aicas realtime

Real-Time Specification for Java and certification aspects.

... Profitting from Java in embedded, realtime and safety-critical applications.

Dr.-Ing. Fridtjof Siebert
Director of Development, aicas GmbH
Java Forum Stuttgart, 3. July 2003

Who is aicas GmbH?

Aim: Promotion of modern software development methods in embedded and time-critical control, systems.

aicas's partners and customers:

aicas Java activities

European projects:

HIDOORS	High Integrity Distributed Object-Oriented Realtime Systems

ESA activities:

AERO	Architecture for Enhanced Reprogrammability...

RT Java Standardisation:

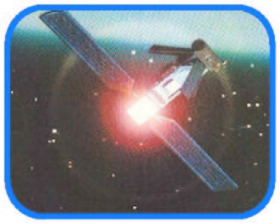
J-Consortium	active member, work on RTDA
The Open Group	RT Java group

Java User Community:

JUGS, JUG	Java User Groups
Live	Embedded-Linux Group

Deeply embedded realtime applications




Examples:

automotive, avionic, industrial automation, telecom, medical, ...

Why Java Technology for realtime systems?

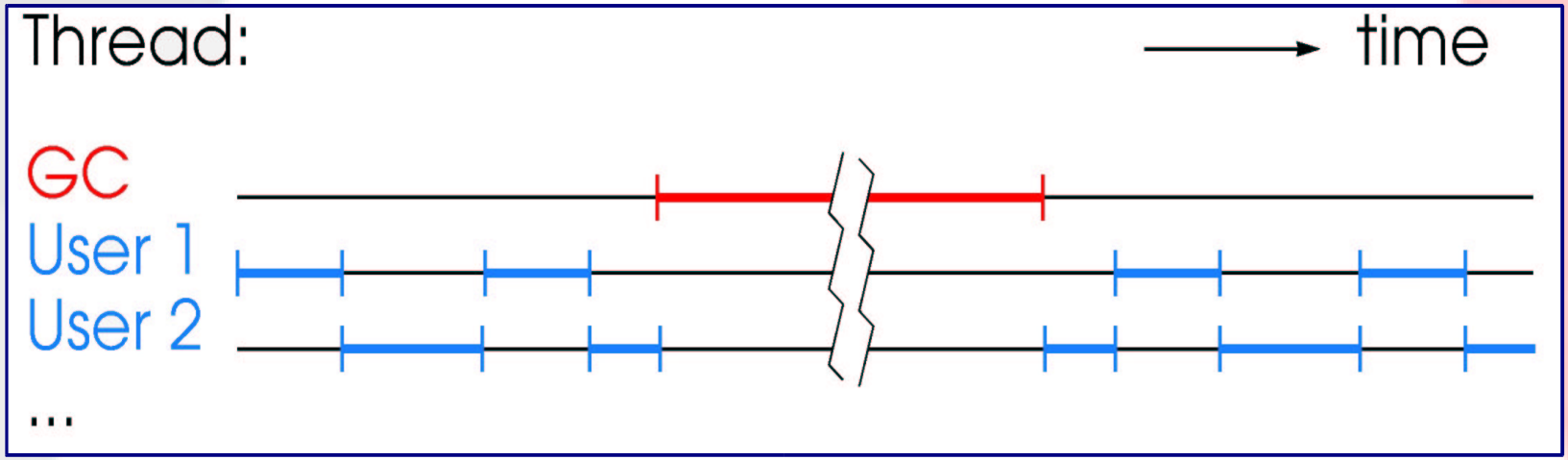
- Higher productivity
- Platform independence
- Reliability (type/pointer safety)
- Flexibility (dynamic loading)

Problems:

- Memory requirements
- Poor runtime performance
- Lack of realtime guarantees

Classic Garbage Collector

GC can stop execution for long periods of time:



Problem:

long, unpredictable pauses during execution.

Real-Time Specification for Java (RTSJ)

Extension of Java APIs to allow realtime programming

Areas addressed by the RTSJ:

- Thread Scheduling
- Synchronisation
- Memory Management
- Asynchronous Events
- Asynchronous Control Flow / Thread Termination
- Access to physical memory

Threads in the RTSJ

Realtime-Threads with at least 28 new priority levels:

New thread classes:

- RealtimeThread
- NoHeapRealtimeThread

Priority preemptive scheduling for these threads.

Priorities are higher than those of standard Java threads.

Synchronisation

Priority Inversion Avoidance mechanisms:

Priority Inheritance Protocol:

- Default behaviour of Java Monitors

Priority Ceiling Protocol:

- Optional protocol

Memory Management

Aims:

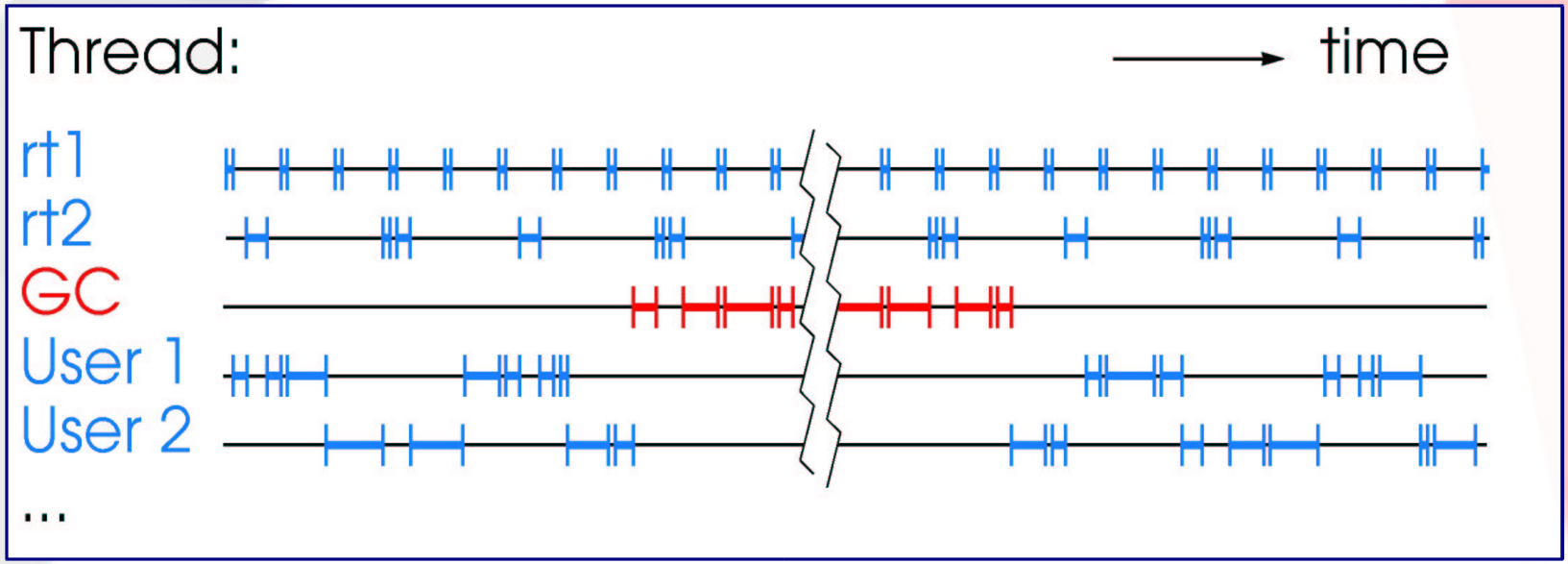
- predictable allocation
- no garbage collection in critical code

New memory areas are available:

- ImmortalMemory -- never collected
- ScopedMemory -- stack-like allocation

Garbage Collection in the RTSJ

Only special threads can interrupt the garbage collector::



The application must be split into a realtime and a non-realtime part. Synchronization between these parts is not possible!

Asynchronous Events

Implementation of Interrupt Handlers.

Features:

- Similar to Realtime Threads
- Priority and scheduling parameters such as Threads
- Handlers are executed in a thread context
- This context might change between invocations

Asynchronous Control Flow

Mechanism to throw an exception into another thread.

Applications:

- Provide a time-out for a calculation
- Terminate a thread that is no longer used

Access to Physical Memory

Safe access to physical memory regions:

Examples:

- Memory Mapped IO
- On-chip caches

Two Ways to access phyiscal memory:

- Raw-Memory, i.e., byte-wise
- As an Memory Area to store Java objects

Periodic Thread Example

```
/* priority for new thread: min+10 */
int pri = PriorityScheduler.instance().getMinPriority()+10;
PriorityParameters prip = new PriorityParameters(pri);
RelativeTime period = new RelativeTime(20 /* ms */,0 /* ns */);
/* release parameters for periodic thread: */
PeriodicParameters perp = new PeriodicParameters(null,period,null,null,null,null);
/* create periodic thread: */
RealtimeThread rt= new RealtimeThread(prip,perp) {
  public void run() {
    int n=1;
    while (waitForNextPeriod() && (n<100)) {
      System.out.println("Hello "+n);
      n++;
      }
    }
  };
rt.start();
```

Garbage Collection in the RTSJ

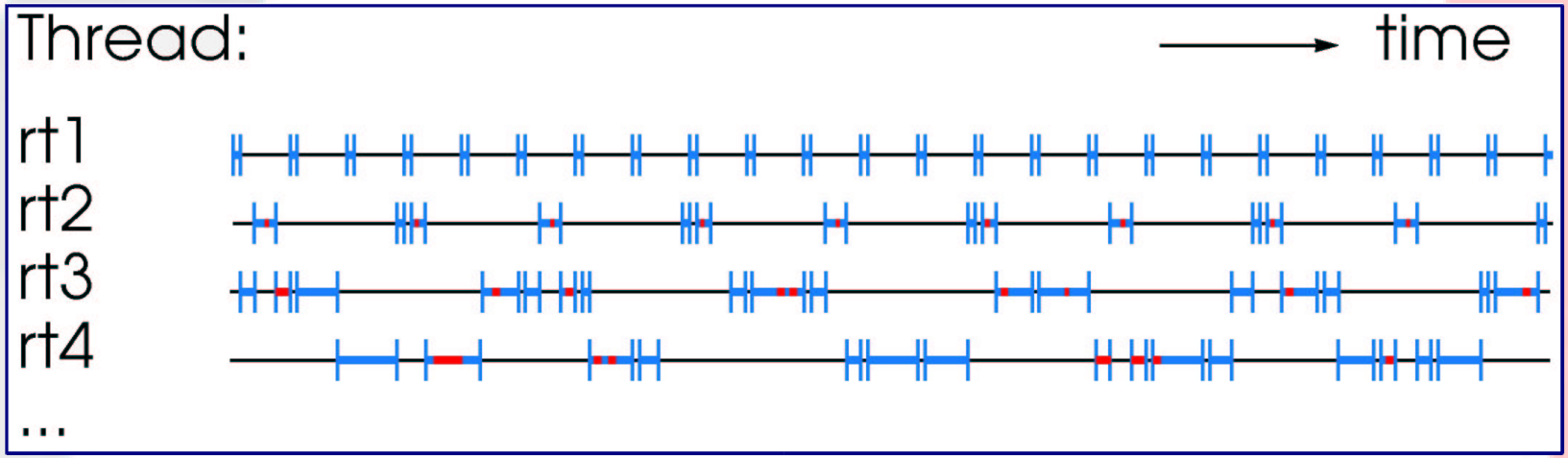
The RTSJ permit the development of realtime applications, but

- realtime code must be separated from 'normal' code
- realtime code can use only special memory areas
- No Garbage Collection, danger of memory leaks
- Danger of Priority-Inversion when synchronization with non-realtime-code

Realtime code can use only a subset of Java features!

Realtime Garbage Collection

All Java-Threads must be realtime threads:



- GC-work is performed at allocation time
- GC-work must be sufficient to recycle enough memory before free memory is exhausted
- WCET for memory allocation is required

Relaxing Constraints of RTSJ with Realtime Garbage Collection

The use of realtime garbage collection permits all threads to access the garbage collected heap.

- No restriction of heap access in realtime code
- Synchronization between realtime and non-realtime code directly possible

JamaicaVM provides realtime garbage collection.

It was used as the basis for the development in the AERO project.

AERO-JVM Java Implementation for Satellite Onboard Software

Developed by a consortium of

- Astrium SAS, France
- aicas GmbH, Germany
- University of Linköping, Sweden
- ESA ESTEC, Netherlends

Work based on JamaicaVM.

Preparation for certification of VM in space domain part of the project.

Tasks Performed within the AERO project:

Evaluation of available Java solutions

- Selection of JamaicaVM as basis

Specification of needs for space system

Implementation

- RTSJ support
- testsuite
- Port for ERC32 and Leon target system
- Static GC

Validation

Validation with Space Applications

OBJA Manager

Java-Version of state-of-the-art interpreted procedure used on Rosetta and E3000 family of telecom satellites.

Consists of mode manager, monitoring, reprogrammable mission functions etc.

Attitude Control System algorithms

Orbit propagator

C-Code Coverage

Determined using toolg gcov

raw value: 61.80%
Adjusted value: 84.73%

Acceptable for beta product

Detailed description of all C-routines, that are not covered or covered only partially

C-Code Coverage

routine	covrg.	explanation
allocJavaString	partial	- count argument is always -1 (design)
allocAndInternJavaString	partial	- Error checking code (Java exception)
catJavaString	partial	- Error checking code (out of memory)
utf82javaString	partial	- utf8 argument is never null (design)
c2javaString	partial	- cstr argument is never null (design)
class2javaString	complete	
feature2javaString	none	- Error checking code
feat2javaString	partial	- feature argument is never null (design)

etc.

Test Suite Results

Testsuite	pass	fail
AERO tests	1506	0
Jamaica Tests	50309	2
MAUVE tests	5558	79

AERO Project Traceability

Specification defines Requirements

unique id: e.g. REQ/ AERO. RT. SCH. 0020

evaluation method: test / analysis / code inspection

Validation Plan

Describes tests to developed

Design Document

Describes mapping of requirements to source code

Evaluation Document

Validation results

AERO Project Future

First Phase Finished Successfully

Second Phase of the Project is being set up

Goals of second phase

- Further Validation towards certifiability:
  - 100% code coverage
  - Removal of dead code
  - Extension of tests to cover open cases
- First Mission: Proba 2 (launch scheduled for 2005)

Conclusions

The RTSJ specification brings features required for realtime programming in the Java language.

The combination with realtime garbage collection overcomes the restrictions of the RTSJ.

Java Implementations that will be certified for use in safety critical applications are being developed and will be the next step.

More information:

www.aicas.com **www.aero-project.org**