Modeling and Analyzing Runtime Properties of Complex Embedded Systems

Technology Licentiate Thesis Proposal

Johan Andersson
Mälardalen Real Time Research Centre
Mälardalen University

johan.x.andersson@mdh.se

Abstract
This is a proposal for a technology licentiate thesis focusing on how to construct and analyze models of existing complex embedded systems, with respect to two important aspects of the system's runtime behaviors, timing and resource usage. Such models can be used to prototype new features and predict their impact early, before implementation. This proposal will present the motivations of this work, the research questions, related work, an outline of the thesis and a time plan.

Introduction

Industrial software systems such as robot controllers, telecom systems and process control systems have high demands on dependability. This since a system failure might cause large costs, as such system might control machinery in a production line or in other ways be of business critical nature. Systems of this kind is often large, containing millions of lines of code, and long-lived, maintained for many years. They are often too large and complex for a single person to understand in detail and they evolve as a result of new features and other maintenance operations. We refer to such systems as *complex embedded system*.

When maintaining complex embedded systems, e.g. when adding a new feature, the productivity tends to be low, due to the high complexity of the system. It is hard to predict how changes will affect the system and to verify the behavior of the system require a large amount of testing. Hard problems occur when the behavior at system level is affected unintentionally by changes in tasks. For instance when a new feature increases the execution times of a task, this might in some situation disturb the execution of other tasks, causing performance losses or even error conditions. Such errors are hard to reproduce, as they are dependent on timing, which in turn is affected by many factors, such as data dependant execution times and effects caused by the hardware.

To reduce to costs and time required to develop new features, there is a need to be able to predict the impact of changing the system with respect to runtime properties of the system. To manually predict this impact is hard as details of the runtime behavior is often not documented.

Two important aspects of the runtime behavior are the timing of the tasks and the usage of logical resources. This work is focusing on how to enable predictions of the impact of a change on such properties, in the context of complex embedded systems.

Model Construction and Impact Analysis

A common method for increasing the understandability of complex systems is to construct models. A model increases the level of abstraction and focus on the relevant information. A related example is the use of UML models for describing software, rather than relying on the "self-descriptive" code. A model describing the run-time behavior of a system could enable impact analysis of the run-time properties of interest. A prototype of the feature is added to the model and by analyzing the updated model it is possible to predict the impact.
However, models suitable for analysis of timing and resource usage properties seldom exist. Even if such models were developed in the initial design of the system, the years of evolution has probably made them outdated. To enable impact analysis thus requires a reengineering effort in order to construct a model of the system, in a suitable notation. In earlier work, we developed the modeling language ART-ML for this purpose.

An ART-ML model is analyzed using a probabilistic discrete-event simulator, generating execution traces. The execution trace is analyzed using an analysis tool developed for the purpose, the Property Evaluation Tool. The tool is rather general as it evaluated properties formulated in the query language PPL. Tasks in an ART-ML model have an explicit notion of execution time and probabilities and can be one-shot, periodic or sporadic. They have explicit attributes such as priority and periodicity. They have a behavior described in C and can communicate using message queues and synchronize using semaphores.

Before basing any decisions on a model of the system, the model needs to be validated. Validating models of complex systems is however not trivial. One straight-forward approach is to compare predictions made using the model with observations of the real system. In this approach, that can be done using the Property Evaluation Tool. A set of properties is selected used as a point of view for a comparison and evaluated with respect to both the execution trace recorded from the real system and the execution trace generated by the simulator. How to select what properties to use, what tolerances to use, and how to interpret discrepancies are however open questions.

Another use of the Property Evaluation Tool is to analyze an existing implementation. The tool allows for analyses of an existing, potentially very complex system to be made with little effort, assuming that recordings of the system can be made using e.g. software probes. By analyzing the system and comparing the results with earlier versions of the system, it is possible to get a better understanding of the system, to identify undesired effects of recent changes to the system and to identify possibly dangerous trends. We refer to this as regression analysis. This is similar to regression testing, which is commonly used technique used to verify that existing functionality of a system is not broken when changing it, for instance when adding new features. However, in this case, the focus is on non-functional properties.

Research Questions

Two research questions have been identified for the thesis, Q1 and Q2. The context of both questions is modeling of two important aspects of the runtime behavior of complex embedded systems, timing and the usage of logical resource. The first question is of a "how" nature. No hypothesis is stated on this question; a solution is proposed in the thesis. The second question is about the feasibility of this approach. In this case a hypothesis has been formulated (H2).

***Q1**: How to construct and validate a model describing the timing and the usage of logical resources of an existing complex embedded system?*

***Q2:** Is it feasible to predict the impact of a change using a model of a complex embedded system, with respect to task timing and resource usage?*

***H2:** A model of a complex embedded system can be used to predict the impact caused by a change, with respect to task timing and the usage of logical resources.*

Related work

Regarding the overall goal of this work, to increase the analyzability and understandability of complex industrial systems, there are many works with similar goals, but very different methods, e.g. special programming languages, code transformation and code visualization techniques. However, no work has been found with a similar approach as the one in this work, i.e. the use of reengineering, modeling and probabilistic simulation in order to enable impact analysis of run-time properties of complex software systems. There is however works that relate to individual components of this approach. The areas of dynamic analysis and reversed engineering are related to the construction of models for impact analysis. The areas of formal methods, real-time systems and simulation related to how models can be analyzed.

Dynamic Analysis is the area of analyzing data generated by a program at execution time and includes e.g. performance analysis, error localization and runtime system monitoring. Parts of the work within the Dynamic Analysis community also deals with reconstructing architectural descriptions of systems, based on observations.

For instance, a system called DiscoTech is presented in [2]. Based on run time observations an architectural view of the system is constructed. If the general design pattern used in the system is known, mappings can be made that transforms low level system events into high level architectural operations and from that construct an architectural description of the system. The system presented is designed for Java based systems. The types of operations that are monitored are typically object creation, method invocation and instance variable assignments. Automated, or mechanical, generation of models based on observations of the system is very related to the construction of ART-ML models. We intend to investigate automated generation/validation of ART-ML models in future work.

Another work related to the construction of ART-ML models is [3]. They present a process for reconstructing software architectures, Symphony. The process incorporates the state of the practice, is problem-driven and uses a rich set of architectural views. It provides guidance for performing reconstruction. Symphony consists of two stages. The first stage is to create a reconstruction strategy, selecting what views to reconstruct. The second stage is the execution of the strategy, i.e. to perform the reconstruction of the selected earlier views.

An approach for deterministic replay is presented in [4]. A common problem when debugging real-time systems is to be able to reproduce an error in order to find the source of this error, i.e. the bug. Due to behaviors such as task-switches and interrupts, finding the bug by studying the code alone is very hard. In their approach, they instrument a real-time system with software probes, collecting various data describing the state of the system. After an error has been observed, the data can be stored and used to replay the execution using a debugger. This is related to our approach of analyzing systems, since they use a similar technique for recording, i.e. software probes, records similar things and has the same overall purpose, to make it easier to develop complex dependable systems reliability. There are clear differences as well. Compared to this approach, much more details are recorded of the execution of the system, but for a much shorter time. They use this very detailed data to exactly reproduce an execution, in order to find bugs, while the data recorded in this approach is used to build models for impact analysis.

There is a lot of work within the area of formal methods. Model Checking is a technique for verifying different properties of models. Compared to the simulation approach in this work, Model Checking gives higher confidence, since all states of the model is explored. However, the state space explosion problem limits the complexity of the models that can be analyzed, so in many situations Model Checking is not an option. For real-time systems, a commonly used tools are Uppaal [5,6] and Kronos [12].

At the University of Aalborg, efforts have been made to automate generate models of real-time systems for UPPAAL [1]. It is however unclear from our sources whether there has been a complete implementation of the method proposed. We have not seen any evaluation of such an implementation.

There is a recently introduced commercial product called VirtualTime [11], from RapitaSystems. It is a simulation framework for real-time systems, seemingly very close to ART-ML. We have not yet been able to evaluate VirtualTime, but from their whitepaper it seems that it is targeting systems using the RTOS OSE Delta. It also seems to lack the probabilistic features of ART-ML.

A tool-suite called STRESS is presented in [7]. The STRESS environment is a collection of tools for analyzing and simulating the behavior of hard real-time safety-critical applications. STRESS contains a modeling language where the behavior of the tasks in the system can be modeled. It is also possible to define algorithms for resource sharing and task scheduling. STRESS is in some ways similar to the ART Framework, but there are a lot of differences too, as STRESS is primarily intended as a tool for testing scheduling and resource management algorithms. It does not allow probabilistic modeling like the ART Framework.

Analytical methods for dealing with probabilistic temporal attributes have been proposed in the literature. In [9], an analytical method for temporal analysis of task models with stochastic execution times is presented. However, sporadic tasks cannot be handled. A solution for this could not easily be found. Without fixed inter-arrival times, i.e. in presence of sporadic tasks, a least common divider of the tasks inter-arrival times can not be found.

Another analytical approach to probabilistic analysis is presented in [10]. Here they assume execution times and deadlines that both vary over time in an unpredictable manner, while their arrival times are fixed. Basically, the task model consists of a set of scenarios where every scenario is associated with a probability. For instance, a task may arrive with a certain execution time and deadline with a specified probability. Tasks execute probabilistically depending on several factors, e.g. the scheduling algorithm. The paper proposes solutions for Earliest Deadline First (EDF), and Least Laxity First (LLF). Even though the computational complexity of this solution has not yet been established, it seems, intuitively, that it is quite large.

Preliminary Results

Tool development

ART-ML - A probabilistic modeling language targeting complex embedded systems.

PPL - A query language for formulating properties that is of interest for analysis.

Property Evaluation Tool - A tool that evaluates PPL queries with respect to a recording.

Tracealyzer - A graphical execution trace browser.

A discrete event simulator for ART-ML models

An execution trace recorder for complex embedded systems, which have been integrated in a robot control system from ABB Robotics.

Publications

Johan Andersson, Anders Wall, Christer Norström, "*Decreasing Maintenance Costs by Introducing Formal Analysis of Real-Time Behavior in Industrial Settings*", In Proceedings of the 1st International Symposium on Leveraging Applications of Formal Methods (ISoLA '04) Paphos, Cyprus , October 2004.

Johan Andersson, Anders Wall, Christer Norström, "*Validating Temporal Behavior Models of Complex Real-Time Systems*", In Proceedings of the Fourth Conference on Software Engineering Research and Practice in Sweden (SERPS'04) Linköping, Sweden , September 2004.

Christer Norström, Anders Wall, Johan Andersson, Kristian Sandström, *"Increasing maintainability in complex industrial real-time systems by employing a non-intrusive method*", In proceedings of the workshop on Migration and Evolvability of Long-life Software Systems (MELLS '03) Erfurt, Germany , September 2003.

Anders Wall, Johan Andersson, Christer Norström,"*Probabilistic Simulation-based Analysis of Complex Real-Times Systems*", In Proceedings of the 6th IEEE International Symposium on Object-oriented Real-time distributed Computing Hakodate, Hokkaido, Japan , May 2003. IEEE Computer Society

Goran Mustapic, Johan Andersson, Christer Norström, "*A Dependable Real-Time Platform for Industrial Robotics*", In Proceedings of ICSE 2003 WADS, Portland, OR USA , May 2003.

Anders Wall, Johan Andersson, Jonas Neander, Christer Norström, Martin Lembke, "*Introducing Temporal Analyzability Late in the Lifecycle of Complex Real-Time Systems*", In proceedings of RTCSA 03, February 2003.

Thesis Outline

The thesis is to be in the form of a book, a monograph, consisting of 6 chapters:

- Introduction
- Related work
- Analyzing Runtime Properties based on Recordings
  - Motivations/Confidence issues
  - Impact of code instrumentation
  - The ART Framework
- Modeling for Impact Analysis
  - About Impact Analysis, the idea
  - Modeling Complex Embedded Systems
  - Model Validation
  - Accuracy and confidence issues
- Regression Analysis of Runtime Properties
  - General Idea, Potential Benefits
  - Introducing Regression Analysis in a Development Organization
  - Findings from introducing regression analysis in a company
  - How to validate the cost cut? Design of a case study.
- Conclusions and Future Work

Time plan

Licentiate Thesis Defense: Friday 25th of March 2005

Draft finished 2 Months before ~ End of January

Time to write a draft: 10 weeks

Related work, 3 weeks
Regression Analysis, 3 weeks
Impact Analysis, 3 weeks
The rest, 1 week

Future Work

The next step in this work is to perform an industrial case study evaluating the benefits of performing Regression Analysis, and a study on impact analysis, where we intend to validate the approach by modelling an existing complex embedded system and use the model to predict how the system will be affected by different changes. One problem with the approach described in this paper is the error-prone work of constructing the model. Instead of manually constructing the whole structural model, tools could be developed that mechanically generate at least parts of it, based on either a static analysis of the code, dynamic analysis of the runtime behavior or a hybrid approach.

References

1. P. K. Jensen. Automated modeling of real-time implementation. Technical Report BRICS RS-98-51, University of Aalborg, December 1998.

2. H. Yan, D. Garlan, B. Schmerl, J. Aldrich, and R. Kazman. Discotech: A system for Discovering Architectures from Running Systems. In Proceedings of the 26th International Conference on Software Engineering, 2004.

3. A. van Deursen, C. Hofmeister, R. Koschke, L. Moonen, and C. Riva. Symphony: View-Driven Software Architecture Reconstruction. In Proceedings of the IEEE/IFIP Working Conference on Software Architecture (WICSA'04).

4. H. Thane, D. Sundmark, J. Huselius, and A. Pettersson. Replay Debugging of Real-Time systems using Time Machines. In Proceedings of the International Parallel and Distributed Processing Symposium, 2003.

5. Johan Bengtsson, Kim G. Larsen, Fredrik Larsson, Paul Pettersson, and Wang Yi. Uppaal — a Tool Suite for Automatic Verification of Real–Time Systems. In Proc. of Workshop on Verification and Control of Hybrid Systems III, number 1066 in Lecture Notes in Computer Science, pages 232–243. Springer–Verlag, October 1995.

6. Gerd Behrmann, Alexandre David, Kim G. Larsen, Oliver Mller, Paul Pettersson and Wang Yi. Uppaal - present and future. In Proc. of 40th IEEE Conference on Decision and Control. IEEE Computer Society Press, 2001.

7. N.C. Audsley, A. Burns, M.F. Richardson, and A.J. Wellings. STRESS: A Simulator for Hard Real-Time Systems. Software-Practive and Experience, 24(6):534,564, 1994.

8. Tripac: RAPID Sim High-Performance Simulation of Real-Time Systems. http://www.tripac.com.
9. S. Manolache, P. Eles, and Z. Peng. Memory and Time-effcient Schedulability Analysis of Task Sets with Stochastic Execution Time. In Proceedings of the 13nd Euromicro Conference on Real-Time Systems. Department of Computer and Information Science, Linköping University, Sweden, 2001.

10. A. Leulseged and N. Nissanke. Probabilistic Analysis of Multi-processor Scheduling of Tasks with Uncertain Parameters. In Proceedings of the 9th Conferance on Real-Time and Embedded Computing Systems and Applications, pages 317–336, 2003.

11. RapitaSystems homepage, www.rapitasystems.com.

12. M. Bozga, C. Daws, O. Maler, A. Olivero, S. Tripakis, and S. Yovine. KRONOS: a model-checking tool for real-time systems. In *Computer Aided Verification, CAV'98*, 1998.