Safety-critical Java for embedded systems

This paper presents the motivation for and outcomes of an engineering research project on certifiable Java for embedded systems. The project supports the upcoming standard for safety-critical Java, which defines a subset of Java and libraries aiming for development of high criticality systems. The outcome of this project includes prototype safety-critical Java implementations, a time-predictable Java processor, analysis tools for memory safety, and example applications to explore the usability of safety-critical Java for this application area. The text summarizes developments and key contributions and concludes with the lessons learned.

Copyright © 2016 John Wiley & Sons, Ltd.
An evaluation of safety-critical Java on a Java processor

The safety-critical Java (SCJ) specification provides a restricted set of the Java language intended for applications that require certification. In order to test the specification, implementations are emerging and the need to evaluate those implementations in a systematic way is becoming important. In this paper we evaluate our SCJ implementation which is based on the Java Optimized Processor JOP and we measure different performance and timeliness criteria relevant to hard real-time systems. Our implementation targets Level 0 and Level1 of the specification and to test it we use a series of micro benchmarks, an application-based benchmark, and a reduced set of a SCJ technology compatibility kit. We evaluate the accuracy of periods, linear-time memory allocation, aperiodic event handling, dispatch latency for interrupts, context switch preemption latency, and synchronization.

General information
State: Published
Organisations: Department of Applied Mathematics and Computer Science, Embedded Systems Engineering
Authors: Rios Rivas, J. R. (Intern), Schoeberl, M. (Intern)
Pages: 276-283
Publication date: 2014

Host publication information
Title of host publication: 2014 IEEE 17th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC)
Publisher: IEEE
Editor: O'Connor, L.
ISBN (Print): 978-1-4799-4430-9
Certifiable Java for Embedded Systems

The Certifiable Java for Embedded Systems (CJ4ES) project aimed 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 help to provide evidence that can be used as part of a safety case. This paper summarizes key contributions within these areas during the three-year project period. In the conclusion the overall result of the project is assessed.

General information

State: Published
Organisations: Department of Applied Mathematics and Computer Science, Embedded Systems Engineering, Aalborg University, VIA University College
Authors: Schoeberl, M. (Intern), Dalsgaard, A. E. (Forskerdatabase), Hansen, R. R. (Ekstern), Korsholm, S. E. (Ekstern), Ravn, A. P. (Ekstern), Rios Rivas, J. R. (Intern), Strøm, T. B. (Intern), Søndergaard, H. (Ekstern)
Pages: 10-19
Publication date: 2014

Host publication information

Title of host publication: Proceedings of the 12th International Workshop on Java Technologies for Real-time and Embedded Systems (JTRES 2014)
Publisher: Association for Computing Machinery
ISBN (Print): 978-1-4503-2813-5
BFI conference series: International workshop on Java technologies for real-time and embedded systems (5000304)
Main Research Area: Technical/natural sciences
Safety-critical Java
DOIs: 10.1145/2661020.2661025
Source: PublicationPreSubmission
Source-ID: 110628663
Publication: Research - peer-review › Article in proceedings – Annual report year: 2015

Reusable libraries for safety-critical Java

The large collection of Java class libraries is a main factor of the success of Java. However, these libraries assume that a garbage-collected heap is used. Safety-critical Java uses scope-based memory areas instead of a garbage-collected heap. Therefore, the Java class libraries are problematic to use in safety-critical Java. We have identified common programming patterns in the Java class libraries that make them unsuitable for safety-critical Java. We propose ways to improve the libraries to avoid the impact of the identified problematic patterns. We illustrate these changes by implementing a total of five scope-safe classes from commonly used libraries.
Safety-Critical Java for Embedded Systems

Safety-critical systems are real-time systems whose failure can have severe or catastrophic consequences, possibly endangering human life. Many safety-critical systems incorporate embedded computers used to control different tasks. Software running on safety-critical systems needs to be certified before its deployment and the most time-consuming step of this process is the testing and verification phase. Due to the increasing complexity in safety-critical systems there is a need for new technologies that can facilitate testing and verification activities. The safety-critical specification for Java aims at providing a reduced set of the Java programming language that can be used for systems that need to be certified at the highest levels of criticality. Safety-critical Java (SCJ) restricts how a developer can structure an application by providing a specific programming model and by restricting the set of methods and libraries that can be used. Furthermore, its memory model do not use a garbage-collected heap but scoped memories.

In this thesis we examine the use of the SCJ specification through an implementation in a time-predictable, FPGA-based Java processor. The specification is now in a mature state and with our implementation we have proved its feasibility in an embedded platform. Moreover, we have explored how simple hardware extensions can reduce the execution time of time-critical operations required by the SCJ specification.

The scoped memory model used in SCJ is perhaps one of its most difficult features to use correctly. Therefore, in this work we have also studied practical aspects of its usage by developing scoped memory use patterns and reusable libraries aiming at facilitating the development of complex software systems.
Hardware Support for Safety-critical Java Scope Checks

Memory management in Safety-Critical Java (SCJ) is based on time bounded, non garbage collected scoped memory regions used to store temporary objects. Scoped memory regions may have different life times during the execution of a program and hence, to avoid leaving dangling pointers, it is necessary to check that reference assignments are performed only from objects in shorter lived scopes to objects in longer lived scopes (or between objects in the same scoped memory area). SCJ offers, compared to the RTSJ, a simplified memory model where only the immortal and mission memory scoped areas are shared between threads and any other scoped region is thread private. In this paper we present how, due to this simplified model, a single scope nesting level can be used to check the legality of every reference assignment. We also show that with simple hardware extensions a processor can see some improvement in terms of execution time for applications where cross-scope references are frequent. Our proposal was implemented and tested on the Java Optimized Processor (JOP).

General information
State: Published
Organisations: Embedded Systems Engineering, Department of Informatics and Mathematical Modeling
Authors: Rios Rivas, J. R. (Intern), Schoeberl, M. (Intern)
Pages: 31-38
Publication date: 2012
Safety-critical Java on a Java processor
The safety-critical Java (SCJ) specification is developed within the Java Community Process under specification request number JSR 302. The specification is available as public draft, but details are still discussed by the expert group. In this stage of the specification we need prototype implementations of SCJ and first test applications that are written with SCJ, even when the specification is not finalized. The feedback from those prototype implementations is needed for final decisions. To help the SCJ expert group, a prototype implementation of SCJ on top of the Java optimized processor is developed and presented in this paper. This implementation raises issues in the SCJ specification and provides feedback to the expert group.

General information
State: Published
Organisations: Department of Informatics and Mathematical Modeling, Embedded Systems Engineering
Authors: Schoeberl, M. (Intern), Rios Rivas, J. R. (Intern)
Pages: 54-61
Publication date: 2012

Host publication information
Title of host publication: Proceedings of the 10th International Workshop on Java Technologies for Real-time and Embedded Systems
Publisher: Association for Computing Machinery
ISBN (Print): 978-1-4503-1688-0
BFI conference series: International workshop on Java technologies for real-time and embedded systems (5000304)
Main Research Area: Technical/natural sciences
Workshop: 10th International Workshop on Java Technologies for Real-time and Embedded Systems (JTRES 2012), Copenhagen, Denmark, 24/10/2012 - 24/10/2012
DOIs: 10.1145/2388936.2388946
Publication: Research - peer-review › Article in proceedings – Annual report year: 2012

Projects:
Certifiable Java for Embedded Systems
Department of Applied Mathematics and Computer Science
Period: 15/03/2011 → 25/09/2014
Number of participants: 6
Phd Student:
Rios Rivas, Juan Ricardo (Intern)
Supervisor:
Pop, Paul (Intern)
Main Supervisor:
Schoeberl, Martin (Intern)
Examiner:
Dragoni, Nicola (Intern)
Puschner, Peter (Ekstern)
Uhrig, Sascha (Ekstern)

Financing sources
Source: Internal funding (public)
Name of research programme: Forskningsrådsfinansiering
Project: PhD