Certifiable Java for Embedded Systems

Objectives of the Project

Embedded systems are increasingly becoming part of our daily life. Some of these systems, for example the control of the Copenhagen Metro, are safety-critical, as our *real* life can depend on it. Such systems need to be certified to be used safely.

The aim of this project is to develop a prototype development environment and platform for safety-critical software for embedded applications. There are three core constituents: A profile of the Java programming language that is tailored for safety-critical applications, a predictable Java processor built with FPGA technology, and an Eclipse based application development environment that binds the profile and the platform together and provides analyses that supports a safety case.

The main novelty of the project is that it consolidates and integrates a number of results from previous research by members of the team and others in a consciously engineered development environment and platform that supports certification of developed applications. The previous results include development of a highly predictable Java processor [23, 25], contributions to profiles for predictable Java [29, 3, 12], and development of analysis tools [4, 28, 19]. As most of the previous results are open-source,[1] the results from this project will be available as open-source as well.

The expected result is of immediate interest for industry engaged in aerospace software development, as witnessed by the recent document by the European Space Agency (ESA) on *On-board Software* [31]. Thus the company GomSpace is following the development closely and is investing resources in giving advice on applicability. In a larger perspective, the technology is important for other companies that develop software intensive systems that have to be certified. The results are also expected to have an impact within the new European, Artemis funded, research project RECOMP.

Background

The key elements in the project are: certification, Java for real-time systems, the time-predictable Java processor JOP, and static program analysis.

Certification

The most costly real-time systems to develop are safety-critical systems. A failure in a safety-critical system can, in the worst case, result in loss of live. Therefore, safety-critical systems undergo a rigorous certification process, e.g., in the United States by independent organizations using for instance the

[1] see `http://www.jopdesign.com/` and `http://www.jopwiki.com/`

DO-178B [22] standard. It is important to note that certification is concerned with a concrete system, where software and digital hardware are just subcomponents. Therefore we use the term certifiable in this project, because we can provide tools and artifacts that contribute to a credible safety case for such components. This will be an important step, because so far, certification has had to be based on laborious manual inspections of software for selected hardware configurations. In the project, we will learn from results of the very recent European Artemis RECOMP project, where several of the applicants are key participants. RECOMP is orthogonal to the proposed project. RECOMP will provide mechanisms for spatial (e.g., protected memory) and temporal (e.g., time slots, virtualization) partitioning of applications with different Safety Integrity Levels.

Java Profile

Java is selected as target language, because object-oriented languages provide benefits when building software. Java is, compared to C/C++, a safer language by design. Other features, such as dynamic method dispatch and garbage collection, are a challenge for the certification process. Therefore, a tight and small subset of the Java virtual machine functionality and libraries is necessary. The standard on Safety Critical Java Technology (SCJ) [15, 12] addresses these issues and enables building safety-critical application certifiable under DO-178B, Level A and other safety-critical standards. The PI of the proposed project is member of the Expert Group for SCJ, therefore the project team has access to early drafts of the standard. Conversely, the insights we gain on Java for safety-critical systems during the project will be fed back to the Expert Group. The SCJ standard itself builds on preliminary work on Java for hard real-time systems [21, 16]. Further profiles for safety-critical Java have been presented by the authors [29, 24, 30, 3].

The Java Processor JOP

JOP (Java Optimized Processor) implements the Java virtual machine (JVM) in hardware [25]. JOP is designed from ground up to provide time-deterministic execution of Java programs. In contrast to other JVM implementations, the execution time of Java bytecodes can be predicted cycle accurate. JOP is an enabling technology for worst-case execution time (WCET) analysis [28, 27, 11, 4] of Java programs, a crucial step in building high integrity real-time systems.

Analyses of Java Programs

Model checking and static analysis are two of the most important and widely used approaches to automated analysis of software and have been used to analyze programs at all levels: from machine code to high level languages, including Java and Java bytecode programs. Both have been used to

verify a wide range of properties, including WCET and cache analysis [6, 13], schedulability analysis [4, 7, 14], non-interference [10, 9], quantitative analysis and verification [17, 8], etc., that are highly relevant and important to safety-critical systems. The restricted nature of SCJ, as compared to standard Java, is likely to enable more precise analyses of an even wider range of properties. We believe that the combination of model checking and static analysis will enable whole new classes of tools to be designed and built.

Related Projects

We know that the research group, led by Prof. Jan Vitek, at the University of Purdue is working on an implementation of Level 0 on top of the Ovm [1] and the Fiji [18] JVM. Their implementation of the core library will be open-source. We are already cooperating with this group on the SCJ implementation and plan a research visit within this project.

Aicas[2] implements the reference implementation (RI) of SCJ on top of their RTSJ [5] based JamaicaVM. As the the RI is part of the specification we have access to this source base as well. An SCJ implementation on top of the RTSJ inherits the full complexity of the RTSJ implementation – something the SCJ specification intends to avoid. Therefore, this type of implementation is not intended to be certified, but for prototyping of SCJ applications on top of a standard RTSJ JVM.

Research Plan

The project has two major work areas: Making JOP ready for safety-critical Java and implementing a development environment with analysis tools.

Safety-Critical Java on a Time-predictable Java Processor (DTU)

We will implement the SCJ standard (levels 0 and 1) on top of JOP. The implementation will cover a uniprocessor version of JOP and a chip-multiprocessor (CMP) verion. Device drivers for low-level I/O access, written entirely in Java, will be supported by our proposed hardware objects [26]. Hardware support in JOP for time critical operations, such as scope checks and ceiling locks on CMP systems, will be investigated. We will also investigate real-time alternative libraries for common idioms in Java (e.g., collection classes) that are not designed for real-time systems.

Development Environment (AAU, DTU)

In a development process, the planned workbench enters at the level of detailed design and adaptation to the platform, thus we assume that some object-oriented analysis and design activity has taken place

[2] `http://www.aicas.com/`

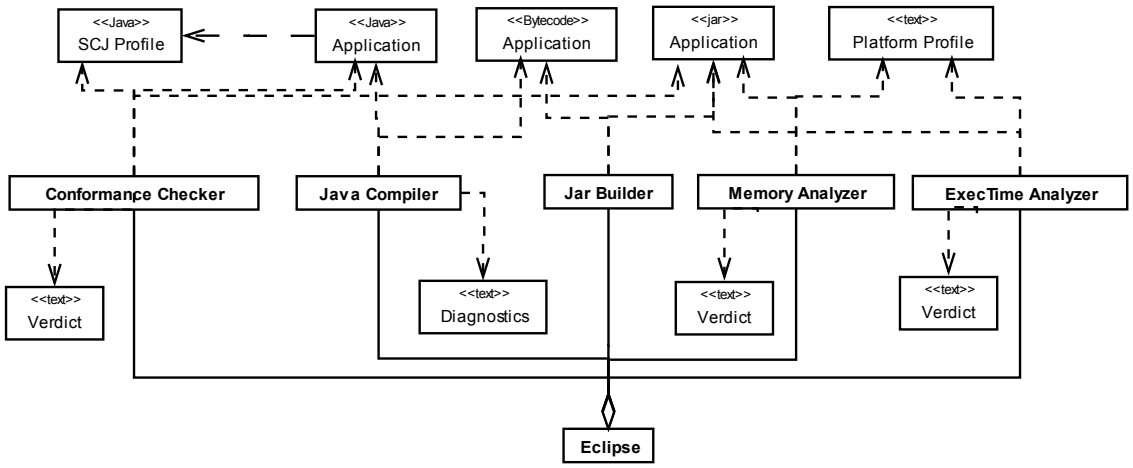

Figure 1: The proposed workbench, integrated into Eclipse

beforehand; perhaps in a model based engineering tool. We assume that the result of that design activity is a set of Java declarations that conforms to a generic architecture consisting of *sporadic* and *periodic event handlers* collected in *missions*. Each mission initializes a set of objects which are shared by its handlers, and each handler has an event handling method that may declare method local objects. Missions may be initiated sequentially. This architecture corresponds to the expected SCJ-profile. The architecture allows to carry out static analyzes of resource utilization for an application.[3] In more detail, the workbench consists of a number of applications that are built as *Eclipse*[4] plug-ins, see Figure 1:[5]

- A Java Profile that defines and delimits the architecture of the application
- A Conformance checker that extracts an abstract version of the application and checks it for conformance with the profile and analyzes it for potential deadlocks, dead code, and uncaught exceptions.
- A standard Java compiler and load (jar) builder.
- A JVM bytecode analyzer for memory requirements for stacks and objects. It works on the load module, and it gives either conservative estimates for the consumption or a verdict that the application is not amenable to analysis.
- A corresponding JVM bytecode analyzer for execution time properties, including blocking

[3]Feasible means that well-behaved programs as found in example embedded systems are analyzable. General programs are not analyzable. We do not pretend to have a sound solution to the Halting Problem.

[4]`http://www.eclipse.org/`

[5]This UML class diagram shows the program classes aggregated under Eclipse and their use (dependency on) of different classes of Java related objects and verdicts for certification purposes.

times. The pessimism of its estimates are dependent on the platform model that is a module in the tool.

Very important for the correct functioning of safety-critical applications are their timing constraints. The correctness depends not only on the results of the computation, but also on the physical instant when the results are produced. Today, many designers still rely on simulation and/or measurements to determine if the various constraints of an embedded system are satisfied. However, simulations and measurement provide no guarantees that the imposed requirements are met. There is a large quantity of research related to scheduling and schedulability analysis [20, 19], with results having been incorporated in tools such as SymTA/S developed at TU Braunschweig and Symtavision, MAST at the University of Cantabria, Spain, and UPPAAL [2] at Aalborg University in Denmark and Uppsala University, Sweden. The state-of-the-art schedulability techniques employed by these tools will be adapted to support the proposed safety-critical Java platform.

Evaluation

The whole system will be evaluated by use cases provided by GomSpace. The programs will be implemented against the SCJ profile and the analysis tools have to provide tight bounds on the resource consumptions.

Dissemination and Publication Schedule

Scientific results will be published and presented at international conferences and in relevant scientific journals. One PhD theses will publish the results from the project. The publication schedule is as follows:

JTRES 2011, Esweek 2011 Prototype of SCJ on JOP, Device drivers in Java

JTRES 2012, Esweek 2012 Presentation of the analysis tool chain

JTRES 2013 Evaluation study of a SCJ application

RTSS 2013 Feasibility of Java for safety-critical systems

2013 Submission of an article to Journal of Systems Architecture (Elsevier)

In the middle of the project and at the end of the project we will organize a workshop for the industry to present the feasibility of safety-critical Java. Furthermore, the results, raw data, and sources will be published on a project web site. It is the intention to keep the project open-source under the GNU GPL. Open-source research projects attract other researchers to use and build on the results of the project.

References

[1] Austin Armbruster, Jason Baker, Antonio Cunei, Chapman Flack, David Holmes, Filip Pizlo, Edward Pla, Marek Prochazka, and Jan Vitek. A real-time Java virtual machine with applications in avionics. *Trans. on Embedded Computing Sys.*, 7(1):1–49, 2007.

[2] Gerd Behrmann, Alexandre David, and Kim G. Larsen. A tutorial on UPPAAL. In Marco Bernardo and Flavio Corradini, editors, *Formal Methods for the Design of Real-Time Systems: 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM-RT 2004*, number 3185 in LNCS, pages 200–236. Springer–Verlag, September 2004.

[3] Thomas Bøgholm, René R. Hansen, Anders P. Ravn, Bent Thomsen, and Hans Søndergaard. A predictable java profile: rationale and implementations. In *JTRES '09: Proceedings of the 7th International Workshop on Java Technologies for Real-Time and Embedded Systems*, pages 150–159, New York, NY, USA, 2009. ACM.

[4] Thomas Bogholm, Henrik Kragh-Hansen, Petur Olsen, Bent Thomsen, and Kim G. Larsen. Model-based schedulability analysis of safety critical hard real-time Java programs. In *Proceedings of the 6th international workshop on Java technologies for real-time and embedded systems (JTRES 2008)*, pages 106–114, New York, NY, USA, 2008. ACM.

[5] Greg Bollella, James Gosling, Benjamin Brosgol, Peter Dibble, Steve Furr, and Mark Turnbull. *The Real-Time Specification for Java*. Java Series. Addison-Wesley, June 2000.

[6] Andreas Dalsgaard, Mads Chr. Olesen, Martin Toft, René Rydhof Hansen, and Kim G. Larsen. WCET analysis of ARM processors using real-time model checking. In *Proceedings of Doctoral Symposium on Systems Software Verification (DS SSV'09), Real Software, Real Problems, Real Solutions*, 2009.

[7] Alexandre David, Jacob Illum, Kim G. Larsen, and Arne Skou. *Model-based Framework for Schedulability Analysis using UPPAAL 4.1*, chapter 1. CRC Press, 2009.

[8] Uli Fahrenberg, Kim Guldstrand Larsen, and Claus Thrane. Verification, performance analysis and controller synthesis for real-time systems. In *Proceedings of 3rd International Conference on Fundamentals of Software Engineering (FSEN 09)*, 2009. To appear.

[9] René Rydhof Hansen. *Flow Logic for Language-Based Safety and Security*. PhD thesis, Technical University of Denmark, 2005.

[10] René Rydhof Hansen and Christian W. Probst. Non-Interference and Erasure Policies for JavaCard Bytecode. In *Workshop on Issues in the Theory of Security, WITS'06*, pages 174–189, 2006.

[11] Trevor Harmon and Raymond Klefstad. Interactive back-annotation of worst-case execution time analysis for Java microprocessors. In *Proceedings of the Thirteenth IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2007)*, August 2007.

[12] Thomas Henties, James J. Hunt, Doug Locke, Kelvin Nilsen, Martin Schoeberl, and Jan Vitek. Java for safety-critical applications. In *2nd International Workshop on the Certification of Safety-Critical Software Controlled Systems (SafeCert 2009)*, York, United Kingdom, Mar. 2009.

[13] Benedikt Huber and Martin Schoeberl. Comparison of implicit path enumeration and model checking based WCET analysis. In *Proceedings of the 9th International Workshop on Worst-Case Execution Time (WCET) Analysis*, pages 23–34, Dublin, Ireland, July 2009. OCG.

[14] Jacob Illum, Kim G. Larsen, Marius Mikucionis, and Steen Palm. Model-based approach for schedulability analysis. Deliverable 2010 for Quasimodo Project.

[15] Java Expert Group. Java specification request JSR 302: Safety critical java technology. Available at `http://jcp.org/en/jsr/detail?id=302`.

[16] Jagun Kwon, Andy Wellings, and Steve King. Ravenscar-Java: A high integrity profile for real-time Java. In *Proceedings of the 2002 joint ACM-ISCOPE conference on Java Grande*, pages 131–140. ACM Press, 2002.

[17] Kim Guldstrand Larsen. Quantitative verification and validation of embedded systems. In *Proceedings of of 3rd IEEE International Symposium on Theoretical Aspects of Software Engineering, TASE09*, 2009. To appear.

[18] Filip Pizlo, Lukasz Ziarek, and Jan Vitek. Real time java on resource-constrained platforms with fiji vm. In *JTRES '09: Proceedings of the 7th International Workshop on Java Technologies for Real-Time and Embedded Systems*, pages 110–119, New York, NY, USA, 2009. ACM.

[19] Paul Pop, Petru Eles, Zebo Peng, and Traian Pop. Analysis and optimization of distributed real-time embedded systems. *ACM Transactions on Design Automation of Electronic Systems*, 11(3):593–625, 2006.

[20] Traian Pop, Paul Pop, Petru Eles, and Zebo Peng. Analysis and optimisation of hierarchically scheduled multiprocessor embedded systems. *International Journal of Parallel Programming : Special Issue on Multiprocessor-based Embedded Systems*, 36(1):37–67, 2008.

[21] Peter Puschner and Andy Wellings. A profile for high integrity real-time Java programs. In *4th IEEE International Symposium on Object-oriented Real-time distributed Computing (ISORC)*, 2001.

[22] RTCA/DO-178B. Software considerations in airborne systems and equipment certification. December 1992.

[23] Martin Schoeberl. *JOP: A Java Optimized Processor for Embedded Real-Time Systems*. PhD thesis, Vienna University of Technology, 2005.

[24] Martin Schoeberl. Mission modes for safety critical Java. In *Software Technologies for Embedded and Ubiquitous Systems, 5th IFIP WG 10.2 International Workshop (SEUS 2007)*, volume 4761 of *Lecture Notes in Computer Science*, pages 105–113. Springer, May 2007.

[25] Martin Schoeberl. A Java processor architecture for embedded real-time systems. *Journal of Systems Architecture*, 54/1–2:265–286, 2008.

[26] Martin Schoeberl, Stephan Korsholm, Tomas Kalibera, and Anders P. Ravn. A hardware abstraction layer in Java. *Trans. on Embedded Computing Sys.*, accepted, 2010.

[27] Martin Schoeberl and Rasmus Pedersen. WCET analysis for a Java processor. In *Proceedings of the 4th International Workshop on Java Technologies for Real-time and Embedded Systems (JTRES 2006)*, pages 202–211, New York, NY, USA, 2006. ACM Press.

[28] Martin Schoeberl, Wolfgang Puffitsch, Rasmus Ulslev Pedersen, and Benedikt Huber. Worst-case execution time analysis for a Java processor. *Software: Practice and Experience*, accepted for publication, 2010.

[29] Martin Schoeberl, Hans Sondergaard, Bent Thomsen, and Anders P. Ravn. A profile for safety critical Java. In *10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC'07)*, pages 94–101, Santorini Island, Greece, May 2007. IEEE Computer Society.

[30] Martin Schoeberl and Jan Vitek. Garbage collection for safety critical Java. In *Proceedings of the 5th International Workshop on Java Technologies for Real-time and Embedded Systems (JTRES 2007)*, pages 85–93, Vienna, Austria, September 2007. ACM Press.

[31] Jean-Loup Terraillon. European space technology harmonisation technical dossier, on-board software. Technical report, European Space Agency, 2009.