



# The Real-Time Specification for Java and related projects

**Christophe Lizzi**  
**Sun Labs Europe**



*Sun*  
microsystems  
We make the net work.

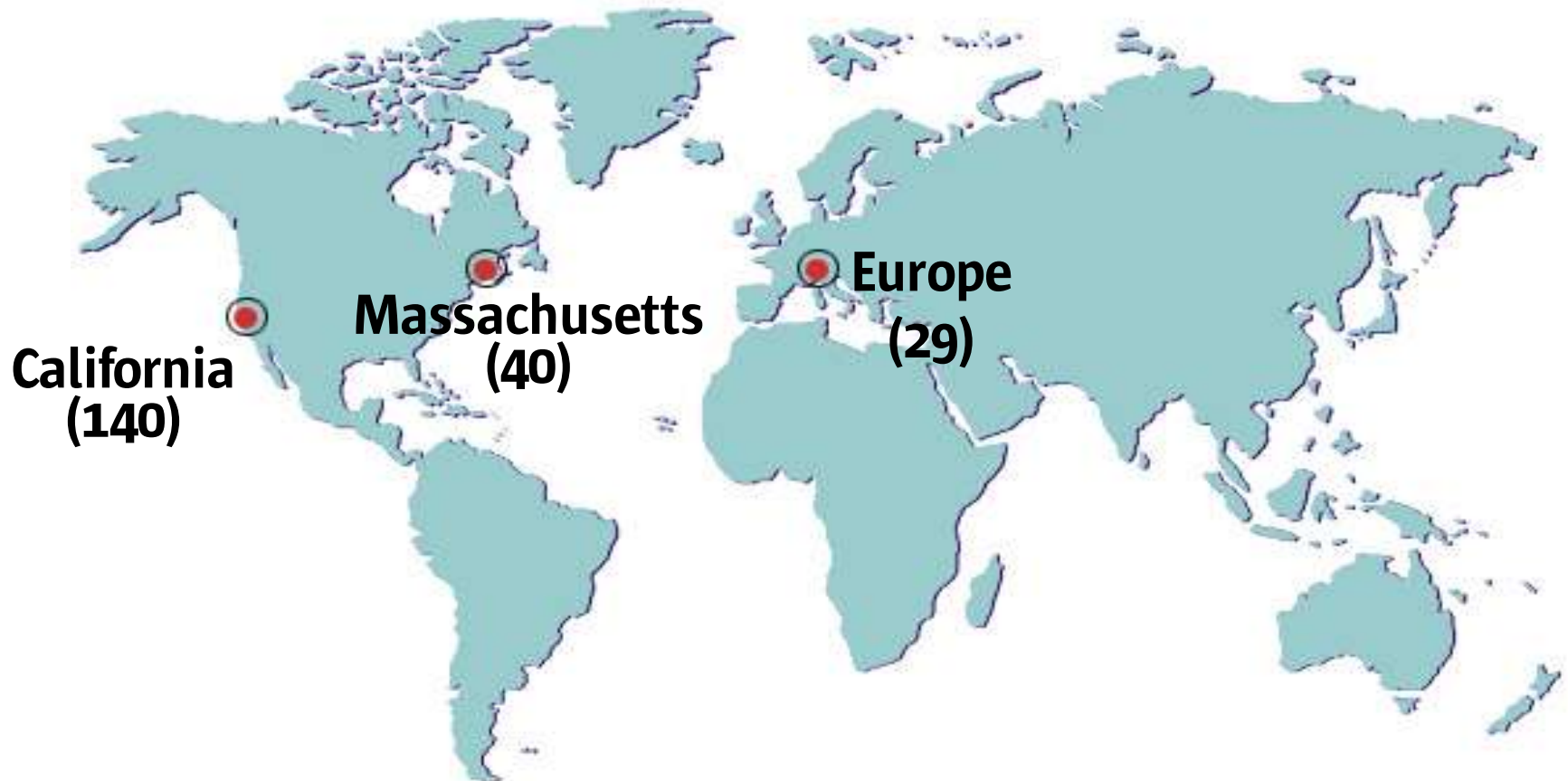
# Outline

- **Sun Labs Europe**
- **The Real-time Specification for Java**
- **Project Golden Gate**
- **Project Mackinac**
- **Project Tancarville**

## **Sun Labs Mission**

**Innovate**  
**Demonstrate**  
**Deploy**

# Sun labs Demographics



# Sun Labs Europe, Grenoble

- **22 researchers, 3 Ph.D interns, 3-5 Interns**
  - High-speed networking; TCP offloading
  - Real-Time Java; highly scalable JVM
  - High Productivity Computing
  - Sensor network architecture
  - Security, mobility
- **R&D relationship with universities, institutions and independent teams through Europe**
- **Located with Grenoble Engineering site**
  - one of 11 engineering sites in EMEA (Ireland, France, Germany, Czech, Israel, Sweden, UK)

# RTSJ History

- **Idea in 1998**
- **1998 Embedded Systems Conference**
- **Java Community Process 1999**
- **JSR-01 Approved Feb. 1999**
- **0.9 Release June 2000**
- **1.0 Release Nov. 2001**
- **Reference Implementation Jan. 2002**
- **Product March 2003**
  
- **Greg Bollella, spec lead, now with Sun Labs Europe**

# RTSJ Guiding Principles

- **Predictable execution**
- **Support current real-time software practice**
- **Backward compatibility**
  - ➔ no syntactic extension
- **Any Java edition**
- **Support leading edge scheduling**
- **Allow for implementation trade-offs**
- **WOCRAC**
  - ➔ write-once carefully, execute anywhere conditionally

# RTSJ Enhanced Areas

- **Scheduling**
- **Memory management**
- **Synchronization**
- **Asynchronous events**
- **Asynchronous transfer of control**
- **Physical memory access**

# Scheduling

- **Generalize the scheduling/dispatching definition and mechanism**
- **Base scheduler, fixed-priority, preemptive**
  - at least 28 unique priority levels
- **Direct support for periodic, aperiodic, and sporadic threads**
- **Temporal reqs expressed in application terms**
  - deadlines, periods, interarrival times, costs
- **Framework for additional schedulers**

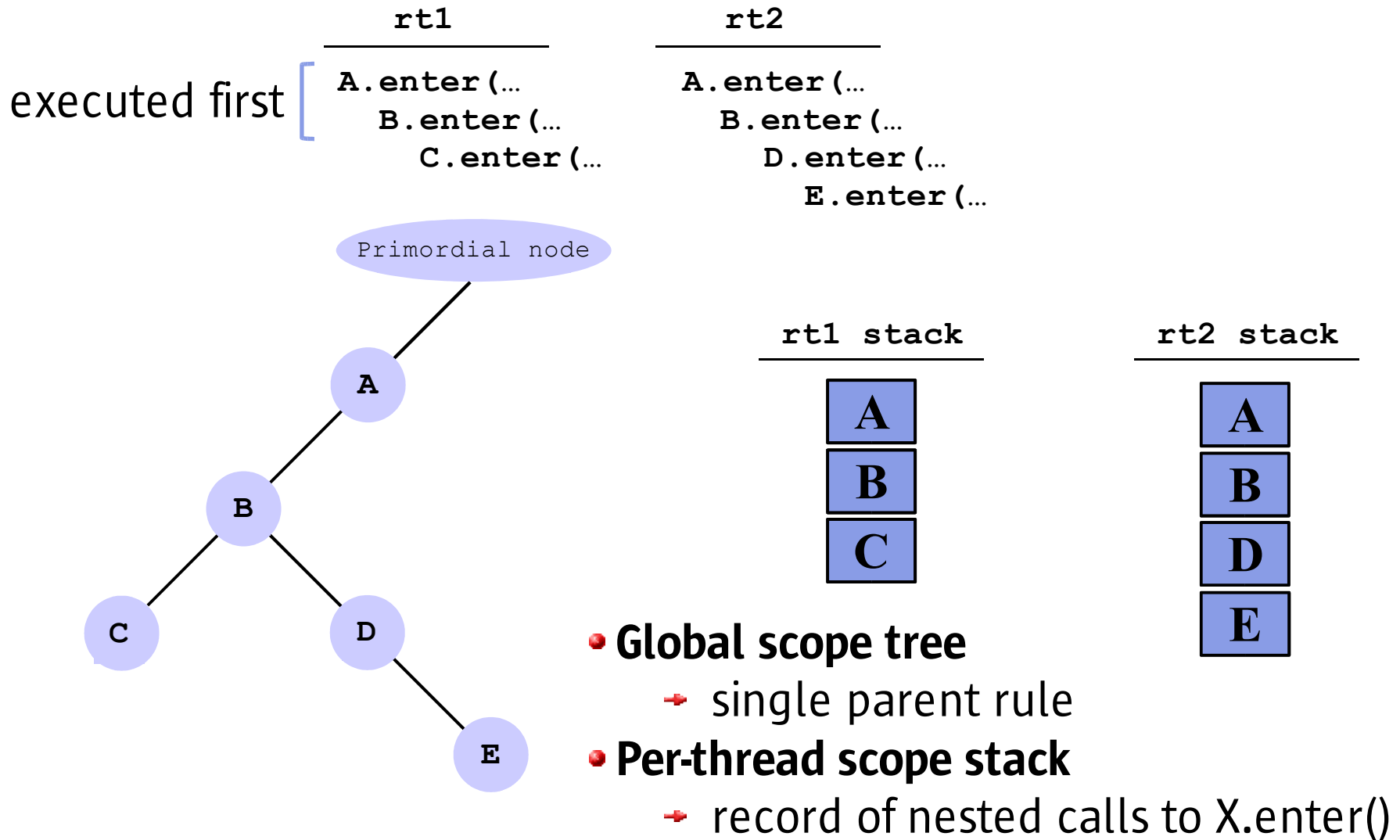
# Memory Management

- **RTSJ changes the notion of object lifetime**
  - ➔ i.e., when an object is a candidate for collection
- **Manual: lifetime controlled by program logic**
- **Automatic: lifetime controlled by visibility**
- **RTSJ Memory Types: lifetime controlled by syntactic scope**
  - ➔ all objects live until control flows out of scope
  - ➔ when control leaves scope, finalizers execute and complete before the memory area is accessed

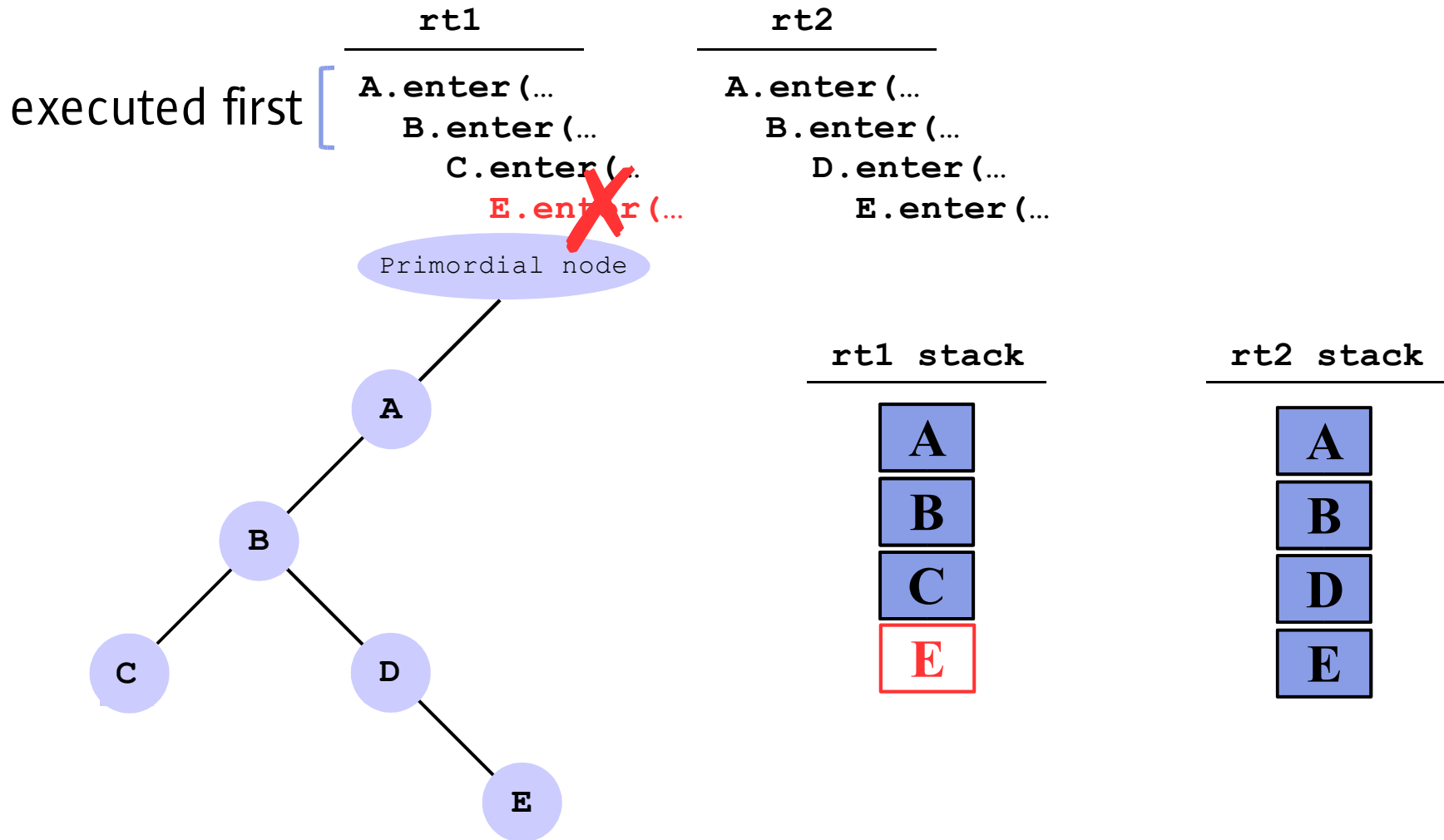
# Scoped Memory

- **Application defined and managed heaps without automatic memory reclamation**
  - ➔ eliminate all latency of garbage collector
  - ➔ temporary objects
  - ➔ NoHeapRealtimeThreads can preempt GC indefinitely
- **Why controversial**
  - ➔ requires to consider memory management
  - ➔ requires access checks
  - ➔ aren't real-time GC good enough?

# Scope tree and stack



# Scope tree and stack (cont.)

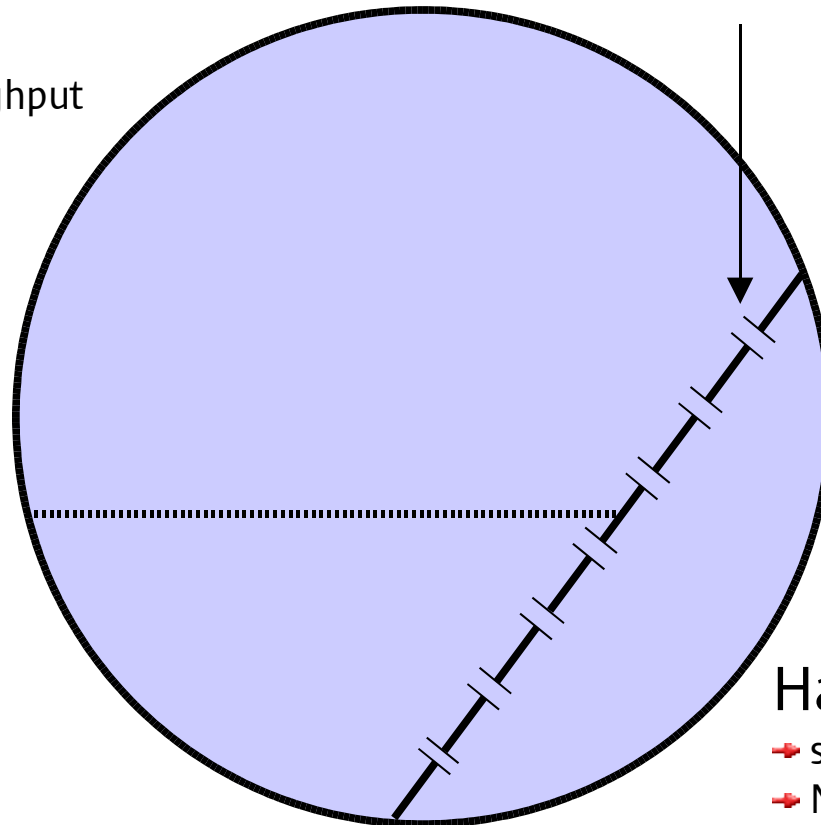


# RTSJ System Model

## Non real-time

- heap memory
- Java threads
- maximized throughput

Data transfer queues



## Soft real-time

- scoped memory
- heap memory
- Realtime threads
- jitter TBD

## Hard real-time

- scoped memory
- NoHeapRealtime threads
- jitter  $\pm 50 \mu\text{s}$

# Project Golden Gate

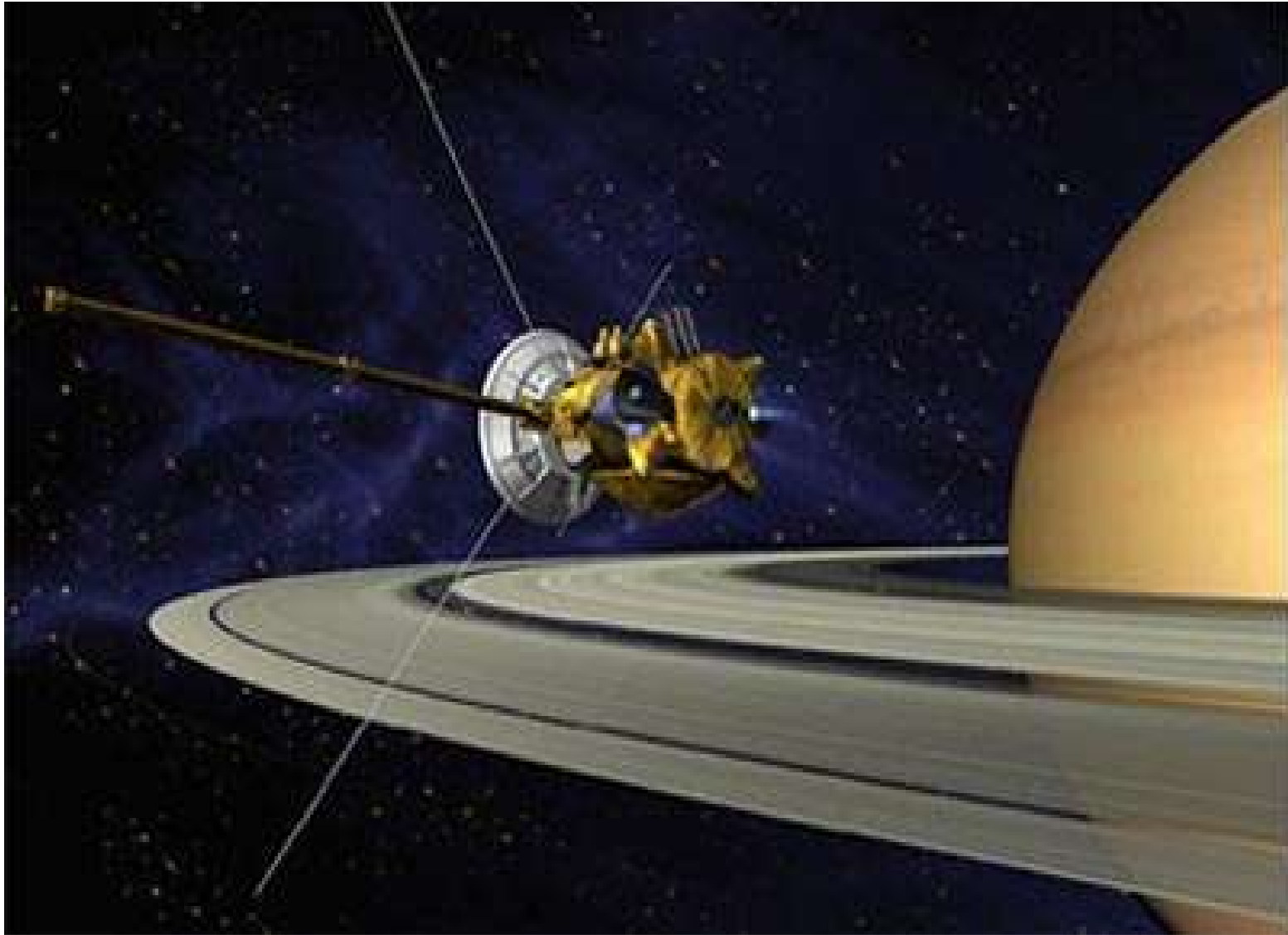
- **Sun Microsystems  
Laboratories**
- **Caltech's  
Jet Propulsion  
Laboratories (JPL)**
- **Carnegie Mellon  
University West**



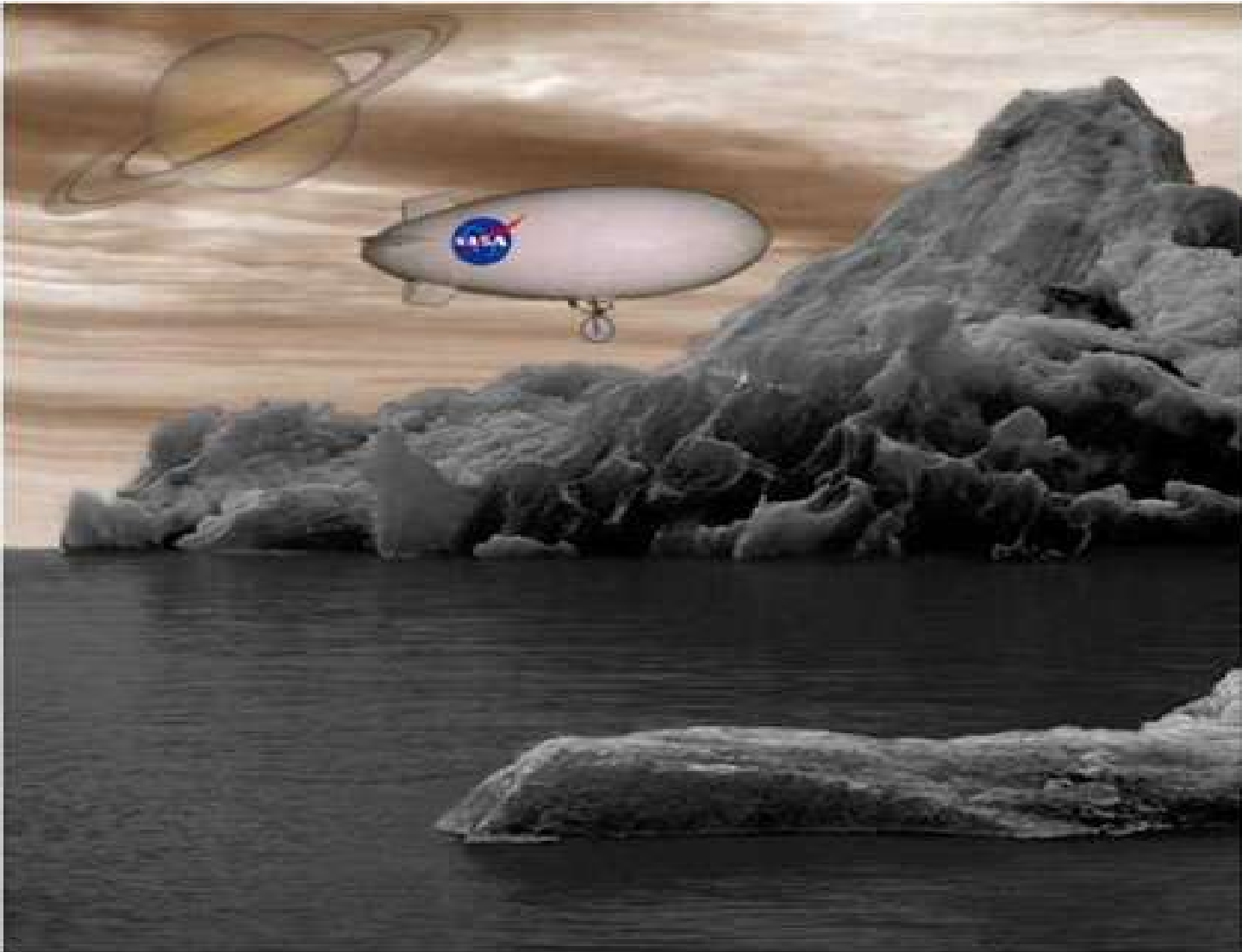
# Golden Gate Overview

- **Implement JPL's Mission Data System using the RTSJ**
  - component-based software architecture for spacecraft
- **Use the implementation for Mars Science Laboratory**
  - launch october 25th, 2009
- **Competes in the decision phase (2005) with:**
  - C++ implementation of MDS
  - NASA's traditional software development methodology

# Cassini-Huygens Mission



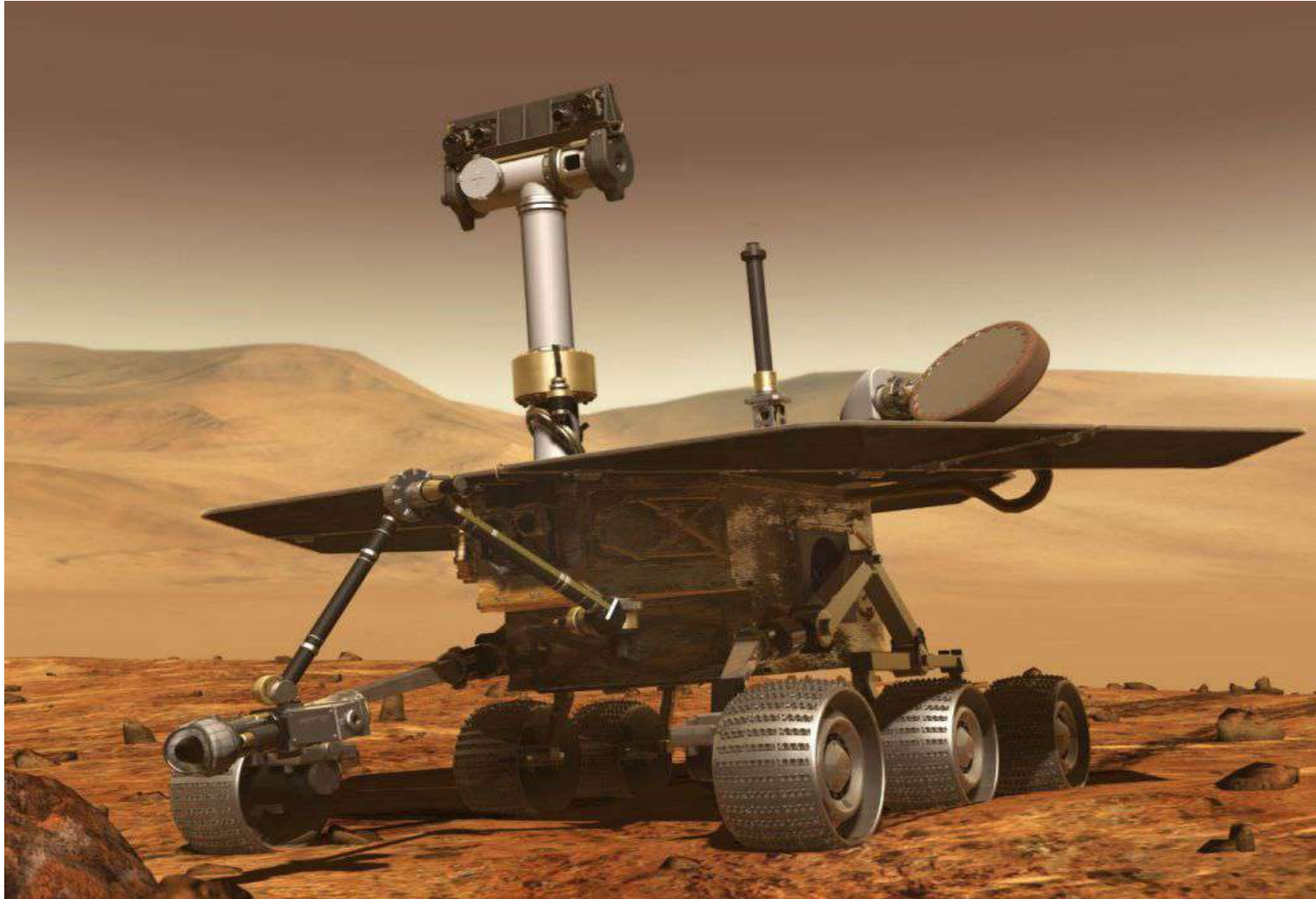
# Titan explorer



# Hydrobot in Europa Ocean



# Mars Exploration Rover



# The World of Side Effects

- **Turning on a disk drive has the following side effects**
  - it reduces available power
  - it causes heating, vibration, electromagnetic radiation
  - it impacts rotational torque
  - it stabilizes orientation around axis of rotation
- **On earth, these side effects are negligible**
- **In a spacecraft, every side effect is significant and must be managed**

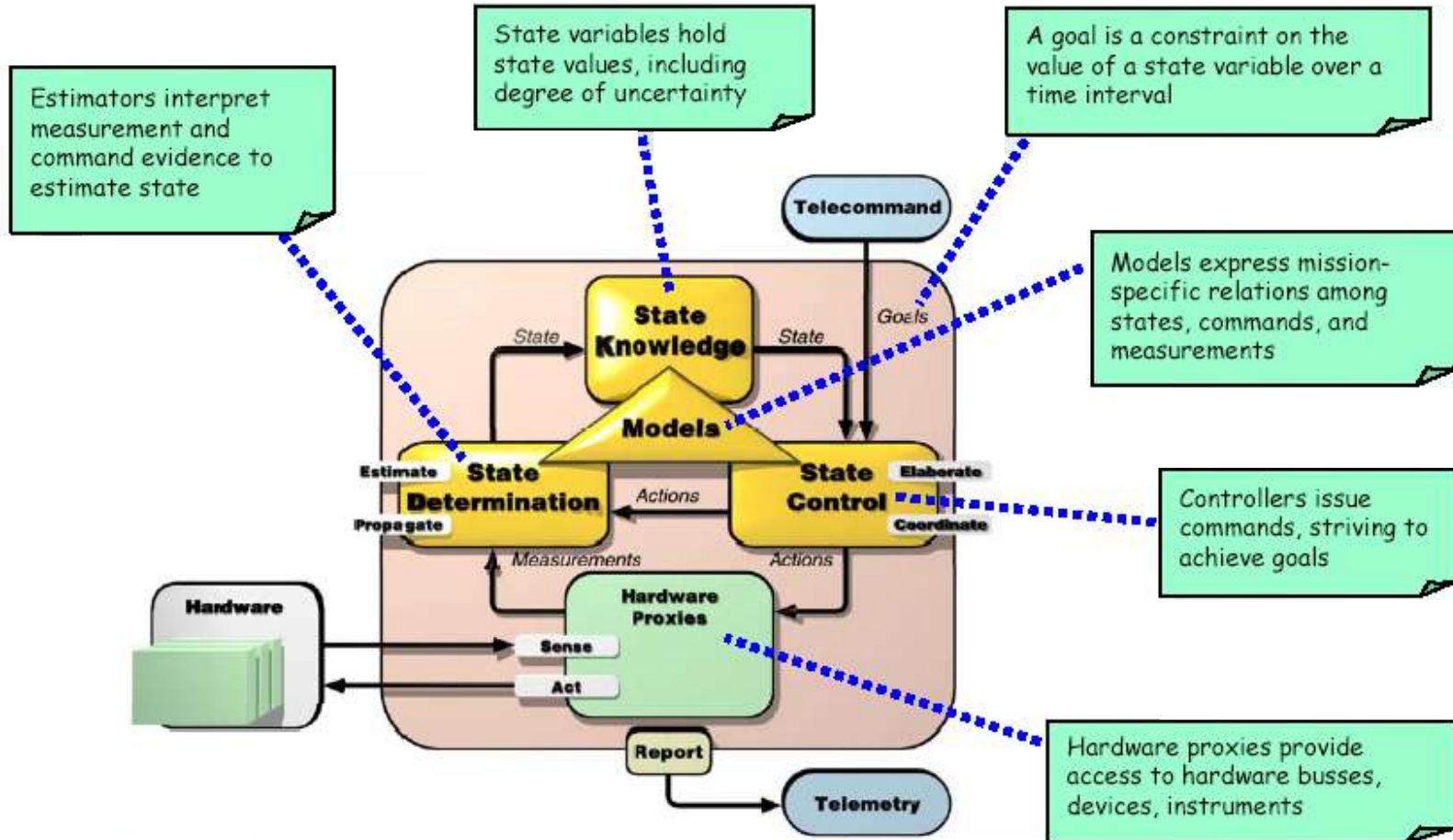
# Software engineering complexity

- **“Side effects” (couplings) are everywhere**
  - ➔ physics has no respect for our mental simplifications
  - ➔ we can’t ignore couplings in some control systems
- **Small-scale, low-level implementation strategies have to give way to high-level, model-based, robust methods applied system-wide**
  - ➔ complex system interactions require the software to reason about the system

# Mission Data System

- **State-model based software architecture for systems comprising, sensors, control, actuators, scientific computing, and complex interactions among components**
- **“Everything measurable has state”**
- **Models define legal, predictable, illegal, and possible state transition**

# State/Model architecture



# Example spacecraft states

- **Dynamics**
  - ➔ vehicle position & attitude, gimbals angles, wheel rotation
- **Environment**
  - ➔ ephemeris, light level, atmospheric profiles, terrain
- **Device status**
  - ➔ configuration, temperature, operating modes, failure modes
- **Parameters**
  - ➔ mass properties, scale factors, biases, alignments, noise levels
- **Resources**
  - ➔ power & energy, propellant, data storage, bandwidth
- **Data product collections**
  - ➔ science data, measurement sets
- **Data management policies**
  - ➔ compression/deletion, transport priority
- **Externally controlled factors**
  - ➔ space link schedule, configuration

# Example spacecraft models

- **Relationships among states**
  - ➔ power varies with solar incidence angle, temperature, occultation
- **Relationships between measurement values and states**
  - ➔ temperature data depends on temperature, but also on calibration parameters and transducer health
- **Relationships between command values and states**
  - ➔ it can take up to half a second from commanding a switch to full on
- **Sequential state machines**
  - ➔ some sequences of valve operations are okay; others are not
- **Dynamical state models**
  - ➔ accelerating to a turn rate takes time
- **Inference rules**
  - ➔ if no communication from the ground in a week, assume the uplink has failed
- **Conditional behaviors**
  - ➔ pointing performance can't be maintained until rates are low
- **Compatibility rules**
  - ➔ reaction wheel momentum cannot be dumped while being used for control

# JPL Golden Gate Goal

**“Retire the risk of using Java for flight software”**

# Project Mackinac

- **Sun Microsystems Laboratories**
- **Sun Global Sales Organization**
- **External industrial partner**



# Mackinac Overview

- **A 'micro' business unit funded by CTO based in the Labs**
- **Implement the RTSJ to product level quality**
- **Use customer applications as the driving requirements**
- **Work closely with industrial partners in early access programs**

# Customer Application Reqs

- **Development driven by a customer (real) application**
  - Fossil-fuel power-generation plant control system
  - 4 nodes per plant, 80 plants per year
- **Application requirements**
  - 16 ms period, 500  $\mu$ s max latency jitter
  - 3000 method calls per period
  - 80 ms max for fault detection and failover
  - additional non-real-time activity

# Mackinac stack

- **HotSpot 1.4 JVM**
  - ➔ modified to conform to the RTSJ
  - ➔ RTSJ TCK compliant (JSR01)
- **Solaris/SPARC platform**
  - ➔ Solaris 9 operating environment
  - ➔ enhanced real-time capabilities via extra kernel modules
  - ➔ may require additional kernel hooks
  - ➔ optional support for high-availability

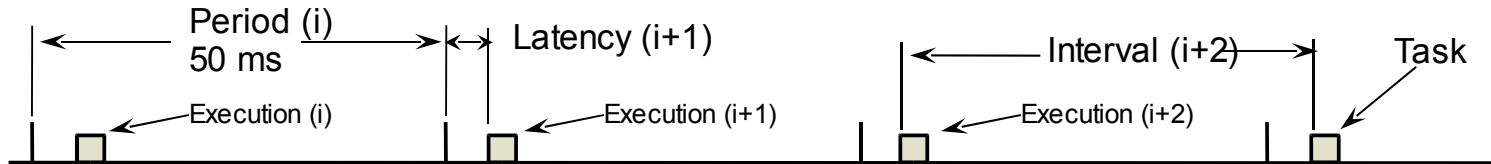
# Objectives

- **Expected latency jitter in the tens of microseconds**
- **Very little performance degradation for non-real-time activities**
- **Early alpha release mid 2004**

# RTpresto

- **Test framework and test cases to measure RTSJ implementation temporal predictability**
- **Created by Sun Labs interns under guidance by Greg Bollella**
- **Included in the JPL ‘Suramandu’ test suite**
  - ➔ to be open sourced through the Open Group
- **Used for Mackinac daily performance**

# Temporal predictability



|  |                               | NoHeapRealtimeThread        |
|--|-------------------------------|-----------------------------|
| Without<br>Garbage<br>Producing<br>Threads | Interval Max / Min            | 50.0015 ms / 49.9986 ms     |
|  | Interval Jitter (max - min)/2 | 1.5 $\mu$ s                 |
|  | Interval Mean / Std dev.      | 50.000 ms / 387 ns          |
|  | Latency Max / Min             | 15.1 $\mu$ s / 12.8 $\mu$ s |
|  | Latency Jitter                | 2.3 $\mu$ s                 |
|  | Latency Mean / Std dev.       | 13.7 $\mu$ s / 253 ns       |
| With<br>Garbage<br>Producing<br>Threads    | Interval Max / Min            | 50.0064 ms / 49.9939 ms     |
|  | Interval Jitter (max - min)/2 | 6.25 $\mu$ s                |
|  | Interval Mean / Std dev.      | 50.000 ms / 549 ns          |
|  | Latency Max / Min             | 19.7 $\mu$ s / 12.8 $\mu$ s |
|  | Latency Jitter                | 6.8 $\mu$ s                 |
|  | Latency Mean / Std dev.       | 14.4 $\mu$ s / 487 ns       |

# Mackinac Goal

- **Demonstrate Sun and Java can meet the challenge of industrial computing**
  - “Industrial Strength Java”
- **Applications**
  - telecom, transportation, military, government, manufacturing (factory), financial, automotive...

# Project Tancarville

- **Sun Microsystems Laboratories**
- **Major avionics vendor**



# Objectives

- **Completely rethink JVM implementation**
  - ➔ re-invent the JVM for DO-178B certification by the FAA for use in safety-critical systems on commercial aircraft
  - ➔ invent a formalism for verifiable correctness (functional and temporal)
  - ➔ prove the existence of identical semantics at the program interface to the JVM across multiple host and target platforms
- **Two positions open in Sun Labs Europe**



# Questions?

[christophe.lizzi@sun.com](mailto:christophe.lizzi@sun.com)



*Sun*  
microsystems  
We make the net work.