Developing Safety Critical Applications in Java with oSCJ/L0

Ales Plsek, Lei Zhao, Veysel H. Sahin, Daniel Tang, Tomas Kalibera, Jan Vitek

www.omvj.net/oscj

Open Safety-Critical Java

PURDUE UNIVERSITY

oSCJ Overview

Open Safety-Critical Java

SCJ Library

- Level 0 support

SCJ-compliant VM

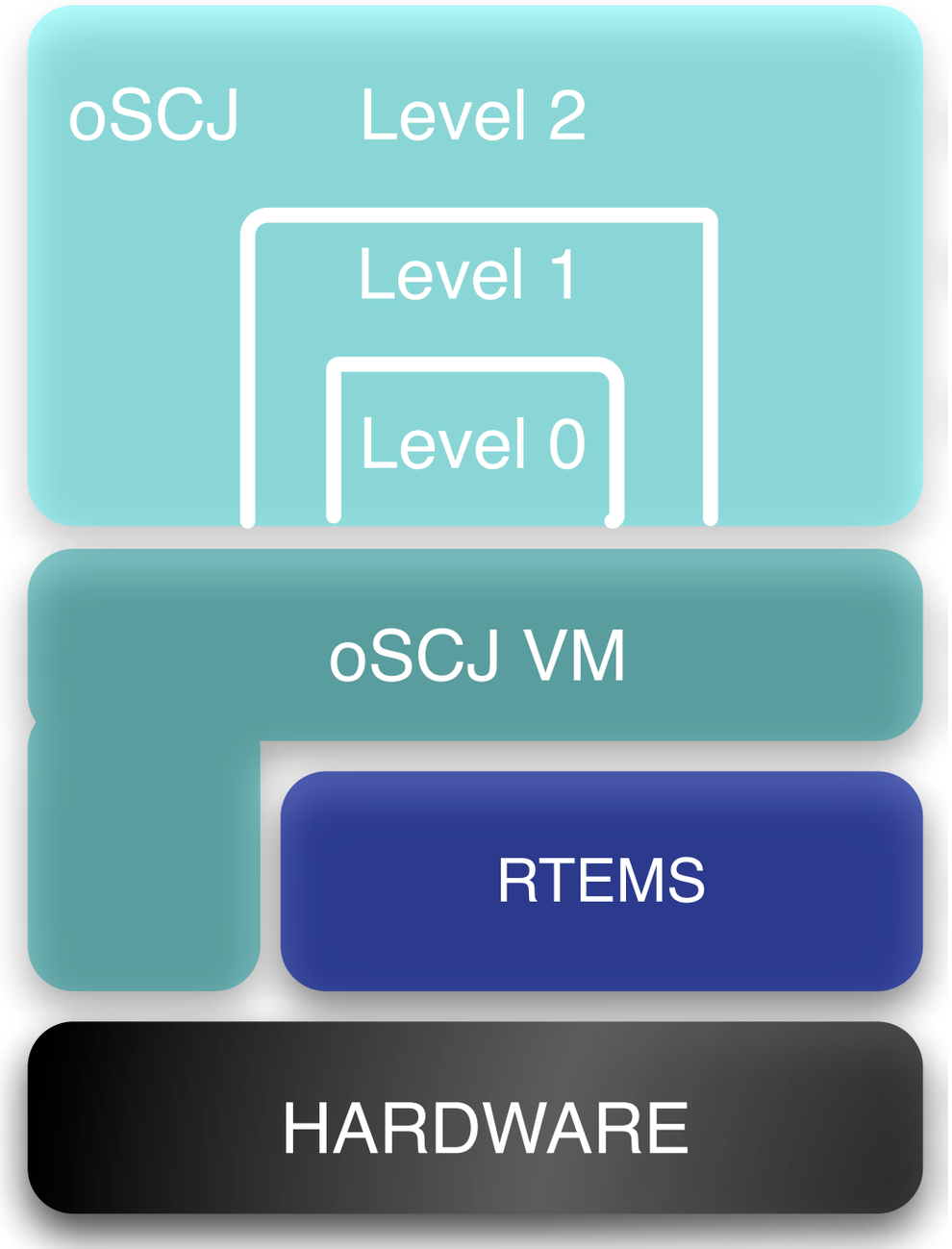
- RTEMS/LEON3 and x86 platforms

Tools

- Static Checker
- Technology Compatibility Kit (TCK)

Benchmark

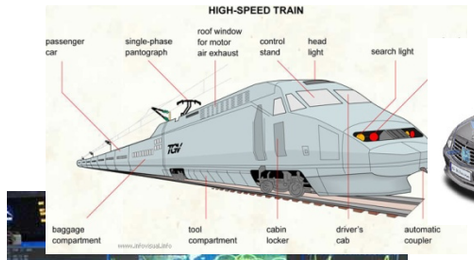
- miniCDj benchmark

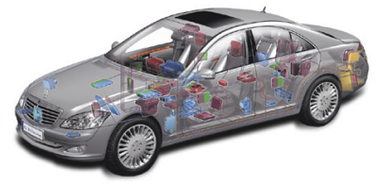


Safety Critical Systems

Safety Critical Systems

- is a system whose failure or malfunction may result in: death or serious injury to people, or loss or severe damage to equipment.





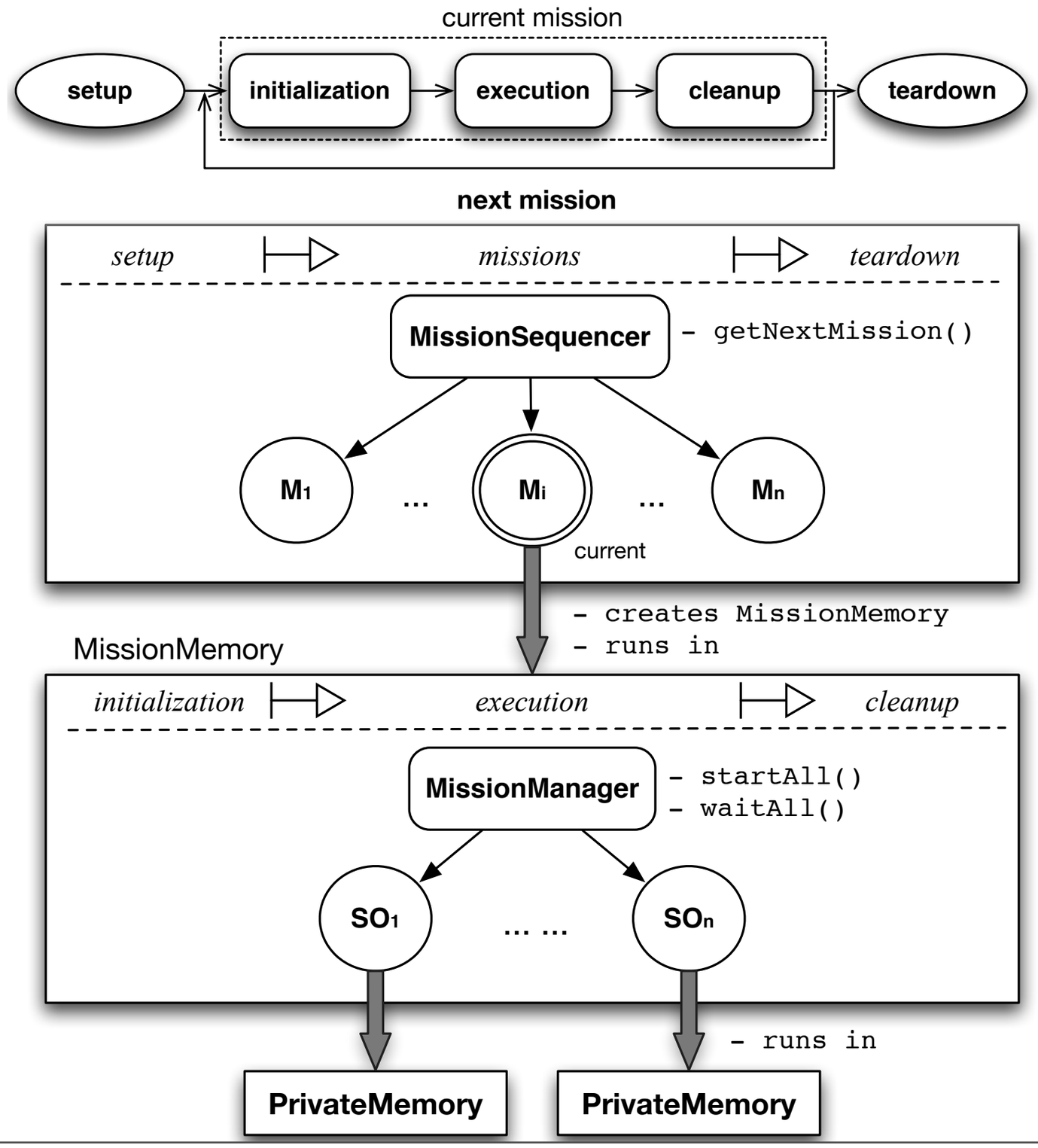
Software engineering challenge

- productivity, reusability, and availability of trained personnel

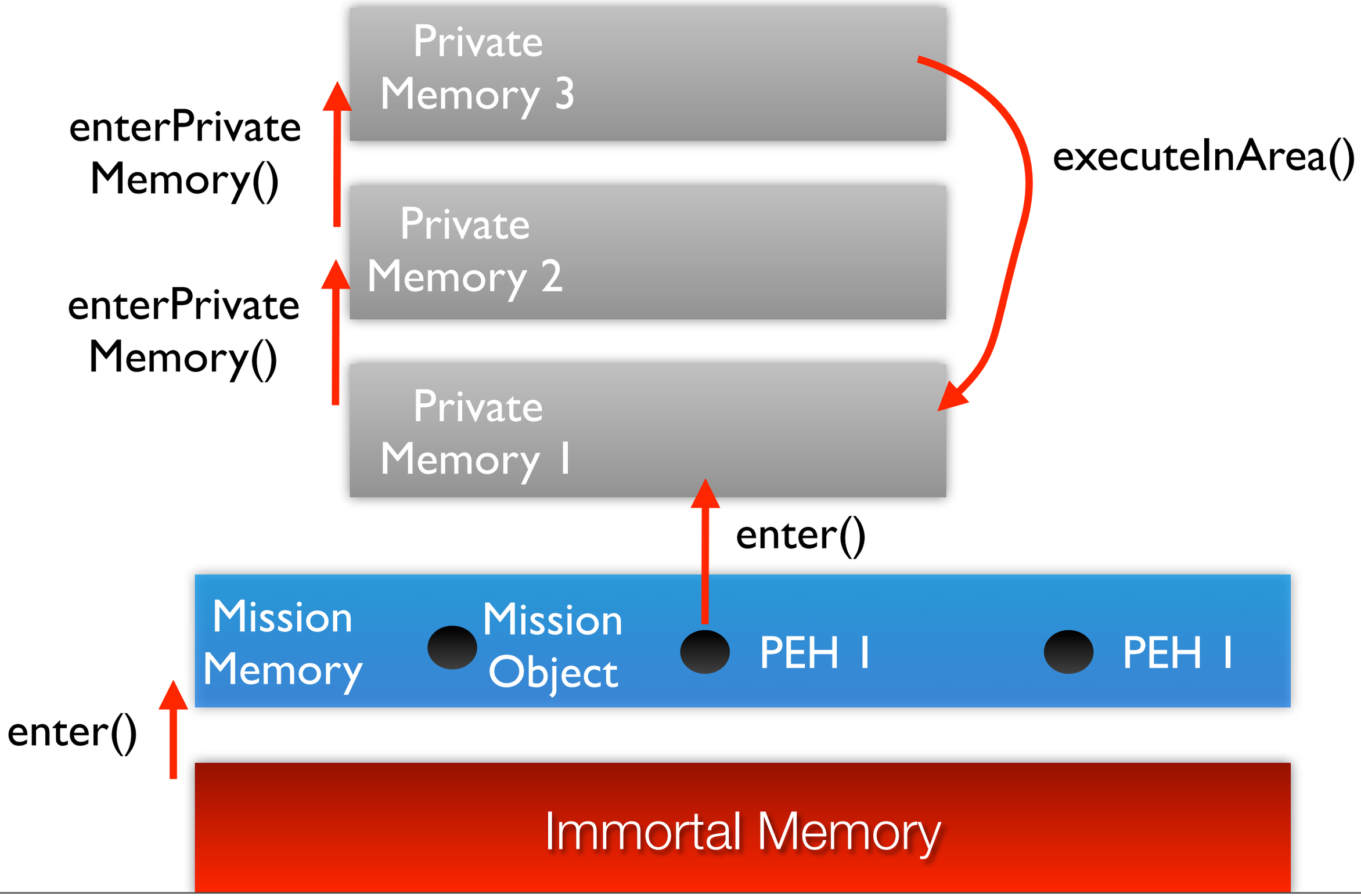
Certification standards

- DO-178 A, B, C and D

The Mission Concept



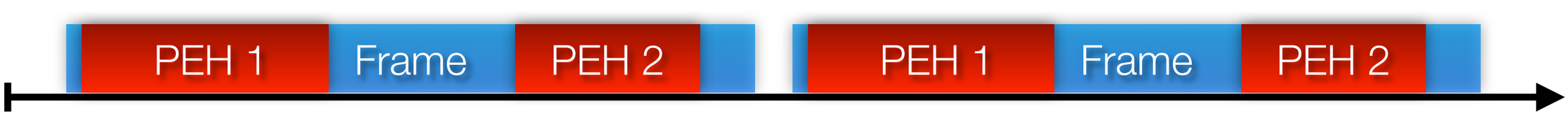
Memory Model



Compliance Levels

Level 0

- single-threaded, Periodic Event Handlers, single Mission
- frame-based cyclic-execution model
- no synchronization support required



Level 1

- AperiodicEvent handlers, Fixed-Priority Preemptive Scheduler Periodic and Aperiodic Event Handlers

Level 2

- sub-missions, ManagedThreads

VM Interface

```
interface VM_Interface {
    Opaque makeExplicitArea (long size);
    Opaque makeArea (MemoryArea ma, long size);
    Opaque setCurrentArea(Opaque scope);
    Opaque getCurrentArea( );
    ...
```

Memory Management

```
    Opaque getCurrentTime{};
    long getClockResolution();

    ...
```

Time

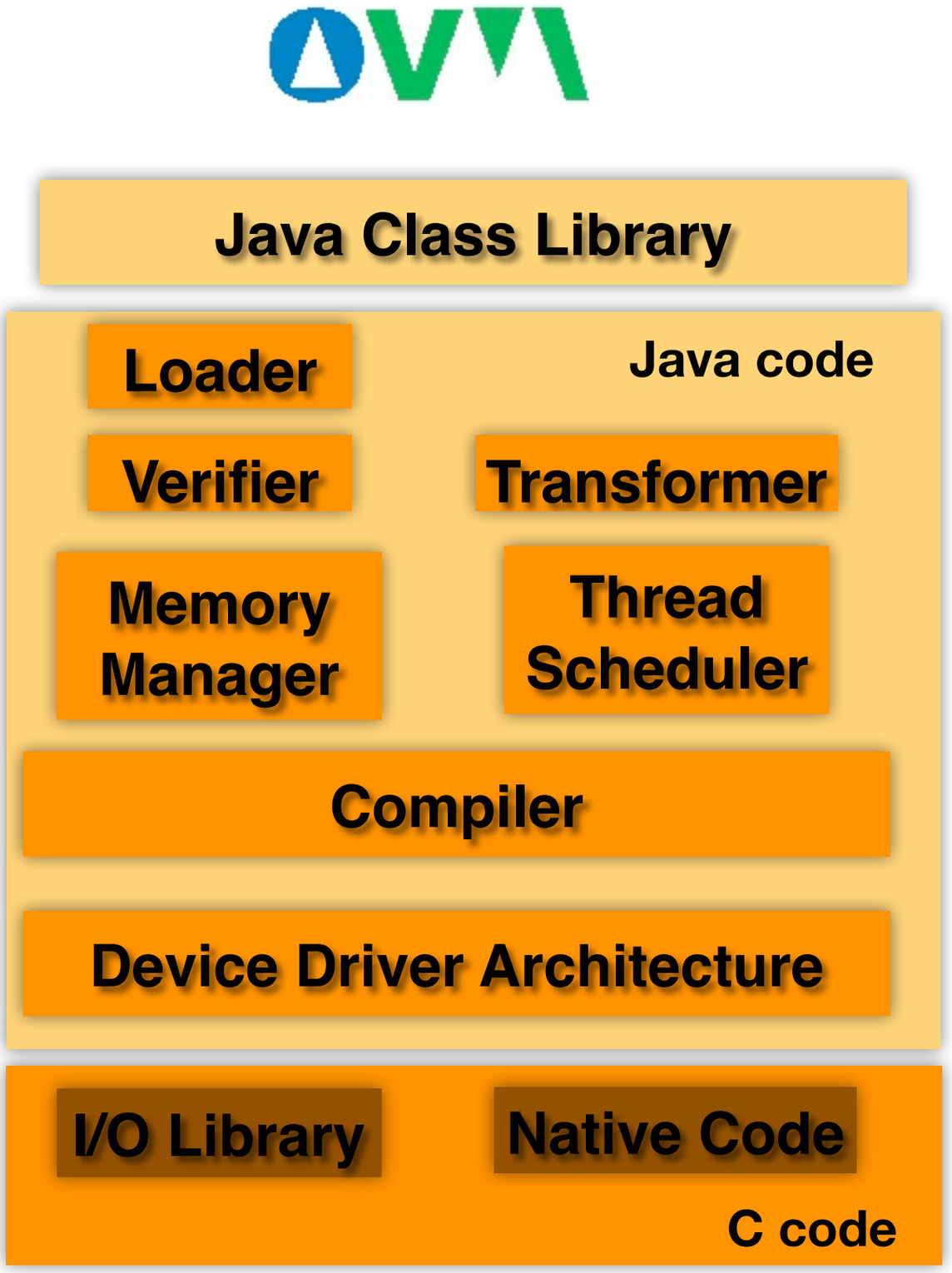
```
    int delayCurrentThreadAbsolute(long nanos);
}
```

Thread

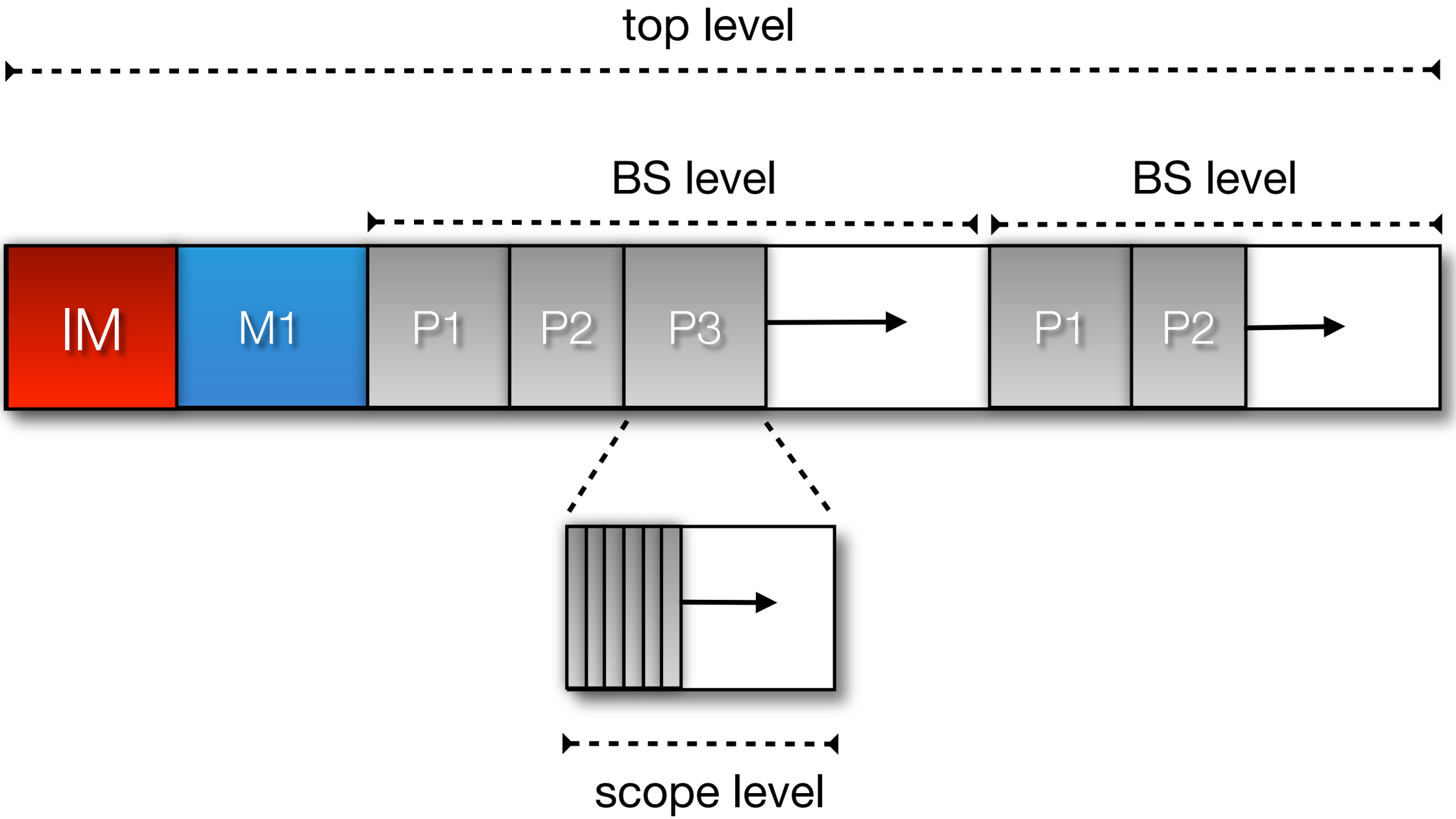
- Library designed independently on the VM
- a dedicated interface for communication with the VM
- tested with 2 VMs - Ovm and FijiVM

SCJ VM Design

- based on OVM
  - a **metacircular** Virtual Machine
  - similarly as J9, FijiVM, Squawk VM, etc.
  - requires a bootstrap JVM to run upon to create a boot image.
- a small C loader is responsible for loading the boot image at runtime.
  - Java code compiled down to C
- supported platforms : RTEMS/LEON3, x86



Memory Model Implementation



Optimizations

Synchronization Support

- Level 0 - single threaded

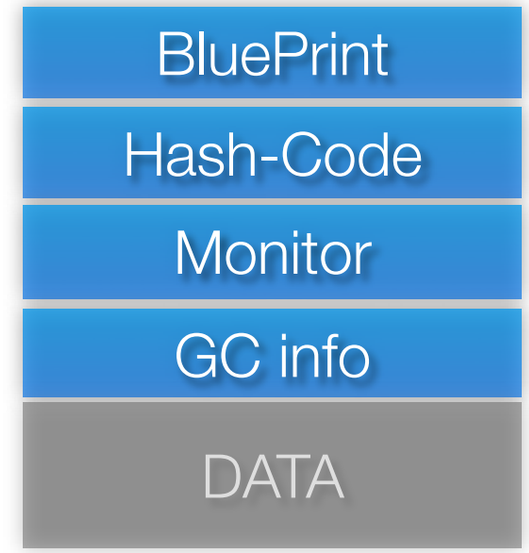
Object Model

- optimized fields
- hash-code, GC information
- hash-code
  - physical address of the object

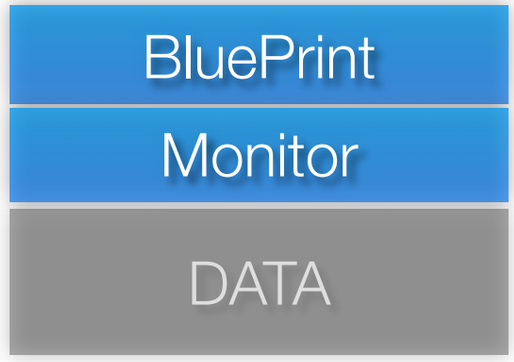
Primordial Thread

- modified to be RealtimeThread

Java Object Model



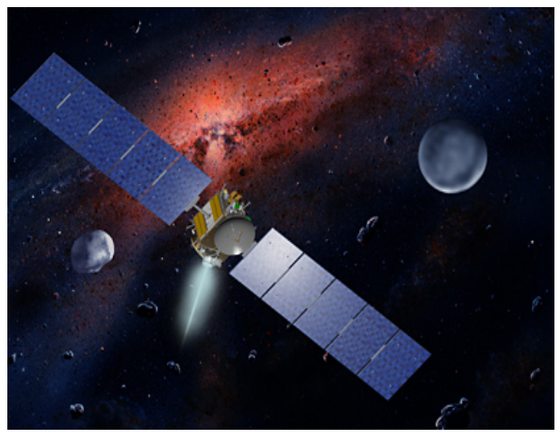
SCJ Object Model



Evaluation

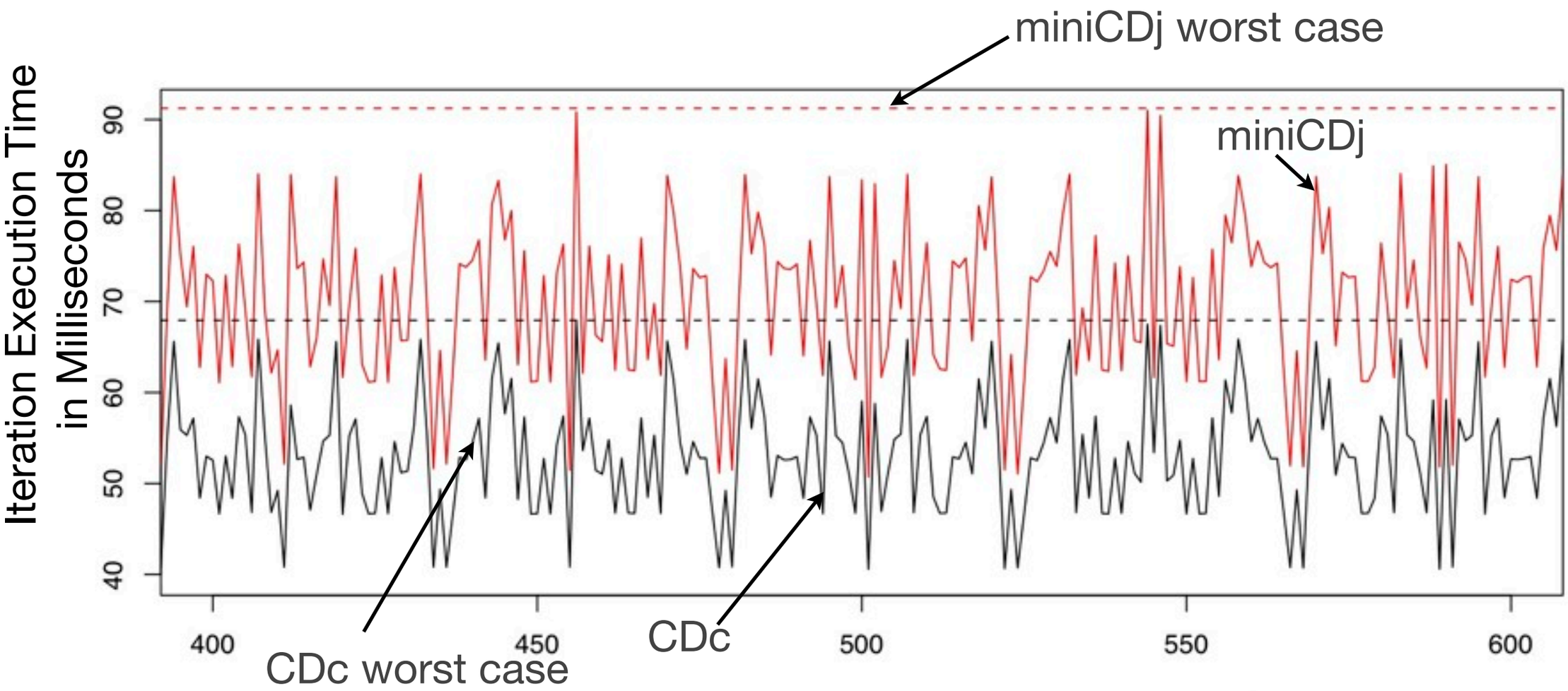
- Benchmark
  - miniCDj - periodic real-time task

- Hardware Platform
  - Xilinx FPGA GR-XC3S-1500 development board
  - 40 MHz, 8MB flash PROM, 64MB SDRAM, no FPU
  - RTEMS 4.9 OS



- SCJ configuration
  - no scope checks, miniCDj statically analyzed by SCJ Checker

Results - LEON3 board



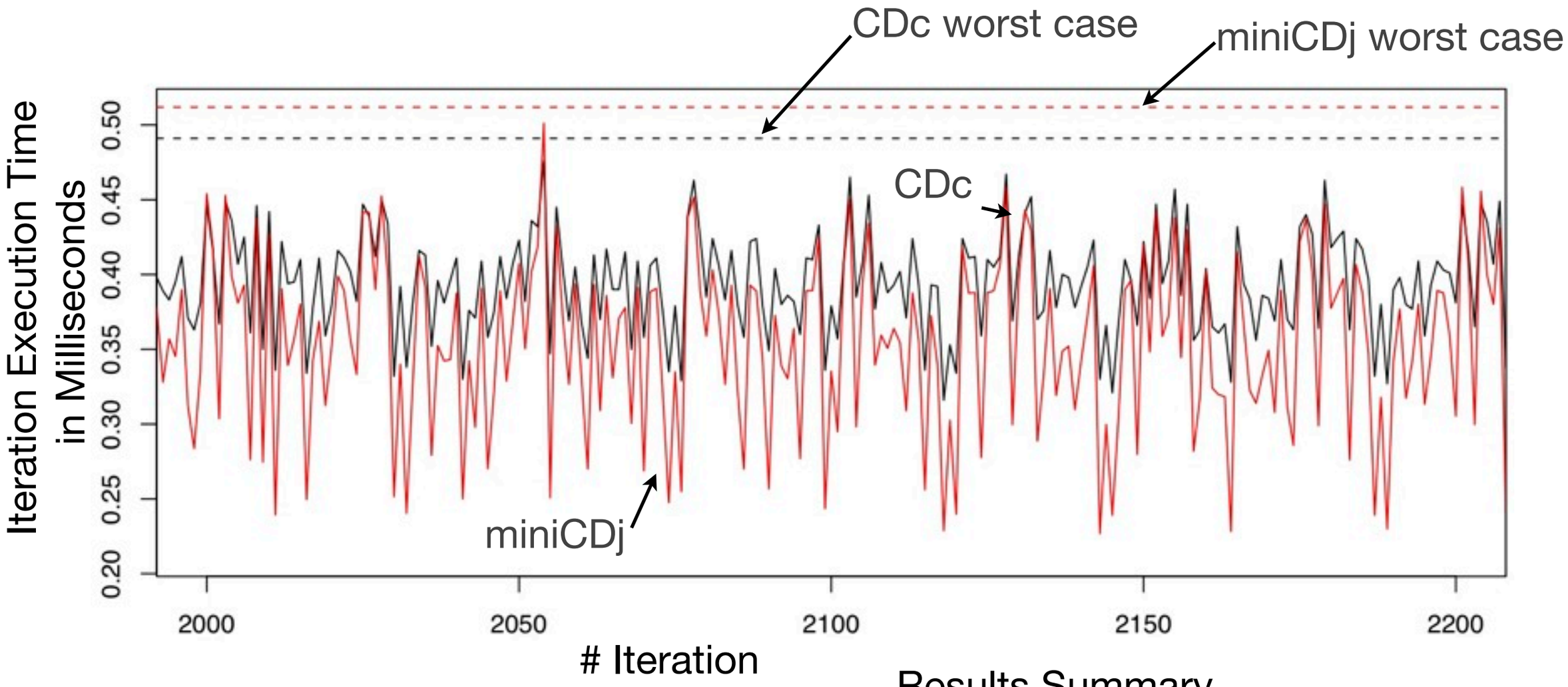
Workload:

- 6 planes
- 120 milliseconds period
- 10 000 iterations

Results Summary

	avg	max
CDc	52.8	67.9
miniCDj	69.6	91.2
Overhead:	**31%**	**34%**

Results - x86



Workload:

- 60 planes
- 120 milliseconds period
- 10 000 iterations

Results Summary

	avg	max
CDc	0.39	0.49
miniCDj	0.34	0.47
Overhead:	**-15%**	**-4%**

Conclusion

Implementation Summary

- SCJ library - 3 months
- SCJ VM
  - Memory model - 3 weeks
  - VM Optimizations - 2 weeks
  - porting VM to RTEMS - 2 months

oSCJ Distribution

- library, VM, tools and benchmark
- open-source at www.omvj.net/oscj

Open Safety-Critical Java