Formal Analysis of Component-Based Embedded Systems
Research Plan

Marcelo Santos, PhD. Student
Department of Computer Science and Electronics, Mälardalen University
email: marcelo.santos@mdh.se
Supervisor: Björn Lisper
Assistant supervisors: Jan Gustafsson and Andreas Ermedahl

April 6, 2007

Contents

1 Research Topic ........................................... 2
  1.1 Area of Research ..................................... 2
  1.2 Hypothesis and Research Questions ................. 2

2 Research Issues ......................................... 3
  2.1 Problem Statement ................................... 3
  2.2 Aims and Objectives .................................. 3
  2.3 Motivation ........................................... 4

3 State of the Art ......................................... 5
  3.1 Related Research ..................................... 5
  3.2 Research Groups ..................................... 6
  3.3 Conferences .......................................... 7

4 Research Methodology .................................. 9

5 Expected Outcomes ..................................... 10

6 Activities ............................................... 10

7 Milestones ............................................. 11
Abstract

The scope of this research is static analysis of component-based software systems, based on formal models. The problem we want to solve is how to do inter-component analysis in a system made up of software components. Firstly, given a set of components, we want to know how we can join two or more components to make an assembly. Secondly, given that each component has some properties, how can we infer the properties of the assembly? Given the broad view of this subject, we want to focus on components for embedded and/or real-time systems, and properties suitable for these kind of systems, mainly timing analysis.

1 Research Topic

1.1 Area of Research

Excerpt taken from the position advertisement:

The Worst-Case Execution Time (WCET) of a program is an important parameter in real-time systems. Static WCET analysis attempts to find safe upper bounds to the WCET by analysing the program off-line. Component-Based Software Engineering strives to increase software productivity through breaking down software into reusable components. The introduction of this design methodology for real-time and embedded software poses new problem as regards how to ensure various properties. The research topic of this PhD is timing analysis of component-based software for embedded systems, and in particular WCET analysis of such software.

1.2 Hypothesis and Research Questions

According to Wikipedia, a hypothesis “consists either of a suggested explanation for a phenomenon or of a reasoned proposal suggesting a possible correlation between multiple phenomena; in common usage in the 21st century, a hypothesis refers to a provisional idea whose merit needs evaluation”. So the hypothesis we want to investigate is:

Hypothesis 1: given the right conditions, a component-based embedded system (CBES) is timing predictable.

For this hypothesis, we want to find answers to the following questions:

1. what are the problems and issues related to timing analysis of CBES?
2. how to design a component for timing predictability?
3. what is a suitable composition mechanism to ensure timing predictability in a CBES?
4. what properties should a component model have in order to be time-predictable?
5. as the timing analysis, mainly WCET, is tied to the hardware, how does the architecture of the hardware dictate the component and formal models?
What we mean by right conditions is that they are statements that must be true in a formal mathematical system, where components and component composition are well defined entities. So the formal system includes:

- a formal model for components and component composition;
- a formal (abstract) model for the hardware;
- a formal model for the software.

But what if most practical systems do not follow the right conditions? Then our work will be of less practical use, mainly due to new hardware performance features that are intrinsically unpredictable. If this is the case, we want to investigate the following

Hypothesis 2: given the probabilities that a component will miss its deadline, and a system made up of such components, we can calculate the probability that the system will miss its deadline.

This hypothesis is also interesting in the context of soft real-time systems, where strict deadlines are not needed, but needs some overall performance. In this case, we want to find answers to the following questions:

1. given that a task (or system) has a maximum time to execute, how often will it miss its deadline?
2. how can we calculate the probability that a component will miss its deadline?

We cannot forget that the system have other properties of interest besides timing analysis, so we need to be sure that they are not in conflict with each other. This can put some limitation on what we can do.

2 Research Issues

2.1 Problem Statement

We want to do research in the formal analysis of component-based software for embedded systems in order to be able to predict properties of interest to such systems, based on the properties of the components. Some properties of interest for these kind of systems are for example resource consumption like memory, energy, time, CPU, etc. The focus of the research will be timing analysis, mainly in order to find the worst-case execution time for an assembly.

2.2 Aims and Objectives

What we want as the ultimate goal with this research is be able to do timing analysis of embedded component-based systems based on the analysis of the components. So the goal of this research project is static formal analysis of embedded component-based systems, with the property of interest being timing analysis, particularly worst case execution time (WCET). In general, we would like to know how the properties of the system can be inferred from the properties of its constituent components. We could also go the other way round: in order to be
able to predict such and such properties of the system, how should the components be designed and implemented and what properties should they have?

2.3 Motivation

The idea of dividing a system into smaller, manageable pieces is not new: structured and modular programming came as a promise to help build better and more reliable software, but failed in this goal due to always increasing size, complexity and requirements of modern software systems. Many would argue that future breakthroughs in software productivity will depend on our ability to combine existing pieces of software to produce new applications (see for example [21]), but this new approach has its own intrinsic problems: some problems with integration of components in the tool Aesop are described in [12]:

- excessive code size;
- poor performance;
- the need to significantly modify the reused subsystems just to get them to work together;
- the need to re-invent existing functionality in order to match the intended use;
- unnecessary complexity of applications built on top of the reused systems;
- a complex, error-prone system construction process.

The authors call these problems architectural mismatch: it is related to the implicit assumptions that the components make about the environment in which they are supposed to be deployed. In order to solve these problems we need to find ways to codify and disseminate principles and rules for composition of software.

The search for a suitable component model is not new. In [18] some general directions and importance are already posed, with no common model existing for the notions of components and composition, making it hard for a uniform description. The authors already said that a practical composition methodology have to accomodate the use of existing and heterogeneous component libraries and applications. Still nowadays, there is not a universally accepted terminology for component-based development (CBD), with no common concept of what a component is. Even with these drawbacks, software component composition is central to component-based development and composition can be made in the design phase (stateless components) [18] or deployment phase. But even if we find a good abstract way to formally specify the composition, we still need to find a good way to map a problem domain to these abstractions.

More recently, [16] presents a survey of some component models, and explains briefly how composition is made in each of them. It is argued that a composition theory is needed in order to reason about composition and that, despite some models having a composition language, they lack such a theory (a composition theory allows us to calculate and predict the result of applying a composition operator to components). For example, the PIN component model [11] has a composition language CCL, but do not have a formal analysis.

Why do we want to do a formal analysis of a software system? The formal analysis is used to predict some (emergent) properties of an assembly of components
and is based on the individual properties of the components (so the components also have to be amenable to some kind of analysis). We can make prediction of properties like correctness, performance, reliability, security, WCET, etc. And we want them to be amenable to theoretical validity and empirical validity, i.e., we want to validate the predictions. For example, given the probabilistic WCET of each component, like in [5, 6], how can we predict the WCET of the assembly?

Some other issues that we need to take into account when doing timing analysis:

- mix of soft and hard-real time tasks in the same system;
- increasing hardware complexity — an abstract model need to include the functionalities of modern architectures:
  - multi-core processors
  - cache memories
  - pipelining
  - branch prediction
  - FPGA

3 State of the Art

3.1 Related Research

Rather than designing a new formal system from scratch, we can use the idea presented in [2], called PECT (Prediction-Enabled Component Technology), where a component technology is integrated with one or more analysis formalisms, possibly restricting the design to make the system analysable. Because the two technologies, formalisms and components, evolved separately, we can integrate them through co-refinement, where we can make them more specific or more general, depending on the needs. And this is the approach where research is going on. For example, in [20], the formalism called Wright, a formal language for describing software architecture that is a variant of CSP [14], is used to model the Enterprise JavaBeans component integration framework. Of course, other formalisms can also be used: for example, Hoare’s CSP is also used in CL, a compositional language for the Pin component model [15]. Other process calculus being combined with component and connector syntax are, for example CCS, π-calculus, FSP (Finite State Process), Petri Nets [9] and Timed Automata [4]. An algebraic method for architecture description is presented in [7] and borrowing ideas from abstract data type, [3] presents ABT (Abstract Behaviour Type) as a foundation for components and component composition.

There are different ways to compose the properties of the components. For example, [8] try to tackle the problem of composing component properties for invariants used in concurrent and reactive systems. But depending on the definition of invariants, they may not be composable. Inductive and trace-based invariants are analysed. There are a reasonable amount of research going on about formal analysis of component systems in general, but none about timing analysis of an assembly based on components (I need to do more bibliographical research - maybe I have missed some). Most of the research is on model checking,
analysing general properties of the system. Others are based on the components, like [20, 1]. Most forms of automatic validation of composition is through type checking, for example, [13]. Another approach that is being taken is to see the composition as an optimization problem. In [17], a genetic algorithm is used to select a best assembly from candidate ones. For deployment, [19] describes a framework for formal analysis, where one can express conditions for successful deployment, meaning that the deployed application works, and safe deployment, meaning that the deployment do not damage existing applications.

3.2 Research Groups

There aren’t many groups doing research specifically on the composition of components. But of course a particular group will be interesting depending on the direction of the research. As we are going to focus on embedded systems, there are several research groups:

**Software Composition Group**, University of Berne, Switzerland. This group conducts research into tools, techniques and methods for constructing flexible software systems from components. The research in this group is for components in general and a formal method Piccola, based on the π-calculus is being developed [1] and is being applied to Java (JPiccola).

The url of the group is: url: http://www.iam.unibe.ch/scg.

**Embedded Systems Research Lab**, coordinated by Prof. P.S. Thiagarajan, School of Computing, National University of Singapore. The research topics of interest to this group are: formal specification, verification and synthesis methods, timed and hybrid systems, compilers and architectures for embedded systems, hardware synthesis and power aware computing. They do hardware oriented and software oriented research. Particular interesting are the projects **Formal Design Techniques for Reactive Embedded Systems** (leader: Prof P.S. Thiagarajan) and **Techniques to Support Timing and Power Guarantees for Embedded Code** (leader: Dr. Tulika Mitra).

The url of the lab is: http://www.comp.nus.edu.sg/~emsys/

**The Embedded Systems Institute - ESI**, Eindhoven Netherlands. Between the system qualities that are important for embedded systems, the ESI focuses are on performance, reliability and evolvability. Of interest to my research is of course performance, where the institute focus is in correct timing behavior and efficient use of resources. There are currently several projects going on at the institute, and the mostly related to my research is Bodere (project manager: Bauke Sijtsma), that is doing research on distributed embedded real-time controllers of complex systems, with six parties envolved. The goal of the project is “an integral approach for a systematic architectural design, modeling, analysis, and validation methodology of such heterogeneous systems”.

Url of the project: http://www.esi.nl/site/projects/trader.html

**VERIMAG Research Center**, Gieres, France, is a leading research center in embedded systems, providing theoretical and technical means for developing embedded systems. The research center tries to maintain a good balance between fundamental, experimental and applied research. There are several research groups in the research center. The **Timed and Hybrid Systems** group (research director: Prof. Oded Maler) is interested in “all aspects of system design, ranging
from theoretical foundation, via design techniques, down to implementation”. Url for the research group: http://www-verimag.imag.fr/TEMPORISE/

Compiler Design lab at Saalands University, Saabuecken, Germany. The research in the lab (leader: Prof. Dr. Reinhard Wilhelm) is about formal analysis and predictability of real time systems, using various techniques like static analysis and abstract interpretation, in order to improve the safety of embedded applications. Url: http://rw4.cs.uni-sb.de/

Computer Engineering Group at Department of Information Technology and Electrical Engineering, ETH Zurich. The Computer Engineering group, headed by Prof. Lothar Thiele, concentrate their activities on design, engineering methodologies, and tools for embedded (software) systems. Url for the research group: http://www.tik.ee.ethz.ch/~tec/

Specification and Analysis of Embedded Systems, at the National Research Institute for Mathematics and Computer Science, Netherlands. The leader is J.C. van de Pol and this research cluster is part of the Software Engineering theme. The main goal of this research is to develop techniques that allow an increase in the quality of embedded and real-time software by employing analysis techniques. Url: http://db.cwi.nl/projecten/thema.php4?themanr=14

3.3 Conferences

Nearly all conferences in software engineering have formal analysis and components as one of the topics. Conferences in this subject are:

- (ICSE) International Conference on Software Engineering, in its 27th edition this year in Minneapolis, USA. Url: http://web4.cs.ucl.ac.uk/icse07
- (SCBSE) the International ACM SIGSOFT Symposium on Component-Based Software Engineering, to be held this year at Tufts University, Massachusetts, USA. Url: http://www.comparevents.org/pages/present.html
- (APSEC) the Asia Pacific Software Engineering Conference, this year in its 13th edition, is going to be held in Bangalore, India. Url: http://www.cse.iitk.ac.in/apsec06/
- (SBES) the Brazilian Symposium on Software Engineering in its 21st edition this year to be held in Joao Pessoa, Brazil. Url: http://www.sbbd-sbes2007.ufpb.br/
- (VMCAI) International Conference on Verification, Model Checking and Abstract Interpretation, this year on the 8th edition that was held in January
in Nice, France.
Url: http://research.microsoft.com/vmcai07/

As this research will deal with formal analysis, events on formal methods are also suitable. Some events with formal methods as main topic:

- (FACS) http://dsrg.mff.cuni.cz/conferences/Promotion-FACS2007.html: 4th International Workshop on Formal Aspects of Component Software, this year to be held in Sophia-Antipolis, France;
- (SC) http://ssel.vub.ac.be/sc2007: workshop on Software Composition, held within the European Joint Conferences on Theory and Practice of Software (ETAPS), this year to be held in Braga, Portugal.
- (FME) Formal Methods Europe, is going to be held at Åbo Akademi University, Turku, Finland. Url: http://www.fm2008.abo.fi/
- (SBMF) Brazilian Symposium on Formal Methods, in its 10th edition this year, to be held in Ouro Preto, Brazil. Url: http://www.sbmf2007.ufop.br/papers.html
- (ICFEM) International Conference on Formal Engineering Methods, with the 8th edition held in 2006. Url: http://www.iist.unu.edu/icfem06/
- (SAS) International Static Analysis Symposium, this year on the 14th edition, to be held at the Technical University of Denmark, Kongens Lyngby, Denmark. Url: http://www2.imm.dtu.dk/sas2007/

The research is going to be focused in embedded systems, mainly real-time systems. Depending on the direction that the research takes, the following conferences in these subjects are also suitable:

- (RTSS) the IEEE Real-Time Systems Symposium is in its 28th edition this year and is going to be held in Tucson, Arizona, USA. Url: http://www.rtss.org/
- (IESS) the International Embedded Systems Symposium, to be held this year in San Diego, USA. Url: http://www.ieess.org/
• (CAV) the International Conference Computer Aided Verification, in the 19th edition this year, is going to be held in Berlin, Germany. Url for the conference: http://cav2007.org/

• (ECRTS) the Euromicro Conference on Real-Time Systems, in the 19th edition this year, is going to be held in Pisa, Italy, particularly the International Workshop on Worst-Case Execution Time (WCET) Analysis. Url for the conference: http://feanor.sssup.it/ecrts07/

• (EMSOFT) annual ACM Conference on Embedded Systems Software, in its 7th edition this year to be held in Salzburg, Austria. It is a guest conference of the Embedded Systems Week. Url: http://www.emsoft.org/

I think that the main events where we can publish our results are:

• SC - Symposium on Software Composition
• FACS - Workshop on Formal Aspects of Component Software
• FORMAT - Conference on Formal Modelling and Analysis of Timed Systems
• SAS - International Static Analysis Symposium
• WCET workshop held in conjunction with the ECRTS conference

4 Research Methodology

In principle, we will use the inductive method, also called bottom up approach: we will observe some real-time embedded systems and identify their commonalities in order to come up with a theory to be used by all systems of this kind. With the observations, we will make a suitable definition of what is meant by a component in this specific application area, for example. With this, we have a standard (formal) way of defining and building components, making them amenable to a formal analysis. There are several aspects that can be observed and the one more interesting to this research is WCET.

But we can also use the methodology categories presented in [10], where research projects in computing are classified into four types, and a project can have a flavour of each type. This is the case for this project:

• it is of type research based, as I will do a investigation of a particular area (backward looking) to establish the field of the research, and then the contributions will come (forward looking), expanding what is already known in the area;
• it is of type development, as new formal methods or expansions to current methods can be proposed;
• it is of type evaluation, as I want to know how good is a method in predicting and analysing assemblies of components;
• it is of type problem solving, because new algorithms or methods might be needed to do a better analysis of systems made of components.
We don't need to develop a new component or formal model from scratch: there exists today several formal models and several component models. Usually these are separate entities and have evolved independently, with some component models having their formal analysis while others not. We also want to do research in what kind of formal models can be applied (or extended) to component models for embedded systems. Some of the models have an IDE (integrated development environment) where it is possible to do testing and verification (model checking), like Petri Nets and Timed Automaton.

5 Expected Outcomes

We expect to make a contribution to the PROGRESS research by defining a formal analysis for component-based embedded systems, in order to do timing analysis, particularly WCET. We also expect to publish the results in some leading conferences (and also symposiums and workshops), besides writing reports and the Licentiate and PhD theses.

6 Activities

The PhD graduate programme consists of three related activities: courses, research and department duties.

Courses: the courses are good because they complement the graduate education, filling gaps in the education not directly related to the research, but important in general. For example, the courses PhD School, Research Planning and Research Methodology for Computer Science and Engineering are quite useful. Other courses are directly related to the research, such as Real-time Embedded Systems. Also the possibility to take courses at another place is very good, as it helps to get involved with other researchers and students. As a graduate student at the department of Computer Science and Electronics I participate in two national graduate education schools: the ARTES network for real-time research and graduate education in Sweden, and the CUGS (National Graduate School in Computer Science). These schools organizes courses and events that I can attend as part of the education. The requirement is that all the courses together amount to at least 60 credits. Depending on course availability, I plan to finish all the credits in the first two years.

Department duties: as a graduate student, I am expected to assist in lecturing, exercises, lab activities, etc. for undergraduate students. The Syllabus for Graduate Education in Computer Science and Engineering state that other duties at the department may include: advising of other graduate students, writing proposals to help fund new (and continuing) students, presentations to visiting researchers (and the public), membership in administrative groups (e.g., network planning). I think it is quite good to have these experiences provided that they do not overtake the research: the research can be quite demanding in some semesters and the twenty percent department duties can become a hindrance, so this must be very carefully planned.

Research: The subject of my research is component based development for real-time and embedded systems, mainly formal analysis, with a focus
on timing analysis and worst case execution time, probably using static analysis and/or a probabilistic approach. In this area, we plan to write reports and papers related to the research questions and literature survey:

1. literature survey about formal analysis of component-based systems in general: we intend to write the state of the art report and send a paper to the FACS conference this year (the deadline is June 12);

2. problems and issues related to timing analysis of CBES. This paper can be sent to a general paper about embedded systems, like FORMAT. But should we focus on the use of abstract interpretation, we can send it to SAS conference instead;

3. properties a component model should have in order to be time-predictable: this work implies a couple of suggestions for a suitable definition of a component model;

4. designing a component for timing predictability: this work is more time demanding, as it implies that we have a suitable formal analysis;

5. composition mechanisms to ensure timing predictability in a CBES.

This year I plan to attend the SC - Symposium on Software Composition and FACS - workshop on Formal Aspects of Component Software. We also plan an international visit/collaboration with a research group (around 6 months) in 2009.

7 Milestones

It is suggested in the Syllabus for Graduate Education in Computer Science and Engineering that the licentiate thesis should be based on three papers that can be published in international workshops or conferences, with the candidate as the main contributor. And that the Ph.D. thesis should be based on a couple of more papers, with at least one of them suitable for publication in an international journal. It also suggests to divide the organization into four milestones, approximately one each full year of the graduate education. This division is quite good in order to assure a good quality and progress of the research:

1. Licentiate thesis proposal
2. Licentiate thesis defence
3. PhD thesis proposal
4. PhD thesis defense

We plan the following timeline to achieve these goals:

- Year 1: First paper (state-of-the-art report, survey of the intended field of the thesis), with at least 20 course credits obtained;
- Year 2: Thesis proposal for Licentiate degree (public presentation of thesis proposal), with at most 10 course credits left for licentiate degree; second and third paper;
- Year 3: Licentiate degree (public licentiate defence); forth paper;
• Year 4: Ph.D thesis proposal (public presentation of this proposal), with no course credits left for Ph.D degree; fifth paper;
• Year 5: Ph.D thesis defence and journal paper.

The PhD started in the end of October 2006 and is planned to be finished in October 2011, with 80% research and 20% department duties. The final date will be anticipated accordingly in case there is no need (or a partial need) for the department duties.

References


