HIDOORS - A high integrity distributed deterministic Java environment

João Ventura
Skysoft Portugal
jcv@skysoft.pt

Fridtjof Siebert
aicas GmbH
siebert@aicas.com

Andy Walter
aicas GmbH
anwalt@aicas.com

James Hunt
Forschungszentrum Informatik
jjh@fzi.de

Abstract

This paper presents the design of HIDOORS, an integrated development environment suitable for embedded distributed real-time systems, based on the Java programming language. HIDOORS will cover all the life-cycle of real-time software development with extensions to existing tools (UML modeling, Java compiler, Java Virtual Machine, and a worst case execution time analysis tool) that will all be integrated into a single integrated development environment. The system will also assist the developer in distributing the application, by providing faster RMI and a distributed event manager that provides strict timing guarantees. This paper is written at the beginning of HIDOORS development, and as such, it presents only the defined objectives and the early architecture of the system; further developments will be the subject of future works.

1 Introduction

The continued rapid development of integrated circuit technology both in increased performance and reduced cost has made it economically feasible to build microprocessors into virtually every conceivable device. This is driving a massive shift from desktop applications to embedded systems.

Not only are many traditional IT systems moving from visible desktop computers to invisible embedded computers in intelligent devices, but new applications are constantly being found for intelligent embedded systems. Industrial automation is becoming increasingly decentralized, relying on distributed embedded devices to acquire and pre-process data and run increasingly sophisticated application programs for control and self-diagnostics. New application domains for embedded controls are being created in automotive, avionics, medical equipment, surveillance, domestic appliance and home entertainment industries. On top of this, mobile communication systems are riding this wave into everyone's pocket. Next generation mobile communication systems will expand the possibilities further.

What many of these applications have in common is that they interact with their environment in real-time and that they need to work reliably and predictably. Increasingly, these systems are employed in safety critical environments, where human life and health depends on the proper functioning of computer devices. Thus, the safety and correctness of these systems are of utmost importance.

Current technologies to build reliable and secure software are inadequate. This is especially critical when the industry demand for qualified programmers far exceeds the increase in supply. Only by making programming embedded systems easier and safer can one hope to raise the productivity of programmers sufficiently to fill this gap. These problems are exacerbated when building distributed applications. A very important step in this di-

rection would be the use of a standard object-oriented programming language that provides features like automatic memory management, protected pointer manipulation, and strong typing.

The Java programming language fills these criteria. It is quickly expanding into areas well beyond its original scope. Java is already finding its way into real-time critical applications, often in small embedded systems. The driving force behind this movement is the enormous advantage that the use of this modern programming tool brings to software development in terms of increased safety and productivity.

Unfortunately, Java cannot yet live up to the full requirements of embedded, real-time systems. So far, Java system providers have not paid much attention to the embedded systems market. Consequently, there are no truly real-time Java implementations that provide suitable interfaces for this market. From a technical point of view, there are several obstacles to the deployment of Java in embedded, real-time systems.

Several key scientific issues need to be solved before Java can be used for safety critical real-time systems:

- predictability (determinism) in the context of automatic memory management, multi-threading, and multiprocessing;
- providing predictable real-time networking and distributed event handling;
- the provision of adequate modeling and analysis tools for Java-based, distributed, real-time systems;
- adaptation to limited system resources (CPU power, memory, etc.).

The goal of HIDOORS is to solve these problems by providing the full functionality of the modern Java programming language for the development of distributed, real-time and safety critical systems and to provide a powerful environment of tools that support modeling, analysis, and proof of correctness of systems developed in Java (see Figure 1).

This paper is organized as follows: section 2 presents the objectives of HIDOORS and their relation to the current state-of-the-art, section 3 presents the architecture of the system and section 4 concludes the paper.

Vendor	Product	Real-time Support	Native Code Support	RT guarantees in Native Code
HIDOORS	HIDOORS	yes	yes	yes
Esmertec	JBed	limited	no	no
HP	ChaiVM	limited	yes	no
Sun	KJava	no	no	no
Charis	pVM	no	yes	no
emwerks	KADA	limited	-	-
Transvirtual	Kaffe	no	yes	no
MachJ	MachJ	no	yes	no
Diab	FastJ	no	-	no
Windriver	TurboJ	no	yes	no
Newmonics	PERC	limited	no	no

Table 1: Java Implementations for embedded systems

2 Context

Currently available commercial or free Java development environments focus mainly on desktop systems, but there are also a variety of implementations for embedded systems available. Only a small subset of these systems provide very limited support for software development for real-time systems. None provides distributed, real-time support. Table 1 presents a summary of features of the existing implementations.

There are currently several other research and development programs in place that propagate the use of Java in embedded systems. AJACS[4] is a project with the goal of establishing Java as a base for automotive applications. Another related project is JOSES[6, 2], which aims to harmonize the special hardware and software requirements of high performance embedded systems with the power and virtues of Java.

These projects show the strong interest for Java in the area of embedded systems, but they do not provide a solution for real-time, distributed systems development in Java.

The HIDOORS environment must overcome major obstacles to the use of Java for software development in distributed, real-time and safety-critical applications, if it is to meet its goals.

1. Current approaches for the application of Java for real-time and safety-critical applications focus on sub-setting the language to a more predictable set of operations. Instead, we will develop and implement techniques to enable a predictable implementation of the full Java language. The implementation will support the full range of modern object-oriented

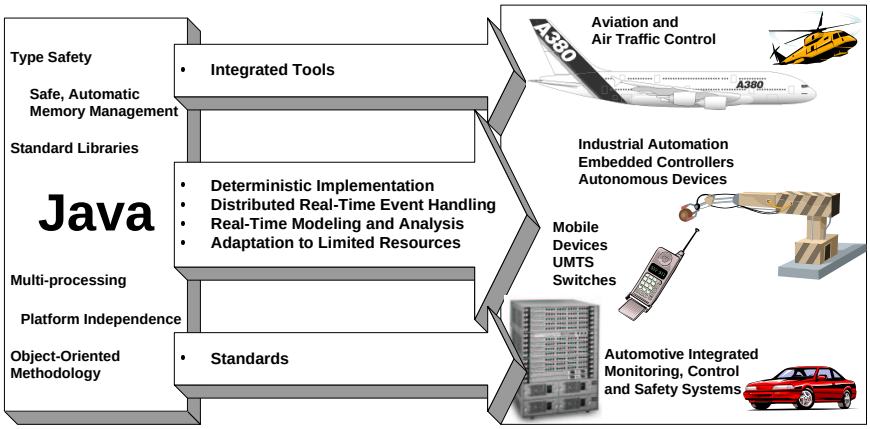

Figure 1: HIDOORS goals

software development for this application domain.

2. Though Java provides Remote Method Invocation (RMI) as a means to communicate between components in a distributed environment, this facility is inadequate for real-time distributed systems. HIDOORS will improve the efficiency and flexibility of RMI to provide for real-time network and distributed event support. This will enable distributed, embedded components to work together in real-time.

3. Currently, no tools exist that support the development and analysis of Java applications in distributed, real-time and safety-critical applications. Tools support must automate as much of the software development process as possible. HIDOORS will include graphical UML-based modeling tools that facilitate the construction and verification of high reliability real-time systems in Java.

4. A major constraint in embedded, real-time system is resource restrictions: limited CPU power and memory are employed while maximum real-time performance and reliability are required. We will focus on improving the memory and runtime performance of the system through the use of new compiler optimization techniques, while maintaining the predictability required for real-time and safety-critical systems.

5. For software developers to accept the resultant technology, the tools need to be integrated in a way to form a consistent, useful product. HIDOORS will integrate the new technologies, supporting tools for software development and analysis, and existing software development tools into a real-time Java integrated development environment.

6. The development of Java for real-time, distributed, embedded systems requires a stronger set of standards than currently available. Since the J-Consortium is one of the organizations most concerned with the issues of using Java for real-time and embedded system, participation in its ongoing standards efforts is paramount to success. HIDOORS will work together with applicable J-Consortium working groups (HIP, RTDA, RTDM, etc.) to shape and demonstrate standards for real-time distributed applications.

The results of HIDOORS will enable the real-time software industry to apply the object-oriented paradigm of Java to the software development process for real-time systems and safety critical systems. HIDOORS will provide the missing technological links that enable the use of modern object-oriented software development methods for real-time distributed applications. Table 2 summarizes the objectives and the current state-of-the-art.

Standardization of the technology is essential for its wide application. HIDOORS will support the develop-

Objective	State-of-the-Art
Deterministic Implementation of full Java Language	Current embedded Java implementation limit language functionality
Real-time network and distributed event support	Standards exist for non real-time systems (CORBA, COM, RMI), but inadequate for real-time systems.
High reliability real-time Java modeling and analysis Tools	Modeling tools do not support time-dependencies or automatic validation.
Improved Java Memory and Runtime performance	Current Java implementation are slow and memory hungry.
Real-time Java Integrated Development Environment	No specialized environment available and/or Java not adequately supported.
Stronger set of standards	Current Java standards for embedded systems are immature and/or inadequate.

Table 2: HIDOORS objectives and current state-of-the-art

ment of these standards by actively contributing to the J-Consortium working groups and providing reference implementation of standards proposed by the J-Consortium.

3 HIDOORS Architecture

The HIDOORS environment will be based on a significant amount of previous work and makes use of an existing code base of prototypes developed by the different members of the team. Figure 2 illustrates the existing parts and the additions that are planned during the development of HIDOORS to extend the code base and integrate the results. The existing code base includes a real-time garbage collector that is integrated in a Java Virtual Machine, a simple static Java byte code to C compiler, a subset of the Java standard classes, profiling tools, and integrated development environment plus UML tools.

3.1 Architecture Components

Several modules are going to be developed for use in HIDOORS. Following is a small description for each of these modules, and its design.

Time Analysis for UML real-time Automata A component which analyses real-time automata on security conditions will be used and integrated in a COTS integrated development environment for UML, in particular into the sub-tool that enables the specification of real-time automata. This work will be based on [7] which proposed a technique to check real-time temporal logic specifications with an abstraction to discrete time automata and so called condition automata.

Optimizing compiler The existing Java compiler will be improved to allow a more predictable garbage collection. On one hand, an optimization of memory management will be performed to minimize the need of the garbage collector, by identifying unnecessary allocations and statically de-allocatable objects and handling them appropriately. Also, the garbage collector itself will be equipped with support for predictable garbage collection, via the use of efficient GC points, constant-time exact root scanning of stacks and registers, write-barrier support and support for the non-fragmenting object model that is employed by the Java implementation. The compiler will also be updated to avoid the intermediate C code generation, by directly generating native code.

Java Virtual Machine HIDOORS will use and update the Jamaica Virtual Machine [9]. Innovations include predictable constant time execution of all Java primitive operations like virtual and interface function calls, class instance-of checking, exception handling and synchronization, and also support routines for optimization techniques like static garbage collection. The VM will also include a predictable Java monitor implementation that has the additional requirement of constant-time execution of the monitor function, based on recent techniques that have a very low run-time and monitor overhead. These techniques will also be extended to support the priority in-

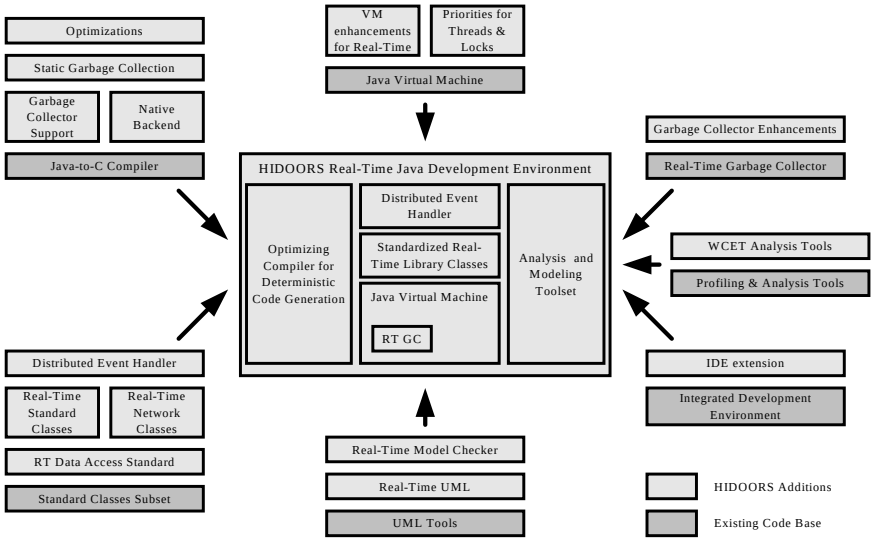

Figure 2: HIDOORS development

heritance protocol [8]. The VM will provide support for direct hardware access, as defined by the J-Consortium [1] or the Real-Time for Java Expert Group's Real-Time Specification for Java [3].

Garbage Collector An existing predictable garbage collector implementation will be further optimized and enhanced. This includes optimization of the main garbage collection loop to minimize garbage collection overhead. The worst-case execution time for garbage collection activities will be determined to provide input to the WCET analysis tool.

Java standard libraries A subset of the most important Java standard libraries shall be modified to provide predictable execution time for all operations.

Real-time Network Support New interfaces for frequently-used networks in the real-time arena (such as CAN in the automotive field and AFDX in the aeronautical) will be provided based on the Java TCP/IP socket interface.

Distributed Event Control A distributed event manager will be provided with library support for event handling with strict timing guarantees. Conceptually based on CORBA, this manager will be distributed in order to avoid a single point of failure.

Fast RMI Current Java RMI implementations were developed for high throughput in non real-time applications, so it will be necessary to adapt existing technology to perform efficient RMI.

Worst-Case Execution Time Analysis Tool A WCET tool will be provided in the HIDOORS environment which will be able to derive the timing guarantees of the software being developed. A generic accurate model of the underlying hardware will be used, taking into account the structure of the different levels of system caches and the processor architectures (with instruction latencies, pipelining and multiple execution units). This will then be used by the tool based on the technology developed in [5] and improving it for the application in Java, making use of the accurate type information available.

Integrated Development Environment Integration of the the resulting Java implementation and analysis and modeling tools into an existing Integrated Development Environment will be performed. This will provide an environment where real-time distributed embedded applications can be developed in Java, with guarantees that the code developed corresponds to the system design model and the analysis model, thus allowing an easier process of validation and verification.

4 Conclusions

HIDOORS will reuse some existing components, which will be updated to provide the desired services. We aim to develop an integrated development environment which includes a predictable Java implementation suitable for critical real-time systems development, together with the tools that provide the developer with the guarantees that the system will conform to the specified timings. The system will also include tools that will allow the developer to deploy the application in a distributed environment, with real-time properties.

References

[1] Realtime data access v.1.5, 2000. Draft International J-Consortium Specification.

[2] U. Aßmann. Proceedings of joses – java optimization strategies for embedded systems workshop at etaps 2001. Technical Report 2001-10, Genova, Italy, Apr. 2001.

[3] G. Bollella, J. Gosling, B. Brosgol, P. Dibble, S. Furr, and M. Turnbull. *The Real-Time Specification for Java*. Java Series. Addison-Wesley, June 2000. URL: `www.javaseries.com/rtj.pdf`.

[4] J. Charousset, A. Kung, and T. Gaul. Ajacs: Applying java to automotive control systems. In *Embedded Intelligence Conference*, Nürnberg, Feb 2001.

[5] C. Ferdinand. *Cache Behaviour Prediction for Real-time Systems*. Dissertation, Universität des Saarlandes, Saarbrücken, 1997.

[6] D. Genius, U. Aßmann, P. Fritzon, H. Sips, R. Wilhelm, H. Schepers, and T. Rindborg. Java and cosy technology for embedded systems: the joses project. In J.-Y. e. a. Roger, editor, *Proceedings of the Electronic Commerce, Multimedia, Embedded Systems and Technologies for Business Processes (EMMSEC'99)*, Stockholm, Sweden, June 1999.

[7] A. Lötzbeyer. *Temporale Realzeitverifikation*. PhD thesis, Universität Karlsruhe, 1999.

[8] L. Sha, R. Rajkumar, and J. P. Lehoczky. Priority inheritance protocols: An approach to real-time synchronization. *IEEE Transactions on Computers*, 39(9):1175–1185, Sept. 1990.

[9] F. Siebert. Hard real-time garbage collection in the jamaica virtual machine. In *Proceedings of the Sixth International Conference on Real-Time Computing Systems and Applications*, pages 96–102, Hong Kong, Dec 1999.