Hard Real-Time Java Technology

For On Board Software

ESA : F.de Bruin, Email : frank.de.bruin@esa.int

Astrium : F.Deladerrière, Email : frederic.deladerriere@astrium-space.com

aicas : F.Siebert, Email : siebert@aicas.com

Real-Time Java for Onboard Software Application

1 Needs of space software application

2 Why Java ?

3 Java in Space: AERO study

4 Questions

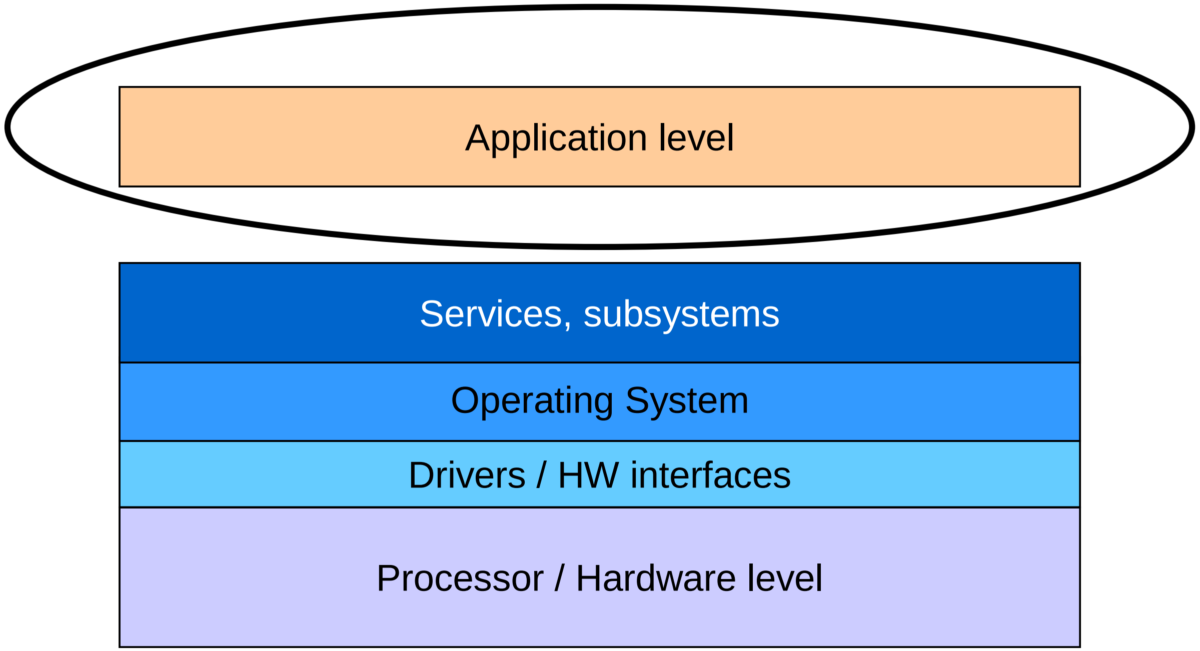
Needs of OBSW

1

Problematic at application level

- **Application level**
  - AOCS/FDIR
  - Mission
  - Payload
  - Equipment manager
  - Ground segment

  Etc.



Requirements of ideal OBSW at Application Level

- **Flexible and sure :**
  - Simple to use, (a)synchronous, determinist, real-time, robustness
  - Avoid error propagation (execution context isolated, error management)
  - Provide a high operability level (better access to system commands),
  - Capabilities for onboard autonomous systems
  - Capabilities to be connected with other existing systems (interface)

- **Standard :**
  - Known and proven solution, possibility to use commercial tools, reuse

- **Dynamic and reprogrammable :**
  - Dynamic update capabilities, security of updates
  - Possibility to specify, develop and tests later during life cycle of software

➔ Current solution = « Interpreted » systems

Limits of current interpreted solutions

- **Language**
  - Dedicated syntax and limited functionalities,
  - Learning
  - Limited evolution capacities

- **Development environment and simulation tools**
  - Dedicated tools and consequently high development and maintenance cost
  - « Poor » solutions (ergonomic and functionalities) compare with commercial tools on « standard » languages and systems.

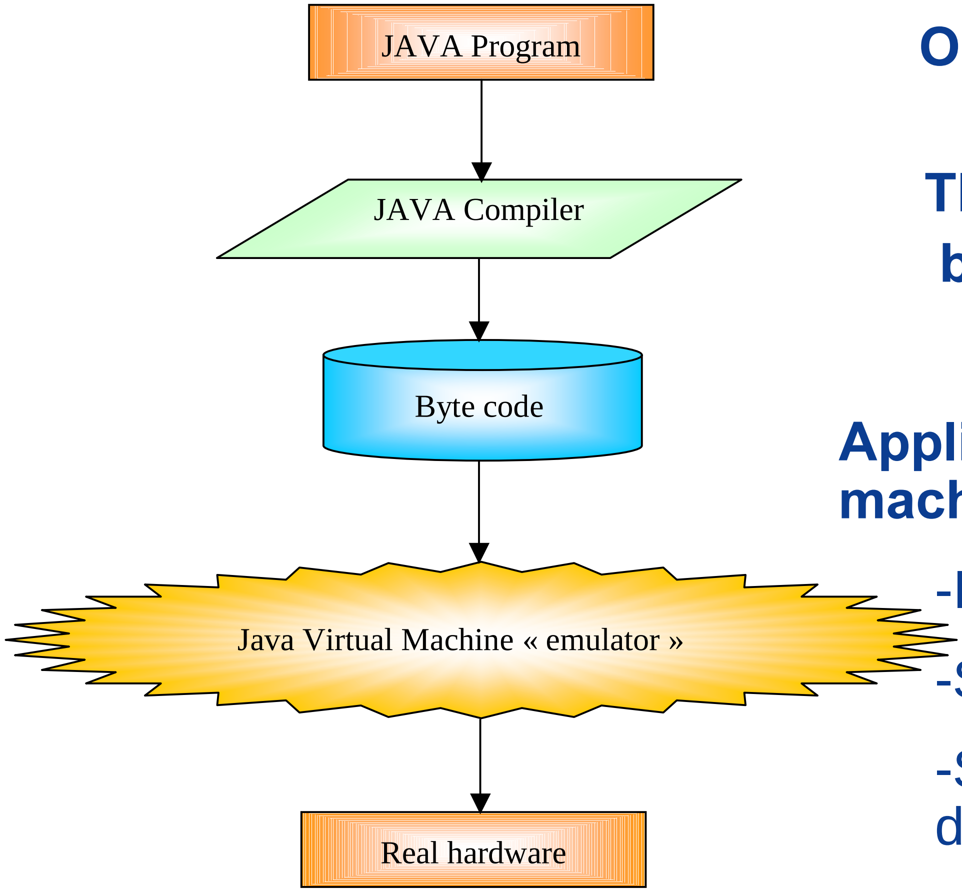
- **Performance**
  - CPU constraints (i1750) with impact on design and algorithms
  - Low ratio of interpretation performance vs. CPU power required

➔ Technological step required, Java is a candidate

Why Java ?

2

Basic principle of the Java language …



Object oriented language

The compiler generate assembler binary code (« byte code »)

Application run inside Java virtual machine (JVM)

-Independency from host machine

-Secure application execution

-Standard tools (IDE, SVF etc.) used during development phase

Java : a good candidate for embedded systems …

- **Robust and sure**
  - Differed allocations, symbolic references, independence of compiler,
  - Memory model, no rewrite in memory, no pointer, restricted access
  - Bytecode verifier, types constrained, overstack check, access control
- **Dynamic object system**
  - Controlled and secure reprogramming, Dynamic Load/Link
- **Familiar**
  - Simplified C syntax (no pointer or memory to manage)
  - ADA mechanisms (genericity, polymorphism, exceptions)
- **Neutral and portable architecture**
  - Virtual machine system: Independence from hardware
  - Standard language

Java : advantages of technology

- **Structured and secure language**
  - Application code more simple
  - Secure execution
- **Commercial standard**
  - Important number of IDE and SVF
  - Large and various libraries at standardized format
  - Fast application development
- **« Real » object oriented language**
  - Easy application design
  - Homogenous development
  - Direct use of object specification methods (LDS, UML etc.)
  - Large reuse of code
- **Syntax inspired from C/C++**
  - Large programmer community
- **Dynamic execution**
  - Dynamic class loading
  - Distant debug
- **Performances**
  - Multiple execution models : interpreted, JIT, AOT, Hardware …
  - Secure memory management
  - Fast reprogramming
  - Powerful multi-tasking

Java : Benefit for space context

- **One solution for every systems / Commercial standard**
  - Access to new SW engineering technology
  - Cost reduction of IDE, tools, test environment etc.
  - Reduced portage cost : only system interface and real-time scheduling to adapt

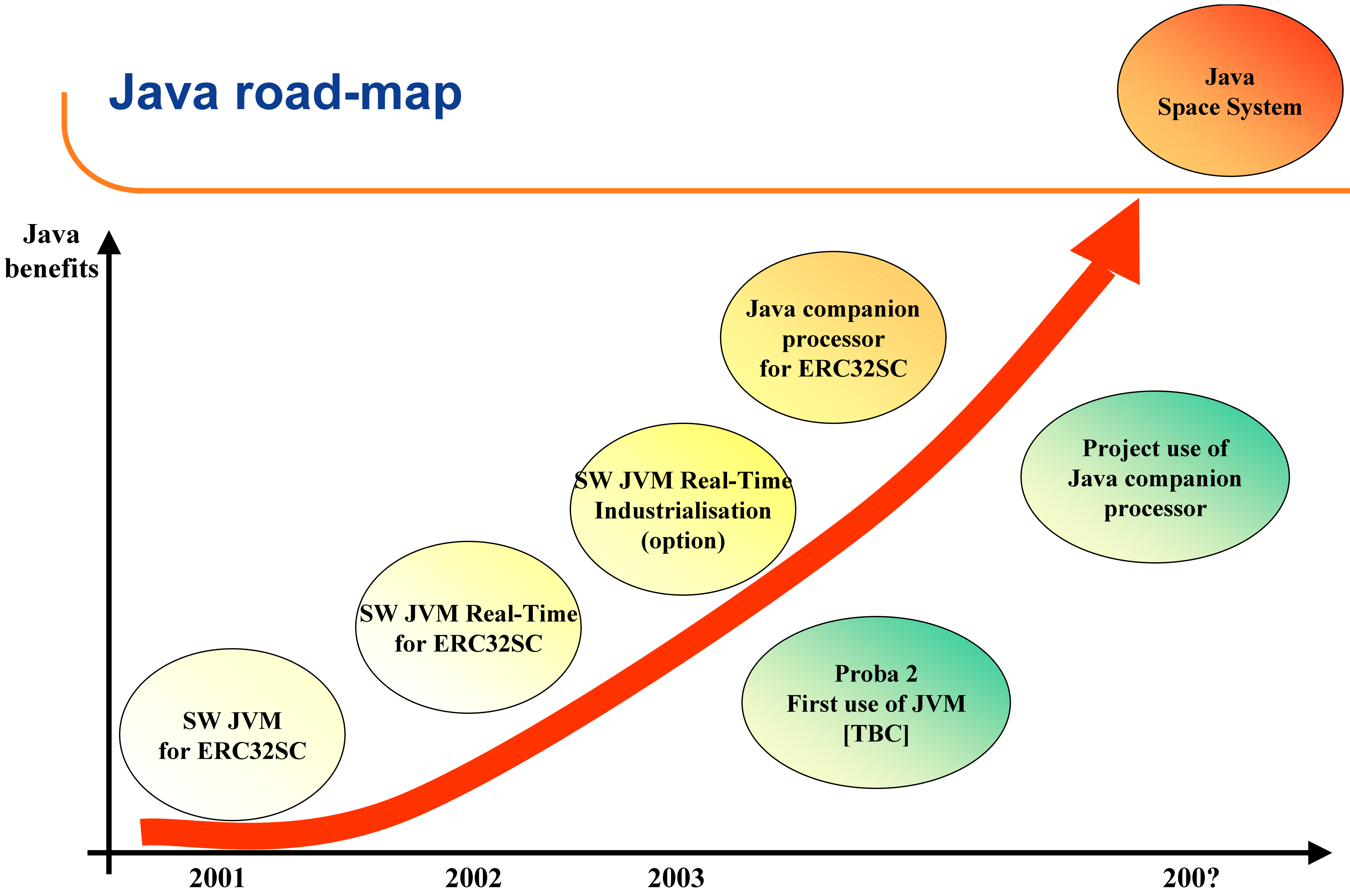
- **Application development and testing cost reduction**
  - Reuse of existing procedure inside APIs / Framework
  - Incremental validation API per API
  - Simpler application based on APIs / Framework

- **Development cycle delay reduced**
  - Reuse
  - Simple and secure :
    - Capabilities to develop a base system before launch
    - Global system update later
    - Mission update, correction etc.

History of Java at Astrium

- 1997 : first experiments with Java technology
- 1998 : development of ground Java software (tools and simulators)
- 1999 : internal study of Java interpreter design and techniques
- 2000 : development of first Java interpreter for ERC32
- *2001* : beta product ERC32-JVM (CLDC compliant JVM for ERC32)
- 2002 : Aero study with new partners
- 2001-2002 : Java Processor market survey
- 2003 : Java Processor internal study
- 2003 : Astrium Beta tester of Java 1.5 Plateforme
- *2003* : 1st Hard real-time JVM for ERC32/Leon : AERO-JVM

Java road-map



3

Architecture for Enhanced Reprogrammability and Operability

ESA AERO Study: a Real-Time JVM for ERC32

- **ESA 2001 Call for Innovative Technology + Astrium funding**
- **ESA Technical Officier : Frank de Bruin**
- **Consortium : Astrium, aicas GmbH, Linköping Universitet**

- **Search of JVM candidate**
  - Market Survey
  - Analysis of 21 solutions and selection of base JVM

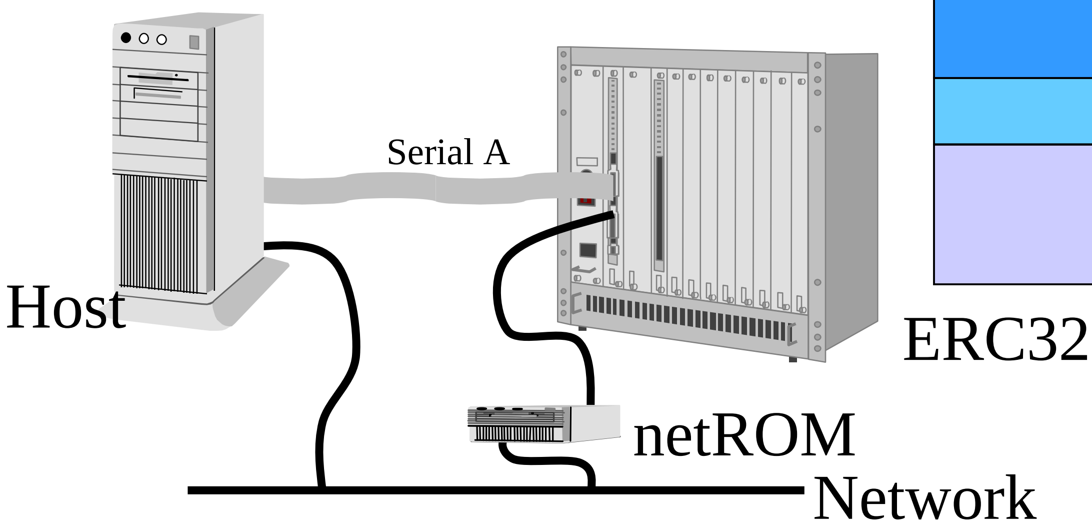
  ➔ aicas GmbH « Jamaica » core technology chosen (mature RT env.)

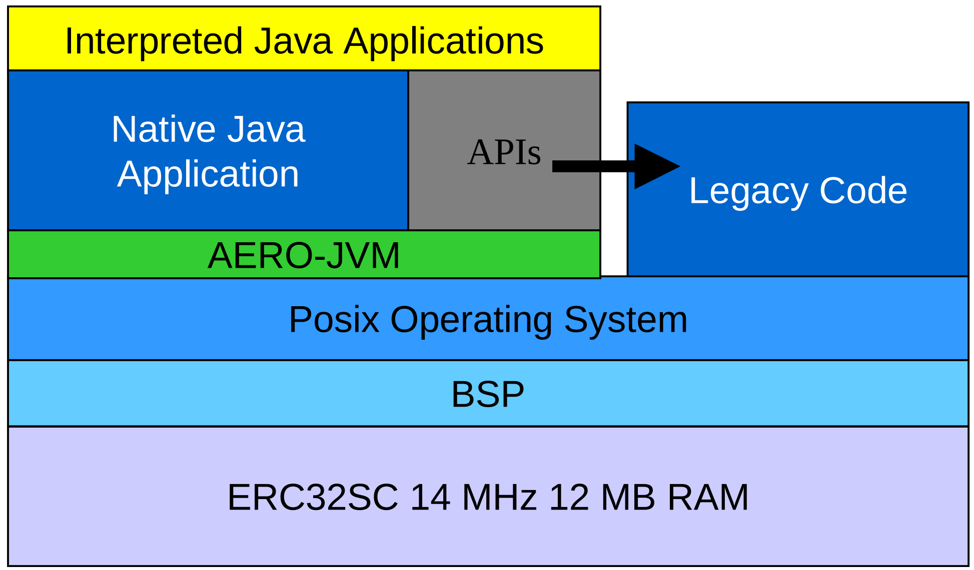
- **Customisation of solution**
  - Specification of design and required functionalities in space context
  - Update of core functions, porting to ERC32 processor, and VxWorks/RTEMS OS.
  - Development of new libraries, drivers, etc.

- **Validation and evaluation**
  - Standard Java compliance and quality assurance checks
  - Performance and functional evaluation on ERC32 bench and Leon (FPGA)
  - I/O access tests (1553 Bus)

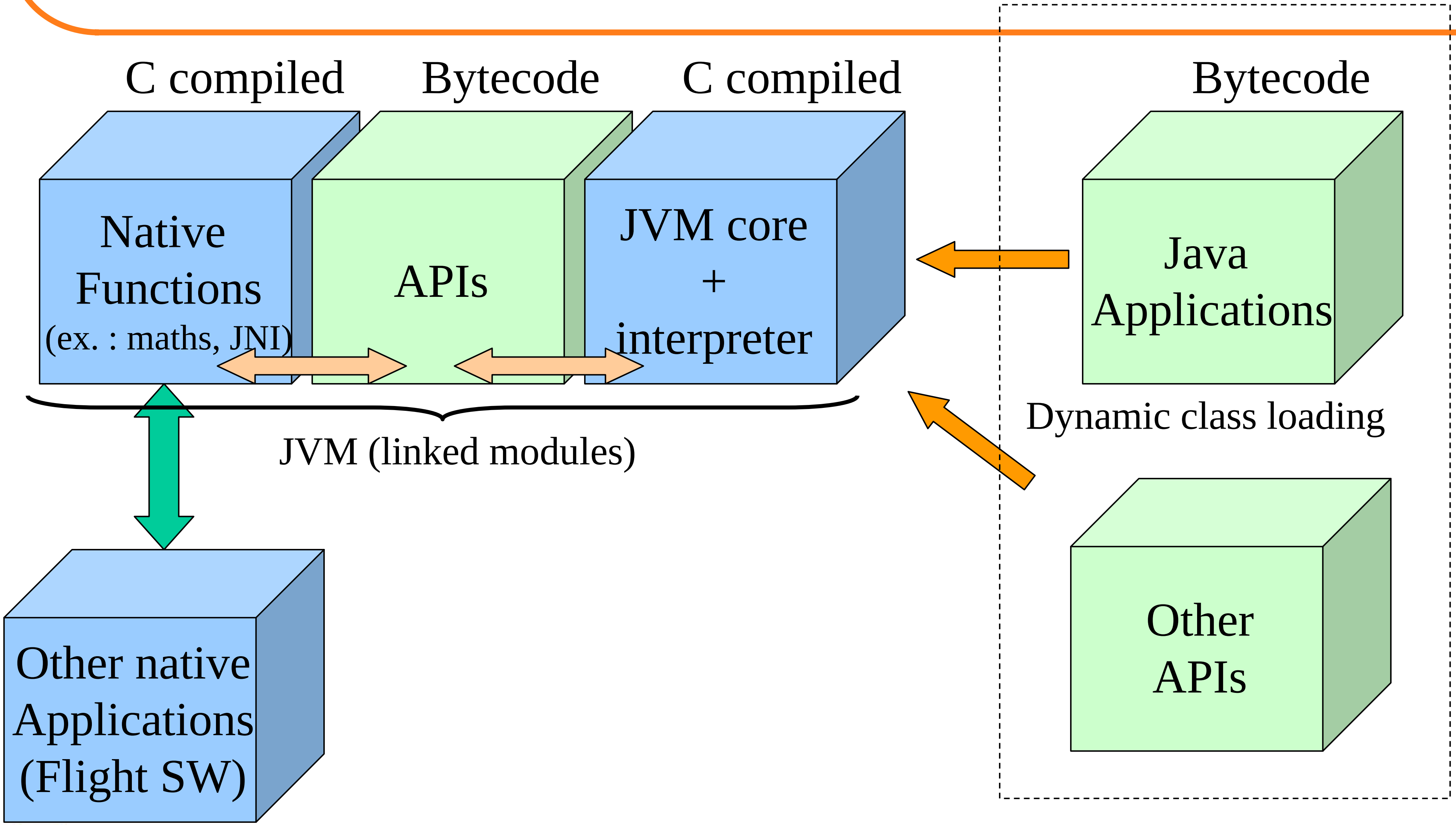
AERO-JVM: Prototype on ERC32 bench

The AERO-JVM enables space systems to use Java and its advantages with a Real-Time deterministic execution model

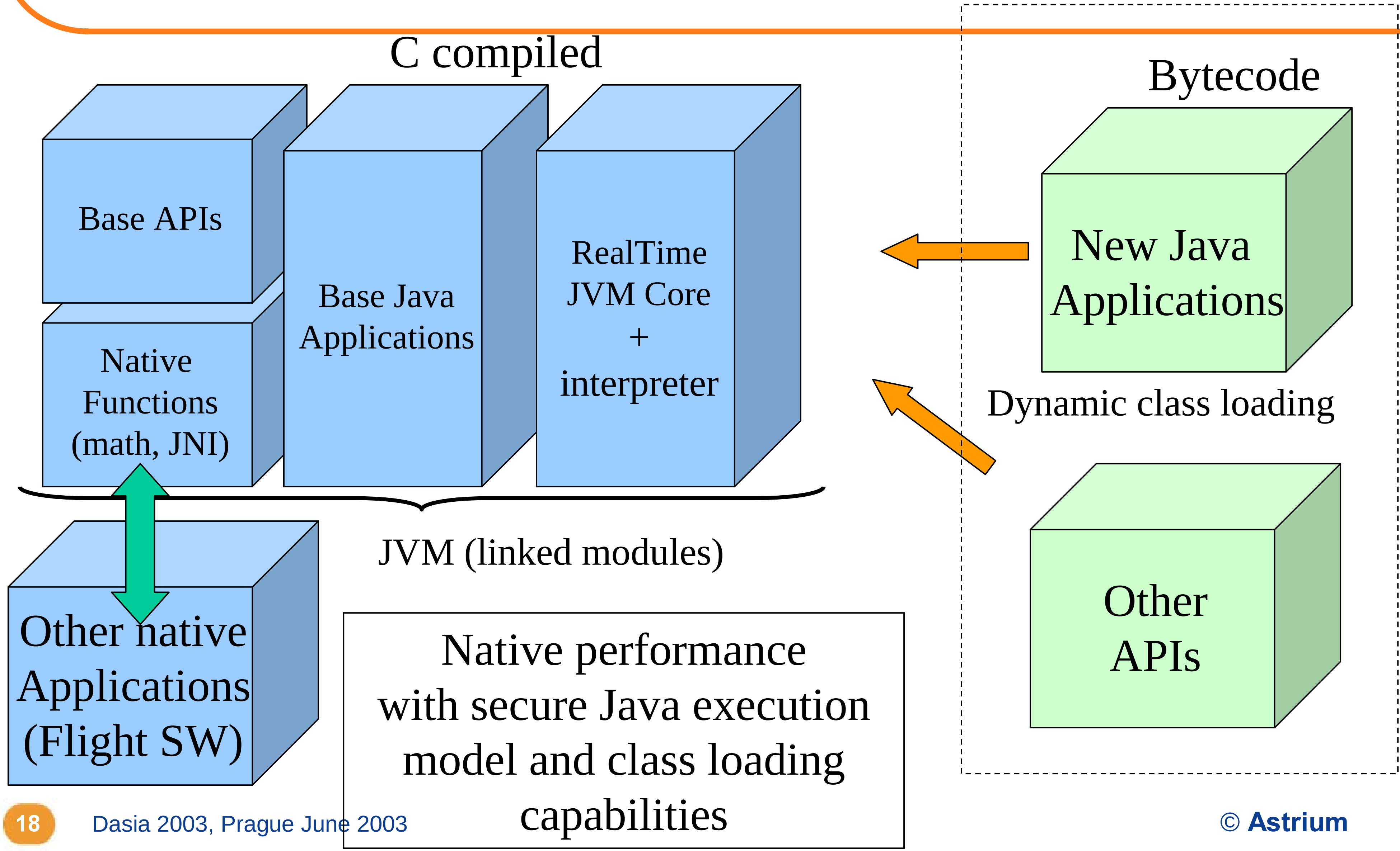




Standard JVM design

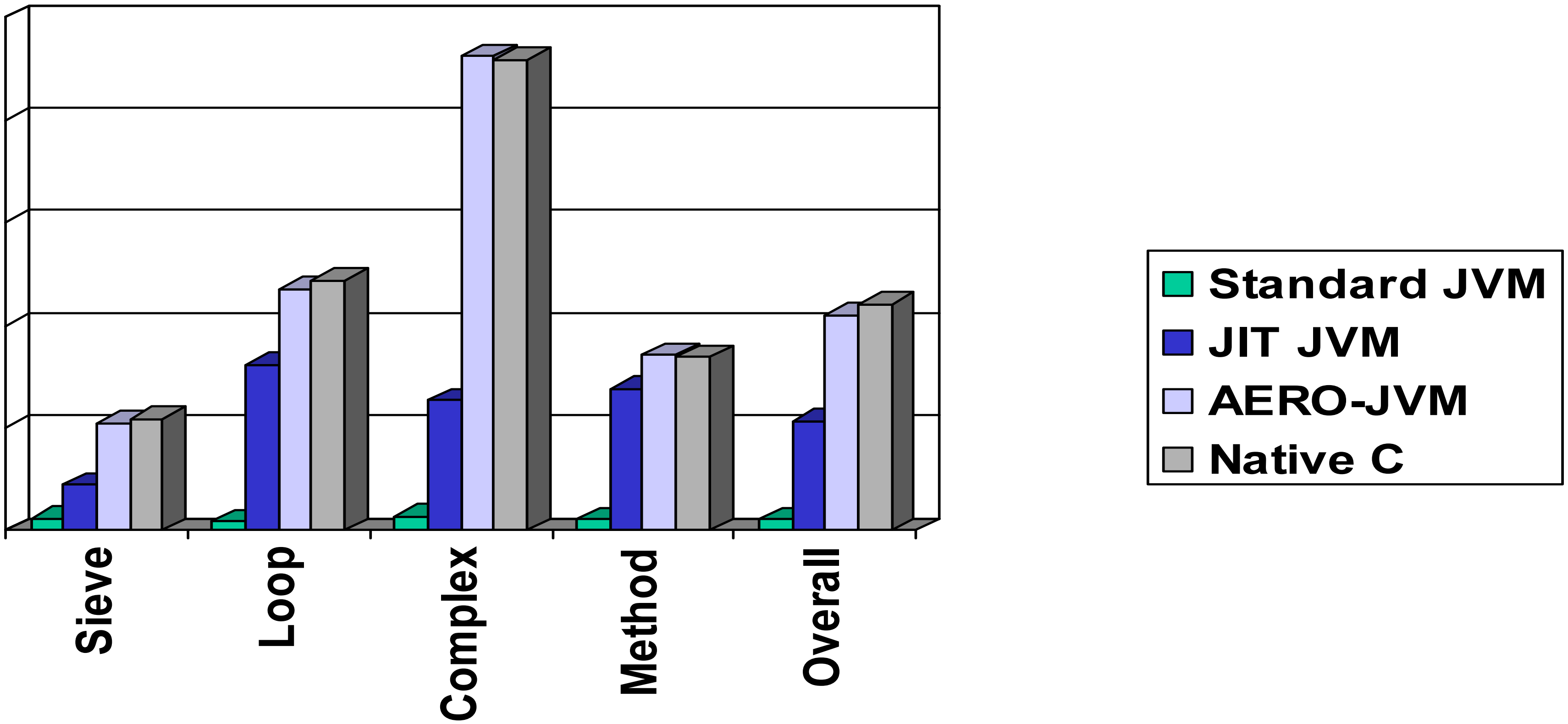


AERO-JVM design



AERO-JVM Performance

Synthesis of first evaluations (comparative ratio on ERC32)



AERO-JVM characteristics and innovations

- Designed for **ERC32/Leon** (but could be ported easily to other processor)
- 32 bits architecture with 64 bit type support
- Posix + Space domain OS supported: VxWorks, RTEMS (under develpt), etc.
- Full Java support (JDK1.3), improved JNI/RMI, reflection, dynamic, remote debug

- Predictable/Deterministic optimised memory management
- Ahead of Time compiler which generates code conforming to standard Java (Native real-time Java runtime)
- **Native performance level**/dual execution mode: **interpreted and native execution**

- Strict Hard Real-time behaviour with multithreading support
- **Real Time Java Specification** (RTSJ) « standard » compliant (incl. Memory API)
- No Licence fees for ESA projects

➔ AERO-JVM brings Java ready for OBSW applications

AERO project status

Project progress :

02/2002	Project start
04/2002	Selection of base solution
06/2002	AERO-JVM Specification, Validation Test plan
08/2002	Customisation and developments (including SVF)
11/2002	Validation & Evaluation on ERC32 bench and Leon FPGA
04/2003	Delivery of AERO-JVM « Beta product »

Work to be done:

- « Industrialisation » phase (proposed to ESA)
- Flight demonstration on Proba 2 (proposed to ESA)

Project web site: **www.aero-project.org**

Contact: Frédéric Deladerrière
frederic.deladerriere@astrium-space.com

➔ Demonstration and evaluation on request

Questions …

?