impressions on the work of extracting individual properties of a model can be found in [SSD+13].
For further development on the reverse engineering approach, AUTOSAR as the targeted meta-model was replaced by AMALTHEA [ama13]. AMALTHEA is an open-source meta-model, which is not only compliant to AUTOSAR but also continuously covers all information during the development of software for embedded systems [SSDM13]. That way it was possible to gather more interesting or relevant pieces of information in the model.
Up to now a set of heuristic methods to determine the most critical parameters from trace recordings is available. The quality of the developed automatic modelling approach is ensured by a model comparison. For this, the Eclipse Modelling Framework (EMF) was used, since both, AUTOSAR and AMALTHEA, have a meta-model description available in EMF. Thus, a plain comparison of models was possible by comparing element by element.

Figure 4 provides sample results from such a comparison. For that chart, 20 random mod-

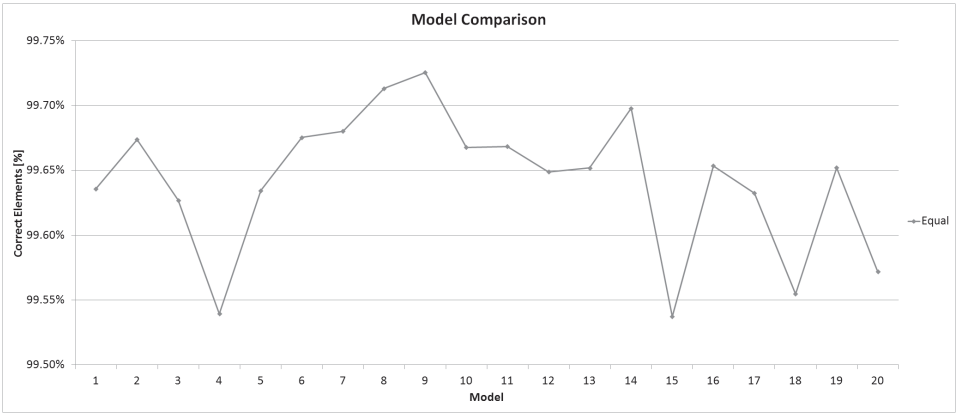


Figure 4: EMF Model Comparison

els were created. Each model represents a purely periodic task set without communication in a preemptive, fixed-priority scheduling environment. The size of the task sets varied from four to a maximum of 18 tasks. Within those, between 1374 and 2797 functions where called, where each function had a constant execution time. The length of the trace was chosen such that all tasks occurred at least once. The comparison of the reverse engineered model with the original model showed, that at least for such a simple example nearly every detail could be determined from the trace recordings. First validations of the current achievements on real hardware yielded promising results too.

In order to be also able to make a statement about the capability of the newly developed approach compared to existing ones, a benchmark is currently defined. Due to the fact that Huselius provides the most related work, the results published in [Hus07] will be considered for this.

Furthermore, one of the major tasks in the near future will be the comparison of characteristics from trace recordings to those of a model. In the course of this research different metrics will be examined and developed in order to define a comparison framework. Such metrics will be on the one hand performance measures of the system such as response times of tasks or execution times of functions, but on the other hand also abstract ones such as the pattern of task interferences. The former are analysed using once again statistical goodness-of-fit tests such as the two-sample Kolmogorov-Smirnov test. Based on the patterns, coherences between the individual metrics are examined by performing correlation, regression and association analyses. Since this is an obvious field of application for machine learning techniques such as supervised learning or pattern recognition, this activity represents a good starting point to also include those techniques in the automatic model extraction.

Before that however, an expressive evaluation of the automatic modelling approach must be defined in order to make a statement on the impact of those techniques on the model quality. In parallel to those activities also the work on validating the preliminary results on

real hardware will be intensified.

## 6 Acknowledgments

This work was partially funded by the ITEA2 project AMALTHEA (ITEA2-09013) through the German Ministry of Education and Research (BMBF) under the funding code 01IS11020F. I would like to thank my supervisors Jürgen Mottok and Gerald Lüttgen as well as Michael Deubzer for the many valuable discussions regarding this research.

## References

[AHNW06] Johan Andersson, Joel Huselius, Christer Norstrom, and Anders Wall. Extracting simulation models from complex embedded real-time systems. In *Software Engineering Advances, International Conference on*, pages 7–7. IEEE, 2006.

[ama13] http://www.amalthea-project.org, November 2013.

[AMS+13] Martin Alfranseder, Matthias Mucha, Stefan Schmidhuber, Andreas Sailer, Michael Niemetz, and Jürgen Mottok. A modified synchronization model for dead-lock free concurrent execution of strongly interacting task sets in embedded systems. In *Applied Electronics (AE), 2013 International Conference on*, pages 1–6. IEEE, 2013.

[aut13] http://www.autosar.org, November 2013.

[Bal90] Osman Balci. Guidelines for successful simulation studies. In *Simulation Conference, 1990. Proceedings., Winter*, pages 25–32. IEEE, 1990.

[CSCS13] Federico Ciccozzi, Mehrdad Saadatmand, Antonio Cicchetti, and Mikael Sjödin. An automated round-trip support towards deployment assessment in component-based embedded systems. In *Proceedings of the 16th International ACM Sigsoft symposium on Component-based software engineering*, pages 179–188. ACM, 2013.

[Gai86] Jason Gait. A probe effect in concurrent programs. *Software: Practice and Experience*, 16(3):225–233, 1986.

[GLT03] Bas Graaf, Marco Lormans, and Hans Toetenel. Embedded software engineering: the state of the practice. *Software, IEEE*, 20(6):61–69, 2003.

[HA05] Joel Huselius and Johan Andersson. Model synthesis for real-time systems. In *Software Maintenance and Reengineering, 2005. CSMR 2005. Ninth European Conference on*, pages 52–60. IEEE, 2005.

[HAHP06] Joel Huselius, Johan Andersson, Hans Hansson, and Sasikumar Punnekkat. Automatic generation and validation of models of legacy software. In *Embedded and Real-Time Computing Systems and Applications, 2006. Proceedings. 12th IEEE International Conference on*, pages 342–349. IEEE, 2006.

[HKHP07] Joel Huselius, Johan Kraft, Hans Hansson, and Sasikumar Punnekkat. Evaluating the quality of models extracted from embedded real-time software. In *Engineering of Computer-Based Systems, 2007. ECBS'07. 14th Annual IEEE International Conference and Workshops on the*, pages 577–585. IEEE, 2007.

[HLL05] Abdelwahab Hamou-Lhadj and Timothy C Lethbridge. Measuring various properties of execution traces to help build better trace analysis tools. In *Engineering of Complex Computer Systems, 2005. ICECCS 2005. Proceedings. 10th IEEE International Conference on*, pages 559–568. IEEE, 2005.

[Hus07] Joel Huselius. *Reverse Engineering of Legacy Real-Time Systems: An Automated Approach Based on Execution-Time Recording*. PhD thesis, Mälardalen University, 2007.

[KKM11] Holger M Kienle, Johan Kraft, and Hausi A Müller. Software Reverse Engineering in the Domain of Complex Embedded Systems. *InTech*, 2011.

[Kra10] Johan Kraft. *Enabling timing analysis of complex embedded software systems*. PhD thesis, Mälardalen University, 2010.

[KWK10] Johan Kraft, Anders Wall, and Holger Kienle. Trace recording for embedded systems: Lessons learned from five industrial projects. In *Runtime Verification*, pages 315–329. Springer, 2010.

[Law09] Averill M Law. How to build valid and credible simulation models. In *Simulation Conference (WSC), Proceedings of the 2009 Winter*, pages 24–33. IEEE, 2009.

[MW10] André Marburger and Bernhard Westfechtel. Graph-based structural analysis for telecommunication systems. In *Graph transformations and model-driven engineering*, pages 363–392. Springer, 2010.

[QHE12] Sophie Quinton, Matthias Hanke, and Rolf Ernst. Formal analysis of sporadic overload in real-time systems. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2012*, pages 515–520. IEEE, 2012.

[RS13] Salman Rafiq and Adriaan Schmidt. Systematic Modeling of Workflows in Trace-Based Software Debugging and Optimization. In *ICSEA 2013, The Eighth International Conference on Software Engineering Advances*, pages 241–248, 2013.

[Sar04] Robert G Sargent. Validation and verification of simulation models. In *Simulation Conference, 2004. Proceedings of the 2004 Winter*, volume 1. IEEE, 2004.

[SSD+13] Andreas Sailer, Stefan Schmidhuber, Michael Deubzer, Martin Alfranseder, Matthias Mucha, and Jürgen Mottok. Optimizing the task allocation step for multi-core processors within AUTOSAR. In *Applied Electronics (AE), 2013 International Conference on*, pages 1–6. IEEE, 2013.

[SSDM13] Andreas Sailer, Stefan Schmidhuber, Michael Deubzer, and Jürgen Mottok. AMALTHEA - Plattform für kontinuierliche, modellbasierte Entwicklung. In *Tagungsband Embedded Software Engineering Kongress (ESE), 2013*, pages 538–543, 2013.

[TA] Timing-Architects. TA Toolsuite Version 13.09.0. TA Academic & Research License Program. [Online]. Available: http://www.timing-architects.com.

[TVD12] J Trumper, Stefan Voigt, and J Dollner. Maintenance of embedded systems: Supporting program comprehension using dynamic analysis. In *Software Engineering for Embedded Systems (SEES), 2012 2nd International Workshop on*, pages 58–64. IEEE, 2012.

[VdAW04] Wil MP Van der Aalst and AJMM Weijters. Process mining: a research agenda. *Computers in industry*, 53(3):231–244, 2004.

[WEE+08] Reinhard Wilhelm, Jakob Engblom, Andreas Ermedahl, Niklas Holsti, Stephan Thesing, David Whalley, Guillem Bernat, Christian Ferdinand, Reinhold Heckmann, Tulika Mitra, et al. The worst-case execution-time problem - overview of methods and survey of tools. *ACM Transactions on Embedded Computing Systems (TECS)*, 7(3):36, 2008.