Session 3: WCET calculation methods

Session chair: Jan Gustafsson (University of Mälardalen, Sweden)

Presentations

The first paper *"A Distributed WCET Computation Scheme for Smart Card Operating Systems"*, by N. Aissa et. al. presented a new method to calculate the WCET on a security protected and very constrained device (a smart card).

The second paper "*Inspection of industrial code for syntactical loop analysis"* by C. Sandberg described methods for analysis of "real" code and some preliminary results.

In the third presentation *"A New Timing Schema for WCET Analysis"* by S. Petters et al., a new technique for handling path-based and low-level analysis.

The forth paper, *"Petri Net Level WCET Analysis"* by F. Stappert described an approach for WCET calculation based on high level Petri Nets.

The last paper in the session, *"Measurement-Based Worst-Case Execution Time Analysis using Automatic Test-Data Generation"* by R. Kirner et al. described a hybrid approach to calculate the WCET partly based on static analysis, partly on measurements.

Discussion

The discussion that ended the session was mainly concerned with the development in the WCET research area since the WCET workshop started in 2001 and the future of the workshop.

Jan Gustafsson: I see two main trends in the WCET area. The first is academic; this area wants to grow and the workshop would like to extend be longer, have more papers etc. So the first question is: *how do you look upon this workshop and the future of it?*

The other trend is to target the industry and to get our methods known and used there. Since the WCET workshop started, at least three tools have been developed for the industrial market. We all hope that these companies will succeed, since this is of benefit for us. We also hope that new companies and tools will emerge. This is because there is a synergy between these tools and the research. We have seen one paper today (the effect of scratchpad memories on WCET prediction[2]) where the aiT tool has been used to support research, and we can expect more papers in the future.

In my department, the aiT tool is used to study the use of static WCET analysis in industrial settings. The results can be used as input to our research.

[2] Influence on Onchip Scratchpad Memories on WCET, by L. Wehmeyer, P. Marwedel, University of Dortmund, Germany

So the other question is obvious: *how do we find our way out to reality*? Can we push even harder? Maybe this workshop can help, too. Maybe we can start an "WCET Interest Group" which is active also between workshops.

One important issue is the one about benchmarks. A common WCET benchmark allows tools and methods to be compared. There is a lot of work to be done here.

Guillem Bernat: There are many questions! What we would like to suggest is that we get a repository where we can store all these codes. For example we could have code with many nested loops.

Friedhelm Stappert: We already have such a repository[3]. We volunteer to have it extended.

Jan Gustafsson: A problem here is that some of the interesting code is not accepted for spreading by the companies. And we want real code. So how do we get it?

Peter Puschner: I don't know if this is a solution, but I think even the patterns are interesting. So, if you find some patterns that you think are interesting, you may create "dummy" code with maybe not the same semantics as the real code, but with similar patterns, say control structure or other property, like dependence on pointers.

Jan Gustafsson: Well, it has to be semantically correct so you can run it and measure it.

Peter Puschner: Well, it should be able to do a simple version with a correct semantics but still with the interesting structure.

Iain Bate: Thanks for the offer! We already have some samples that we used before. We are looking at some software that we might have there on the benchmark and which is available stuff. What we don't want is too many examples. If you look at other benchmarks in other areas you see that they are small. We don't want dozens of examples but rather get it down to 3 or 4 examples. Otherwise, the different groups will look at different subsets of the examples.

A second issue is benchmarks for hardware. It would be good to choose 2 or 3 different processors, maybe one with a simple pipeline, another one that is a little bit more complex, and the last one at the "bleeding edge". This is because it makes no point to have common software and then compile them to and analyze them in different targets.

Jan Gustafsson: So you mean that you should force people to use the same codes, and not select the examples that works just for them?

Iain Bate: Yes. And, for the software, we could have the similar selection as for hardware, that is a simple one, in intermediate on and a complex one.

Raimund Kirner: I have a comment on the type of codes we want to collect. A nice thing with the codes that Christer studied is that "this is code from reality". Also, if you have a nested loop with some dependencies inside which show some difficulties, which poses problems for the WCET analysis, you might argue that this is a sorting algorithm and these problems might not be interesting for other areas of practice. So, we should keep in mind what is the typical application and avoid studying code patterns that are not relevant in practice.

[3] http://www.c-lab.de/home/en/download.html#wcet

Also there are a lot of code examples in books, and maybe we can copy them for a use like this.

Jan Staschulat: My observation is that the WCET community is always a number of years behind the latest developments in processor architecture. I have seen this with the pipelines, then the caches, and now we have the scratchpad memory. First, a new hardware feature is introduced and then a couple of years later an analysis by some research team is proposed to cope with the new complexity. Instead, shouldn't the WCET community propose how a processor should look like to be fast and predictable?

Unidentifier speaker: Right now the processors are going in a direction with more complex features like out-of-order execution and similar. So on the contrary – it is the WCET community has to follow. I think that it is not possible to force the industry to produce processors that are fast and predictable. It is probably more useful to try to overcome the problems with really unpredictable processors.

Jan Gustafsson: In our project in Sweden we have deliberately concentrated on simple processors for embedded systems. These are today typically without caches and even pipelines. These embedded systems also often have hard real-time requirements. In the other end with the really complex processors it may be better with statistical WCET methods, but then you will not have the full safety. This is my simple view, but if you can come up with something better, many should buy that.
I don't think that we can steer the hardware community. It is too much money involved. But we can pick the processors that suit our methods and the customers can do that, too.

Tullio Vardanega: It really seems to me that there are two views. The first is *speed-oriented* software, whether it is hardware or software. The other is *timing-aware* processing and coding.
I can see that there are real academic challenges to take any sort of jumbled code and to derive information from that. But it would be much more valuable to tell the users that there a number of coding styles or idioms that are timing-aware and much more amenable to timing analysis, no matter what the processor is. I see no effort indicating to people that there are idioms that are wrong, and it's no good to squeeze and push our tools that they can understand this rubbish!

Jan Gustafsson: Have you seen rubbish? Are there coding styles which are no good?

Tullio Vardanega: I have seen lots and lots of coding styles which are speed-oriented and which are totally wrong. One can tell (or smell) that what was behind that code was speed and nothing else.
Everybody have a responsibility, when you are a programmer or designer, to solve the timing problem. It is no good to delegate it to a magic tool – to write rubbish and to send it to a tool to solve it. I don't believe in magic.

Peter Puschner: Exactly as you say we have to look for the priorities. Speed is often number one, and what we would like to add is predictability. But really it depends on the area we are looking at. If we are looking at hard real-time systems that need to be dependable, we want predictable timing. Then predictability is the number one issue, but speed if of course still important.
If we want predictability we should design our systems with this in mind. The whole architecture, both hardware and software, should be laid out towards that goal.

We are doing some work in these issues and my feeling is that even if you aim at predictability, the speed, in terms of short worst-case timing, will come as an add-on.

Tullio Vardanega: People are implicitly presenting the notion that real-time programming is complex. For example going from Java to Real-time Java adds on a lot of classes. But when you are programming against time, you take a number of constraints (or even short-cuts) with you and you are more disciplined. It the cost of something, of course.
It is a service to the community if these things are told up-front, and not are discovered at the end of the slope.
The cost you pay is a certain limitation of freedom. I would really like this community of cultivated people to say this to the programmers and don't just say "Give us anything, we will process anything".

Peter Puschner: Just a picture of the user: if you work with a nail you use a hammer, and if you work with a screw you need a screwdriver, but not the other way around. And it's the same here. Use the right tools!

Iain Bate: The only counter-argument to what I am hearing is that you already know all this. For example a lot of the industrial examples are simple hardware and software. But from the academic perspective I assume that this is less interesting. It is useful to look at the more nasty control flow graph for example. It is more of a challenge.

Jan Gustafsson: I agree with Tullio but I don't think that the word is spread all around.

Guillem Bernat: People are aware of what is possible and what is not possible. Moreover, in a real setting there is a lot of priorities, the software, the processor, the development process, etc. To make radical changes, for example: do not use this processor, may be impossible since it may have been selected by other criteria.
So if we work for this aim it is good, but we have only limited possibilities in our position. Sometimes we will have to live with the fact that other criteria completely steers the situation.

Iain Bate: I would like to bring up the second question that Jan brought up namely: how we raise the profile of what we are doing, and what is the next step for a workshop like this?
Our workshop has a good form, and is interesting, but obviously from an academic perspective what we need is to move towards longer papers, more stringently reviewed, longer presentations etc. That is, moving towards a conference, and in my view, have proceedings that are more academically credible, like an IEEE publication with an ISBN number and all of these good things.

Guillem Bernat: We have been discussing this before with Peter Puschner. We are very happy to see all these new faces and you are all very welcome. I hope we all share the same view, i.e., move towards IEEE proceedings etc.
Have many of you have come only to the workshop and not the full ECRTS? (Many raised hands) So, there is no longer a great risk of this workshop would die if it was organized on its own and not as a satellite, as the situation was a number of years ago.
I would like to go to a longer workshop with longer papers etc., but the issue is would we get enough contributions for that?

Peter Puschner: One important point is the flavor of the workshop. The original idea when we started the series was to have room and time for discussion, and not to make this "yet another conference" with one paper presented after the other with almost no discussion, and people submit papers only to get a publication. The flavor shouldn't change a lot so we should avoid moving to these of these "standard conferences".

But, it is definitely a good idea to think about for example publication that goes beyond just a technical report.

Jan Gustafsson: Another thing that would be good is to continue these discussions between users and developers. This workshop would become a meeting point with prepared talks and prepared discussions. This is not in conflict with higher academic goals. Maybe this requires two days.

Peter Puschner: It is also this point of having something to present. How many of you wouldn't have the possibility to come without having a paper presented here? (Some raised hands)

Guillem Bernat: Well if your paper does not show up in the proceedings it does not always give the proper founding for travel. So this calls for a proper proceedings which you can reference and so on.

Iain Bate: Maybe a solution is to have a two day workshop with not many more papers but longer and more detailed papers, while still maintaining the discussions, which can be very useful. To come for 4 or 5 days, the whole week is gone. Two days is just as easy.

Björn Lisper: Here is another suggestion. We can keep the present format of the workshop, with short papers, but then invite authors will have to submit a longer version afterwards. These papers should be reviewed and then be published as post-proceedings. Then we would get proceedings that are peer reviewed and on par with ordinary conference proceedings.

Guillem Bernat: Well, this would not solve the problem that some people do not get funding if they do not have a paper accepted.

Iain Bate: It is more than that. You want to align to a journal or something like that. In the UK for example it is important where you publish and a workshop like this is not of the same high value as a journal.

Guillem Bernat: I would like to go back to the benchmark issue. We can have something like a competition between PhD students who can go to this workshop. They can try different methods on different problems and in that way find open problems, using real-life pieces of code.

Jan Gustafsson: This is a nice suggestion; we can give this community problems to solve ad we can then discuss the solutions. This should be on free will – you can write about this or something else if you like.

Guillem Bernat: Yes, you can just register if you're interested. There is an interesting Matlab programming competition where they set up a problem and then they get a score for their solutions. It is interesting to see how people can really compete to really squeeze smart solutions out of this.

Jan Gustafsson: So, there is still work to do for the organizing committee. So finalizing this workshop, I would like to give the word to Isabelle [Puaut] for some final words.

Isabelle Puaut: I would like to thank everybody for coming, a special thanks to the session chairs, and the organizing committee. And don't forget to come to the restaurant tonight!