

# A FIRST STEP TOWARDS USING REAL-TIME JAVA FOR SPACECRAFT ON-BOARD SOFTWARE

Marek Prochazka<sup>(1)</sup>, Roger Ward<sup>(1)</sup>, Andy Wellings<sup>(2)</sup>

<sup>(1)</sup> *SciSys, Clothier Road, Bristol, BS4 5SS, United Kingdom,  
Email: marek.prochazka@scisys.co.uk, roger.ward@scisys.co.uk*

<sup>(2)</sup> *Department of Computer Science, The University of York, York, YO10 5DD, United Kingdom,  
Email: andy.wellings@cs.york.ac.uk*

## ABSTRACT

This paper outlines directions for an evaluation of Real-Time Java suitability to be used in ESA spacecraft on-board software. It starts with an overview of related existing standards and ongoing standardisation efforts, as well as several use cases in the avionics and space domain. Then it identifies steps which should be followed to research how particular Java features match particular requirements of different spacecraft on-board software application domains. The paper takes into account the goals of the upcoming ESA Real-Time Java Assessment Project [1].

## 1. INTRODUCTION

The key motivation to get Java on-board is to use a state-of-the art type-safe programming language with an extensive set of freely-available libraries, excellent support by various development tools, as well as the Java WORA (Write Once Run Anywhere) approach. Various research studies and use cases show that, comparing to other major programming languages, software written in Java usually exhibits less bugs in the code, it is faster for development, safer and more maintainable (e.g. [2], [3], [4], [15]). In this paper we present some of the existing standards, profiles and projects which are relevant for getting Real-Time Java to be used in on-board software. We also outline directions for deeper Java evaluation for different on-board software application domains.

## 2. MAKING JAVA SUITABLE FOR SPACECRAFT ON-BOARD SOFTWARE

Shortly after Java became a widely used programming language, various attempts have been launched to make it suitable for real-time, high-integrity and embedded applications:

- As a part of the Java Specification Request (JSR 1, [5]), real-time capability has been added in the Real-Time Specification for Java (RTSJ) [6], whose key goal was to extend the Java Language Specification and the Java Virtual Machine (JVM) Specification with an API and semantics that enables use of Java threads whose correctness conditions include timeliness constraints, as well as

a variety of related features. RTSJ provides both current-practice (fixed-priority preemptive scheduling) and advanced features (e.g. CPU time monitoring), requires no syntactic extension and allows variation in implementation decisions.

- The J-Consortium in its Real-Time Core (RTCore) Extensions Specification [7] addresses better resource performance and execution predictability at the cost of reducing interoperability with normal Java. RTCore proposes real-time Java programs to execute on a Core engine, independently of the normal JVM, whilst allowing those with less stringent integrity and predictability requirements to use typical JVM services such as dynamic loading and APIs via controlled interaction with the JVM.

More recently, some attempts have been made to make (Real-Time) Java suitable for mission-critical and safety-critical systems:

- JSR-302 [8] aims at providing a specification based on the Real-Time Specification for Java, containing minimal features necessary for safety critical systems capable of certification, e.g., DO-178B.
- Ravenscar Java: The Ravenscar profile of the RTSJ [9] in essence removes all non-deterministic features from the original RTSJ.
- The HIJA Project [10] aimed at developing a JVM implementing a profile of RTSJ and contributing to standards development activities for safety-critical Java, mission-critical Java, and the ARINC 653 extended services of the aerospace safety standards. The project has identified three separate Real-Time Java profiles [11],
  - o hard real-time Java profile, which follows the Ravenscar Java approach,
  - o soft real-time Java profile for business-critical applications,
  - o flexible real-time Java profile providing execution environment for multiple independent applications, isolated from each other with respect to resource management, quality of service and execution behaviour.
- Aonix have been working on guidelines for development hard real-time, soft real-time and safety critical applications in Java and corresponding Java profiles [12]. As a part of this

work, a type system has been defined which makes the use of the RTSJ scope memory safe with respect to runtime exceptions [13].

### 3. USE CASES IN SPACE AND AVIONICS SYSTEMS

Java is increasingly used for avionics and space applications. As early as in 2001-2003 the NASA Jet Propulsion Laboratory, Sun Microsystems and Carnegie Mellon University participated on the Golden Gate Project which tested the RTSJ capability for controlling a Mars rover [14]. Various projects have examined the RTSJ capability to be used for avionics and safety-critical applications ([15], [16]).

As a part of the DARPA PCES project, Boeing and Purdue University demonstrated that a RTSJ implementation called Open Virtual Machine (OVM) could serve as an execution environment for autonomous navigation capabilities on the ScanEagle Unmanned Aerial Vehicle (UAV) ([17], [18]). As the PCES project finished with a live demonstration in the White Sands Missile Range, New Mexico, USA on 14 April 2005, this milestone marked the first flight using the RTSJ and received a Java 2005 Duke's Choice Award for innovation in Java technology.

In 2006, JamaicaVM by aicas followed the success of OVM and was used on the Barracuda UAV by EADS [19]. JamaicaVM was also selected as a platform for communication of the Boeing 787 Dreamliner with ground stations.

In October 2006, Lockheed Martin has selected the Aonix PERC Ultra VM for the Aegis Weapon System Open Architecture Program which aims at enhancing the capabilities and service life of the U.S. Navy's premier surface combat system while also reducing its cost [20].

The options for using the Java Virtual Machine as the control environment for execution of the on-board control procedures (OBCP) have been explored in several studies (e.g. [21], [22]). This work promises an interesting (though very specialised) application of Java in on-board software.

### 4. EVALUATION OF REAL-TIME JAVA CAPABILITY FOR SPACE APPLICATIONS

Java has become a popular option as both a general software development platform and also a platform for development of real-time and embedded systems [23]. An effort therefore has to be invested into an evaluation of both the existing Java standards and various profiles as well as available commercial and research products as to how these meet requirements for development of spacecraft on-board software. Some of the key directions of this effort are as follows:

- First of all, analyse the current and future trends of avionics and space software systems architecture,

design, development and validation, identify application domains and their specific requirements for programming languages, execution platforms, methodologies, real-time computing primitives, safety guarantees, etc. Also, this study should also focus on why these trends eventually cannot be met in C, C++ or Ada.

- After that, study potential use of different (Real-Time) Java standards, profiles and products in future satellite and exploration probe systems, including selection of appropriate target applications.
- Analyse properties of the selected commercial or research (Real-Time) Java implementations, such as robustness, performance, interpreted execution or ahead-of-time compilation.
- As this seems to be a very specific but interesting application of Real-Time, study in detail its ability to serve as the execution environment (interpreter) of on-board control procedures (OBCP).
- Study potential use of Java as the execution environment for the whole on-board software system.
- Analyse options for integration of Java with legacy systems written in C, C++ or Ada.

Note that the first identified task is by far the most important. Current and future trends of avionics in spacecraft on-board software systems in different application domains must play the key role in evaluating the Java ability to be used for these systems. Java features which play an important role for particular on-board space applications must be identified, e.g. interpreted execution with dynamic class-loading for the execution control of OBCPs. Based on this feature identification, suggestions could be provided for the domain usability. Here are some features of Real-Time Java which must be studied in particular:

- Timing features. The ability to fulfil timing constraints. Available tools supporting static analysis as well as run-time monitoring facilities for soft-real time or validation purposes. Means for avoiding priority inversions.
- Memory management. Automatic vs. explicit memory management, garbage collecting, scope memory, immortal memory, other options. Tools for ensuring safe use of the available memory models. Use of standard Java libraries with non-standard memory models.
- Partitioning. The ability to ensure CPU and memory budgets of Java applications running on the same processor. Both static and dynamic approaches. Also take into account the Application Isolation API Specification (Java isolates, [24]), which allows multiple Java programs to share the same JVM, and consider its real-time extension.
- Integration with hardware. The ability to use physical memory, access standalone hardware

drivers and equipments, signals, asynchronous and non-blocking I/O. The ability to ensure timeliness and good time/space partitioning between software and hardware-related tasks.

- Execution. Ahead-of-time compilation, just-in-time compilation, interpreting, mixed execution. Dynamic class loading. Tools for determining and ensuring transitive closure of classes used in the application.
- Performance. Comparison with other languages using simple benchmarks. Pure execution performance, predictability of key primitives, memory utilisation, footprint and fragmentation.
- Failure management. Resistance to certain types of failures, ease of restart, support for warm restart, support for remote observability and commandability.
- Interoperability. JNI implementation or other means to support interoperability with legacy applications written in C or Ada.
- Qualification. The status and future options for verified execution and certification of Java VM.
- Tools. Static analysis, memory utilisation, profiling, schedulability analysis, code generation (e.g. from AADL, UML).
- Quality assurance. Development tools, documentation, compliance with quality standards.

## 5. CONCLUSIONS

This paper has outlined directions for a deep evaluation of Real-Time Java's suitability to be used in spacecraft on-board software. It has started with an overview of related existing standards and ongoing standardisation efforts, as well as use cases in avionics and space domain. It has then identified steps which should be followed to research how particular Java features match particular requirements of different spacecraft on-board software application domains.

## REFERENCES

1. The Real-Time Java Assessment Project, ESA RFQ/3-11880/06/NL/JD, 2007.
2. Patricia K. Lawlis, "Guidelines for Choosing a Computer Language: Support for the Visionary Organization", 2nd Edition, C.J. Kemp Systems, Inc., August 1997.
3. Geoffrey Phipps, "Comparing Observed Bug and Productivity Rates for Java and C++", *Software Practice & Experience*, Vol. 29, No. 4, p. 345-358, April 10, 1999.
4. Gary McGraw, Greg Morrisett, "Attacking Malicious Code: A Report to the Infosec Research Council", *IEEE Software*, Vol. 17, No. 5, September 2000.
5. Java Specification Requests, JSR 1: Real-time Specification for Java, <http://jcp.org/en/jsr/detail?id=1>, 1998.
6. Greg Bollella, James Gosling, Benjamin Brosgol, Peter Dibble, Steve Furr, and Mark Turnbull, "The Real-Time Specification for Java", Version 1.0.2, Addison-Wesley, 2000.
7. J-Consortium, "Real-Time Core Extensions", Version 1.0.14, 2000.
8. Java Specification Requests, JSR 302: Safety Critical Java Technology, <http://jcp.org/en/jsr/detail?id=302>, 2004.
9. Jagun Kwon, Andy Wellings, and Steve King, "Ravenscar-Java: A High Integrity Profile for Real-Time Java", Department of Computer Science, University of York, Technical Report YCS 342, May 2002.
10. HIJA: High-Integrity Java Applications, <https://www.hija.info/>.
11. Antonio Kung, Scott Hansen, "ANRTS Platforms", the 4th International Workshop on Java Technologies for Real-time and Embedded Systems (JTRES 2006), Paris, France, September 2006.
12. Kelvin Nilsen, "Guidelines for Scalable Java Development of Real-Time Systems", Aonix, <http://research.aonix.com/jsc/rtjava.guidelines.3-28-06.pdf>, March 28, 2006.
13. Kelvin Nilsen, "A Type System to Assure Scope Safety within Safety Critical Java Modules", the 4th International Workshop on Java Technologies for Real-time and Embedded Systems (JTRES 2006), Paris, France, September 2006.
14. Daniel Dvorak, Gregory Bollella, Tim Canham, Vanessa Carson, Virgil Champlin, Brian Giovannoni, Mark Indictor, Kenny Meyer, Alex Murray, Kirk Reinholtz, "Project Golden Gate: Towards Real-Time Java in Space Missions", 7th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2004), Viena, Austria, May 2004.
15. Edward G. Benowitz, Albert F. Niessner, "Experiences in Adopting Real-Time Java for Flight-like Software", 1st International Workshop on Java Technologies for Real-time and Embedded Systems (JTRES 2003), Catania, Italy, November 2003.
16. David Sharp, "Real-Time Distributed Object Computing: Ready for Mission-Critical Embedded System Applications", 3rd International Symposium on Distributed Objects and Applications (DOA 2001), Rome, Italy, September 2001.
17. Jason Baker, Antonio Cunei, Chapman Flack, Filip Pizlo, Marek Prochazka, Jan Vitek, Austin Armbuster, Edward Pla, David Holmes, "A Real-time Java Virtual Machine for Avionics, An Experience Report", 12th IEEE Real-Time and Embedded Technology and Applications

Symposium (RTAS 2006), San Jose, California, USA, April 2006.

18. A. Armbruster, J. Baker, A. Cunei, C. Flack, D. Holmes, F. Pizlo, E. Pla, M. Prochazka, J. Vitek, "A Real-Time Java Virtual Machine with Applications in Avionics", ACM Transactions on Embedded Computing Systems (TECS), to appear.
19. aicas, JamaicaVM, <http://aicas.com/jamaica.html>.
20. Aonix,PERC, <http://www.aonix.co.uk/perc.html>.
21. G. Garcia, C. Roubion, and S. Prunier, "Java as a standardized on-board control procedures platform?", Data Systems in Aerospace (DASIA 2004), Nice, France, June 2004.
22. Alain Rossignol, Yves Mulet, Bernard Polle, Luc Planche, "OBCP's – From interpreted programs to a standardised command control component", Data Systems in Aerospace (DASIA 2006), Berlin, Germany, May 2006.
23. Yaofei Chen, Dios, R., Mili, A., Lan Wu, Kefei Wang, "An empirical study of programming language trends", IEEE Software, Vol. 22, Issue 3, pp 72 – 79, May-June 2005.
24. Java Specification Requests, JSR 121: Application Isolation API Specification, <http://jcp.org/en/jsr/detail?id=121>, 2001.