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 have 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 Introduction

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. 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 [13]). Some problems with integration in the tool Aesop are described in [7]:

- 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 properties of interest depend on the environment that the component will work. For example, for a component used in an internet transaction, security might be the most important requirement, but in a hard real-time system, the time is the most important requirement.

2 Research Topic

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. WCET analysis of Component-Based Embedded Systems 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.

Some 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:
  - multi-core processors
  - cache memories
  - pipelining
  - branch prediction
3 Research Issues

3.1 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 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.

3.2 Aims
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.

3.3 Objectives
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?

3.4 Motivation
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 [2, 3], how can we predict the WCET of the assembly?

Component composition is central to component-based development. A composition can be made in the design phase (stateless components) [11] or deployment phase. Still nowadays, there is not a universally accepted terminology for CBD, with no common concept of what a component is.

The search for a model is not new. In [11] some general directions and importance are already posed, no common model exists for the notions of components and composition, making it hard for a uniform description; that’s in 1995. The authors already said that a practical composition methodology have to accommodate the use of existing and heterogeneous component libraries and applications. 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, [9] 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 have 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 [6] has a composition language CCL, but do not have a formal analysis.
4 State of the Art

4.1 Related Research

How do we compose the properties of the components? For example, [4] try to tackle this problem for invariants used in concurrent and reactive systems. 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 [12, 4]. Most forms of automatic validation of composition is through types, for example, [8]. Another approach that is being taken is to see the composition as an optimization problem. In [10], a genetic algorithm is used to select a best assembly from candidate ones.

4.2 Research Groups

There aren’t many groups doing research specifically on the composition of components. The Software Composition Group conducts research into tools, techniques and methods for constructing flexible software systems from components, located at the University of Berne, Switzerland (url: http://www.lam.unibe.ch/scg). This group does research for components in general and a formal method Piccola is being developed [1] and is being applied to Java (JPiccola). 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, for example:

- Embedded Systems Research Group in Singapore (School of Computing, National University of Singapore).
  Url: http://www.comp.nus.edu.sg/embsys/
- Embedded Systems Research Group in the Department of Electrical and Computer Engineering, University of Auckland, New Zealand.
  Url: http://www.ece.auckland.ac.nz/embsys/
- The Embedded Systems Institute, Eindhoven Netherlands.
  Url: http://www esi.nl/
- VERIMAG research center, Gieres, France.
  Url: http://www-verimag.imag.fr/
- Compiler Design lab at Saalands University, Saabrucken, Germany.
  Url: http://rw4.cs.uni-sbg.de/
- Software for Systems on Silicon, Aachen, Germany.
  Url: http://www.iss.rwth-aachen.de/

4.3 Conferences

Nearly all conferences in software engineering have formal analysis and components as topics. The most important one seems to be the ICSE - International Conference on Software Engineering, in its 27th edition this year in Minneapolis, USA (url: http://web4.cs.ucl.ac.uk/icse07). Other conferences in this subject are:

- the 6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering in its 6th edition this year, to be held in Dubrovnik, Croatia.
- the International ACM SIGSOFT Symposium on Component-Based Software Engineering, to be held this year at Tufts University, Massachusetts, USA. 
  Url: http://www.comparc-evens.org/pages/present.html
- 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/
- 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/

With formal methods, the IEEE International Conference on Software Engineering and Formal Methods is in its 5th edition. Url: http://www.iist.unu.edu/SEFM07. There are also some specific suitable workshops:
- http://ssel.vub.ac.be/se2007: 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.

As this research will deal with formal analysis, events on formal methods are also suitable:
- Formal Methods 2008, organized by Formal Methods Europe, is going to be held at Åbo Akademi University, Turku, Finland Url: http://www.fm2008.abo.fi/
- 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

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:
- 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/
- the International Embedded Systems Symposium, to be held this year in San Diego, USA. Url: http://www.iess.org/
- the International Conference Computer Aided Verification, in the 19th edition this year, is going to be held in Berlin, Germany. Url: http://cav2007.org/
- 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: http://feanor.sssup.it/ecrts07/

5 Research Methodology

There exists today several formal models and several component models. Usually these are separate entities and have evolved independently, but some component models have their formal analysis while others don't. We also want to research what kind of formal models can be applied (or extended) component models for embedded systems. Some of the models have a IDE (integrated development environment) where it is possible to do testing and verification (model checking).

In [5], the 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 an 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.

6 Expected Outcomes

We expect to make a contribution to the PROGRESS research by defining a formal analysis for component-based embedded systems, particularly timing analysis. We also expect to publish the results in some leading conferences (but also symposiums and workshops).

7 Milestones

The Graduate Programme proposes the following milestones for the PhD with eighty percent research (four years full time):

• Year 1: First paper (state-of-the-art report, survey of the intended field of the thesis), with at least 10 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;

• Year 3: Licentiate degree (public licentiate defence);

• Year 4: Ph.D thesis proposal (public presentation of this proposal), with at most 10 course credits left for Ph.D degree;

• Year 5: Ph.D thesis defence.

The PhD started in 2006 and plan to finish in 2011.

References


