Special Issue on 12th International Workshop on Java Technologies for Real-Time and Embedded Systems JTRES2014

Geoffrey Fox, School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN, 47401

This editorial describes a special issue of papers from the 2014 workshop on Java Technologies for Real-Time and Embedded Systems [1]. There are 4 papers in this special issue.

The first paper [2] presents HVM_{TP}, a time predictable and portable Java virtual machine (JVM) implementation with applications in resource-constrained, hard real-time embedded systems, which implements all levels of the safety-critical Java (SCJ) specification. Time predictability is achieved by a combination of time-predictable algorithms, exploiting the programming model of the SCJ profile and harnessing the static knowledge of the hosted SCJ system. This paper presents HVM_{TP} in terms of its design and capabilities and demonstrates how a complete timing model of the JVM represented as a network of timed automata can be obtained using the tool $TETASARTS_{JVM}$. The timing model readily integrates with the rest of the TETASARTS toolset for temporal verification of SCJ systems. A complete timing scheme in terms of safe worst-case execution times and best-case execution times of the Java bytecodes is derived from the model. The paper takes a first look at how to support the new Java 8 language feature of Lambda expressions in an SCJ context – looking in particular at how the bytecode, used for invoking Java 8 closures, can be implemented in a time-predictable way and integrated into HVM_{TP}.

The second paper [3] presents the motivation for and outcomes of an engineering research project on certifiable Java for embedded systems. The project supports the upcoming SCJ standard for safety-critical Java, which defines a subset of Java and libraries aiming for the 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.

The third paper [4] presents a model-based development of a Technology Compatibility Kit test suite and a test execution tool for the draft safety-critical Java profile specification. Passing all tests in an associated Technology Compatibility Kit is needed in order for a Java implementation to claim conformance with a Java Specification Request,.The Java Modeling Language is used to model conformance constraints for the profile. Java Modeling Language annotations define contracts for classes and interfaces. The annotations are translated by a tool into runtime assertion checks. Hereby, the design and elaboration of the concrete test cases are simplified, because the expected results are derived from contracts and thus do not need to be provided explicitly. Bottom-up testing is applied for testing methods of the safety-critical Java classes, whereas top-down testing is applied for testing global properties, such as protocols, memory

management, and real-time properties, including scheduling. The tests are executed using a simplified version of JUnit, which makes the test suite executable on resource-constrained platforms.

The last paper [5] shows the weak predictability of Android's internal intra-process and inter-process communication based on Intent messaging and presents a concept to improve its soft real-time capability. The proposed approach introduces a priority-based broadcast handling using explicit priority values, instead of the original first in – first out processing. Furthermore, the size of responsible critical sections is reduced in order to improve the preemptibility and to assure predictable processing times for applications with real-time requirements. Their evaluation highlights the improvements in comparison with the original Android implementation without any loss of the system performance. Additionally, the compatibility with already existing components and applications is preserved.

We thank Wolfgang Puffitsch for his work on this special issue.

References

1. The 12th International Workshop on Java Technologies for Real-time and Embedded Systems - JTRES 2014, October 13th - 14th, Niagara Falls, NY, USA http://jtres2014.compute.dtu.dk/
2. Kasper Søe Luckow, Bent Thomsen,, Stephan Erbs Korsholm, "HVM_{TP}: A time predictable and portable java virtual machine for hard real-time embedded systems", Concurrency and Computation: Practice and Experience [this issue]. DOI 10.1002/cpe.3828
3. Martin Schoeberl, Andreas Engelbredt Dalsgaard, René Rydhof Hansen, Stephan E. Korsholm, Anders P. Ravn, Juan Ricardo Rios Rivas, Tórur Biskopstø Strøm, Hans Søndergaard, Andy Wellings, Shuai Zhao, "Safety-critical Java for embedded systems", Concurrency and Computation: Practice and Experience [this issue]. DOI 10.1002/cpe.3963
4. Hans Søndergaard, Stephan E. Korsholm, Anders P. Ravn, "Conformance test development with the Java modeling language", Concurrency and Computation: Practice and Experience [this issue]. DOI 10.1002/cpe.4071
5. Igor Kalkov, Alexandru Gurghian, Stefan Kowalewski, "Explicit prioritization of parallel Intent broadcasts in real-time Android", Concurrency and Computation: Practice and Experience [this issue]. DOI 10.1002/cpe.4122