



# Squawk: Enabling Java™ on small devices

**Eric Arseneau**  
Principal Investigator  
Sun Microsystems



**2007  
Sun Labs  
Open House**



# Goal of this Talk

*Learn what Squawk is and how it intends to ease the development of new smarter products through the enabling of Java™ technology on small devices*

# Agenda

Design Goals of Squawk

Why Yet Another JVM ?

Design Overview

Progress

Future

# Design Goals of Squawk

- Optimize for small devices
- Java™ technology enabler
- Java™ in Java™
- Java™ on Java™
- Ease of application development

# Design Goal: Optimize For Small Devices

- Small: A class of device that has limited resources
- Type of resources
  - Memory
    - FLASH/ROM
    - RAM
  - Power/Energy
  - Processing
  - Hardware components

# Design Goal: Java™ Technology Enabler

- Lower the cost of porting and running Java™ technology
- Expand the types of devices able to leverage Java™ technology
  - > From low power CPUs to Micro-controllers
- JME IMP compliant portable VM
- Java™ technology on bare metal
  - > No OS required



# Design Goal: Java™ in Java™

- Write VM itself in the Java™ language
- Use Java™ -> C translation to get native implementation
- Simplifies the generation of
  - > Native VMs
  - > Optimizations, space and time
  - > Compact VMs
  - > Complete products

# Design Goal: Java™ on Java™

- Write as much of the system itself in Java™ language
- Limit required native code to bare minimum
  - > Bootloader
  - > Low level glue-code
- Implement OS services in Java™ language
- Reduce number of native methods
- Device drivers in Java™ language

# Design Goal: Ease of Application Development

- Bring Java™ technology to the embedded space
- Support standard tools
- Leverage existing Java™ technology expertise
- Can run on desktop/server and devices
- Enable Java™ technology end to end

# Design Goal: Ease of Product Development

- Hardware
  - > Enabling Java™ technology on platform of choice
  - > Reduce hardware requirements
- Software
  - > Java™ language
  - > Java™ tools
  - > Squawk OS services

# Agenda

Design Goals of Squawk

Why Yet Another JVM ?

Design Overview

Progress

Future

# Why Another Sun JVM Option ?

- Most of our VMs and Java platforms are geared towards dynamic distribution of applications
- Must include every feature of platform
- Supports the distribution of Java™ application
- HotSpot
  - > Great for server/workstation with little regard for resource consumption
- CLDC HotSpot/Monty
  - > Great for devices that are charged regularly and where wireless communication is desired

# Why Another Sun JVM Option ?

- Squawk is geared towards a static set of applications on device
- Extending set of applications requires an external agent
- Tailoring Java to manufacturing
- What is distributed to the end user is a “thing” that happens to be using Java™ technology
- Embedded CPUs and micro-controllers far outnumber volume of general CPUs

# Why Another Sun JVM Option ?

“If you could fit that platform into low end MCUs that are used in the toy industry, everyone would adopt it for convenience, for its speed to market and its upgradeability.”

**Davin Suffer**  
CTO

WOWWEE Corporation

# Agenda

Design Goals of Squawk

Why Yet Another JVM ?

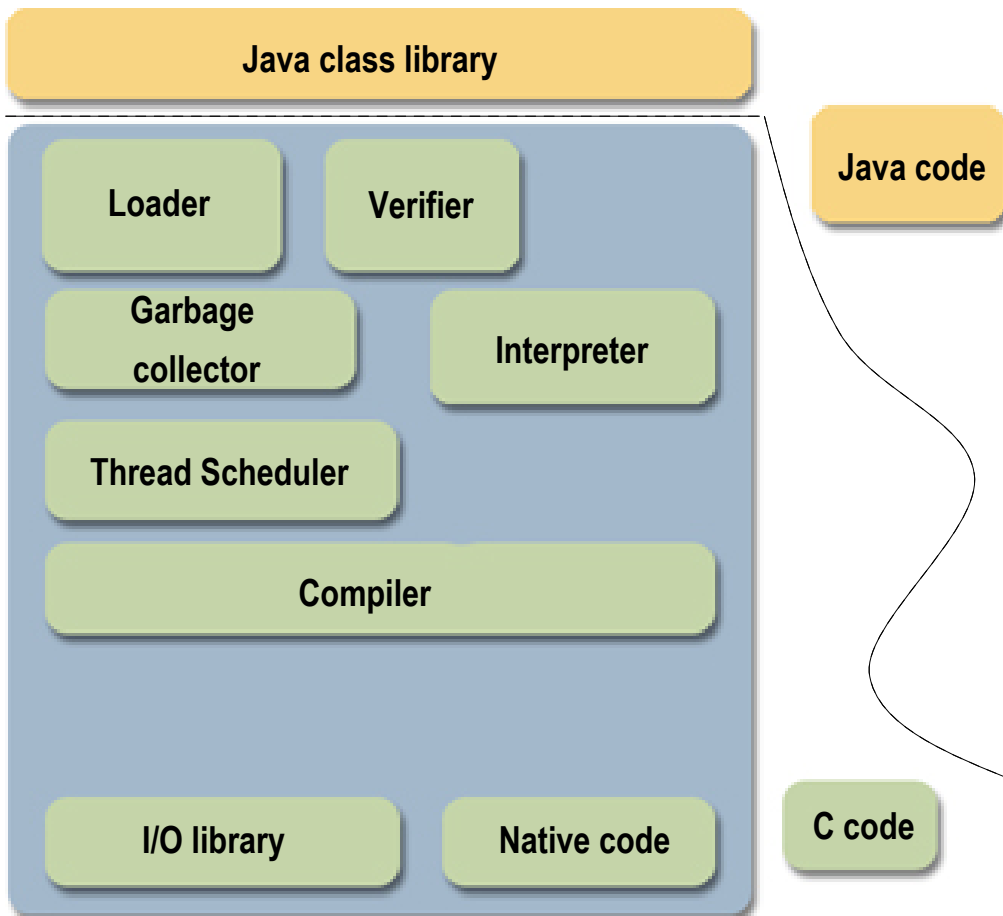
Design Overview

Progress

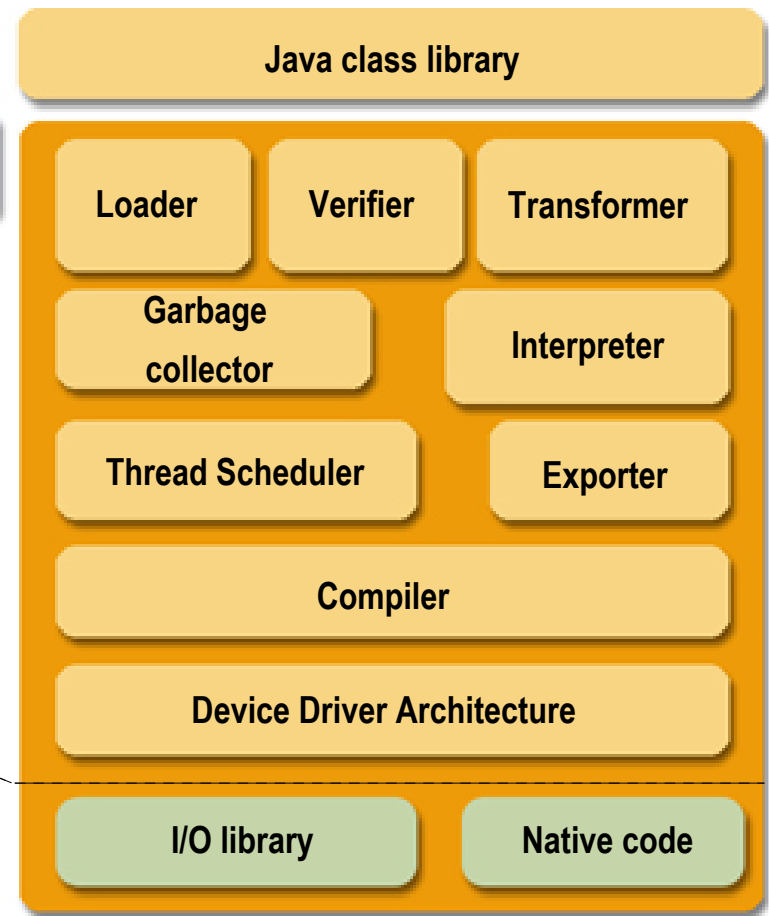
Future

# Design Overview: Standard JVM vs Squawk JVM

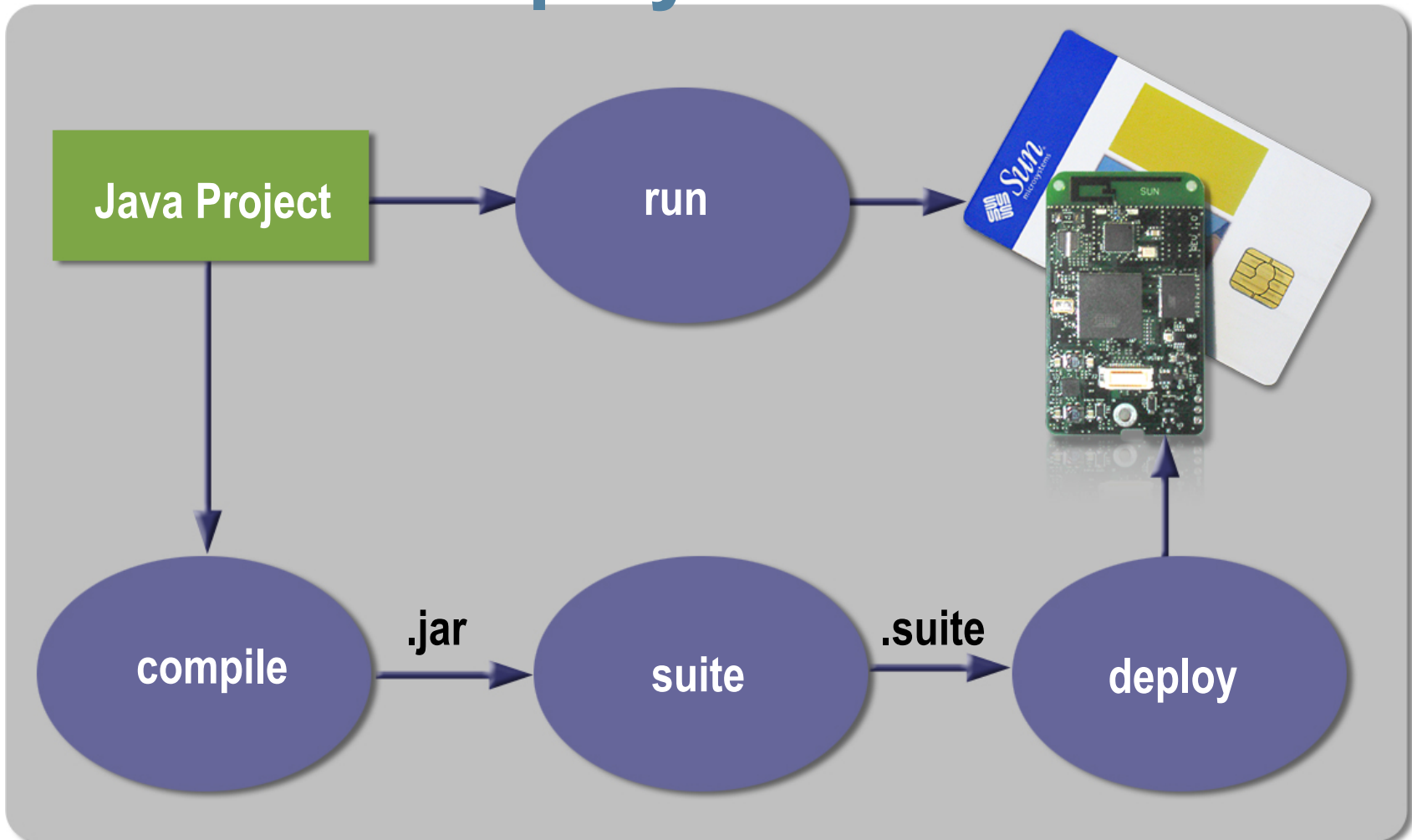
## Standard JVM



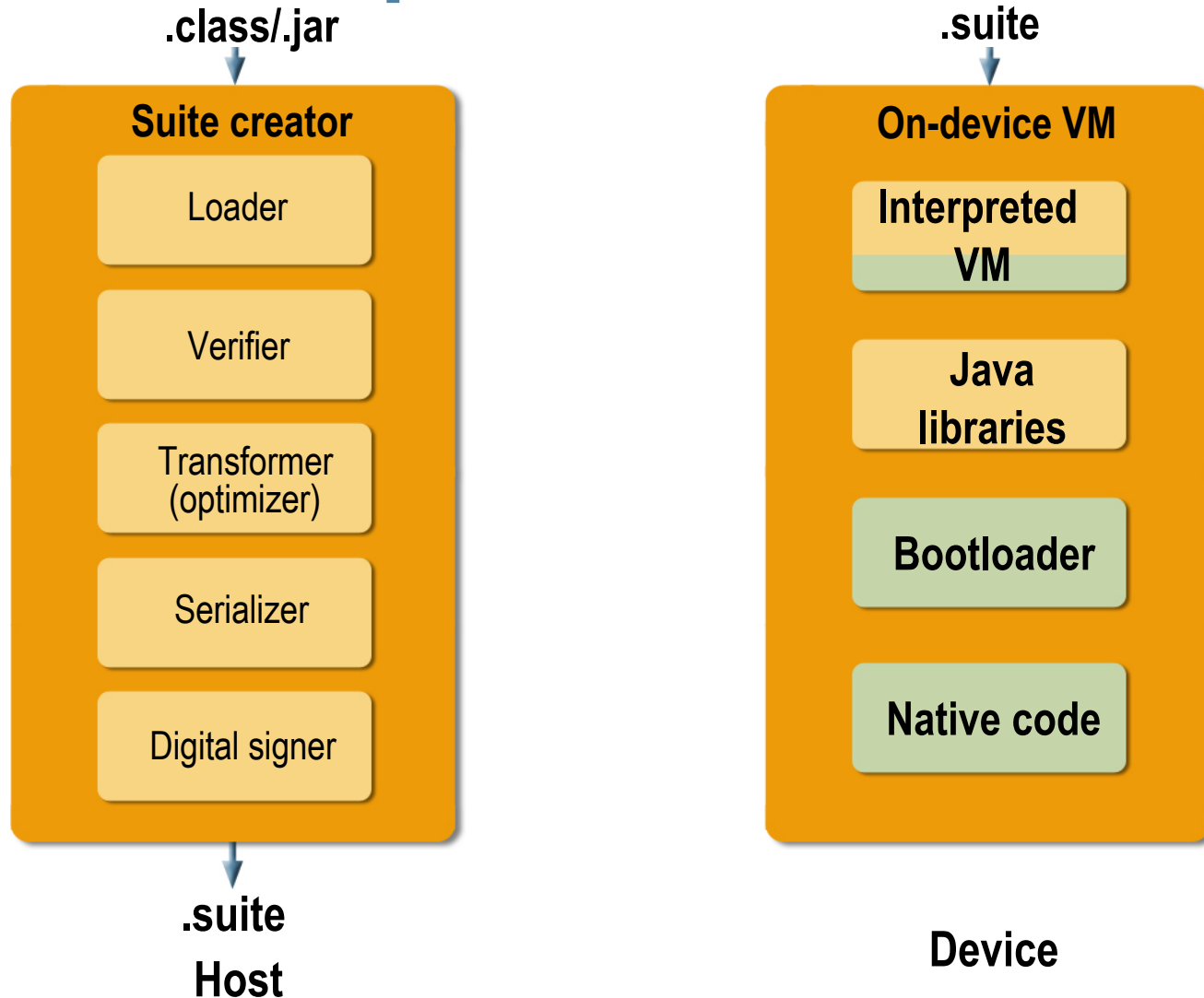
## Squawk JVM



# Design Overview: Build and Deploy Process

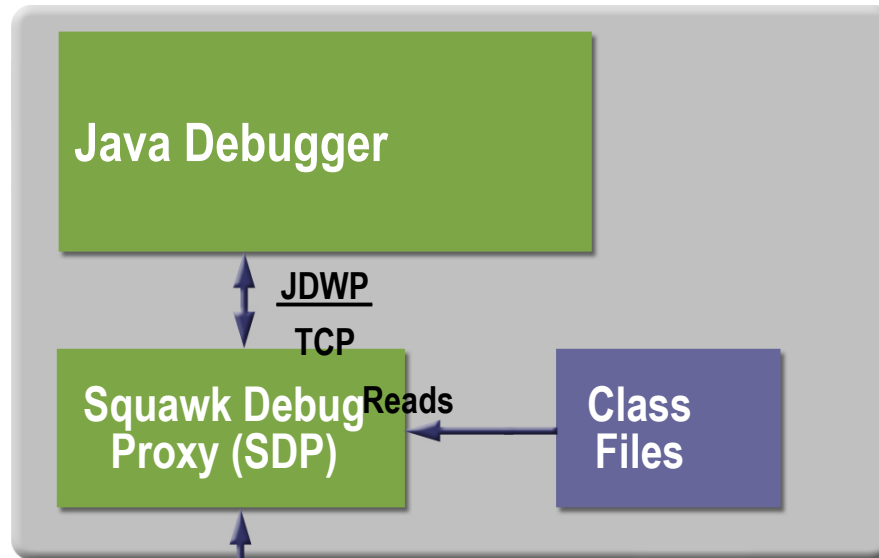


# Design Overview: Squawk's Split VM Architecture

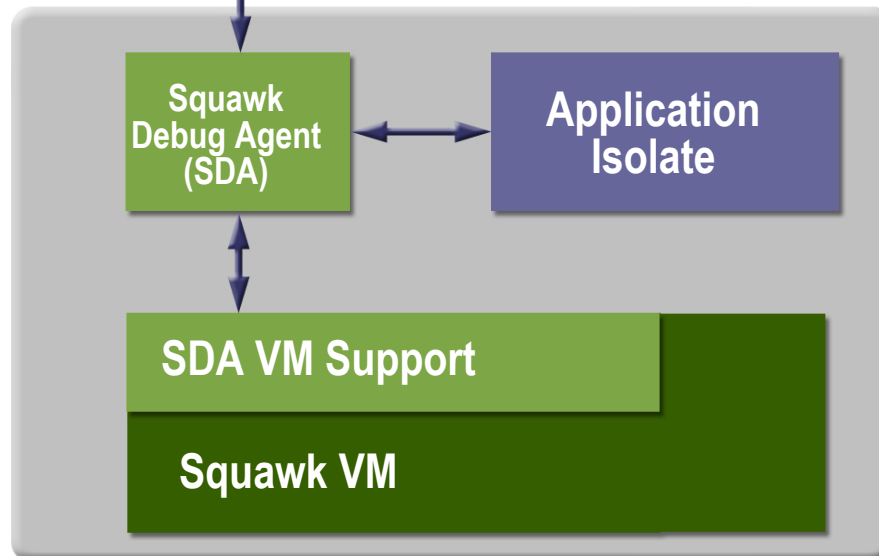


# Debugger

**Developer Workstation**



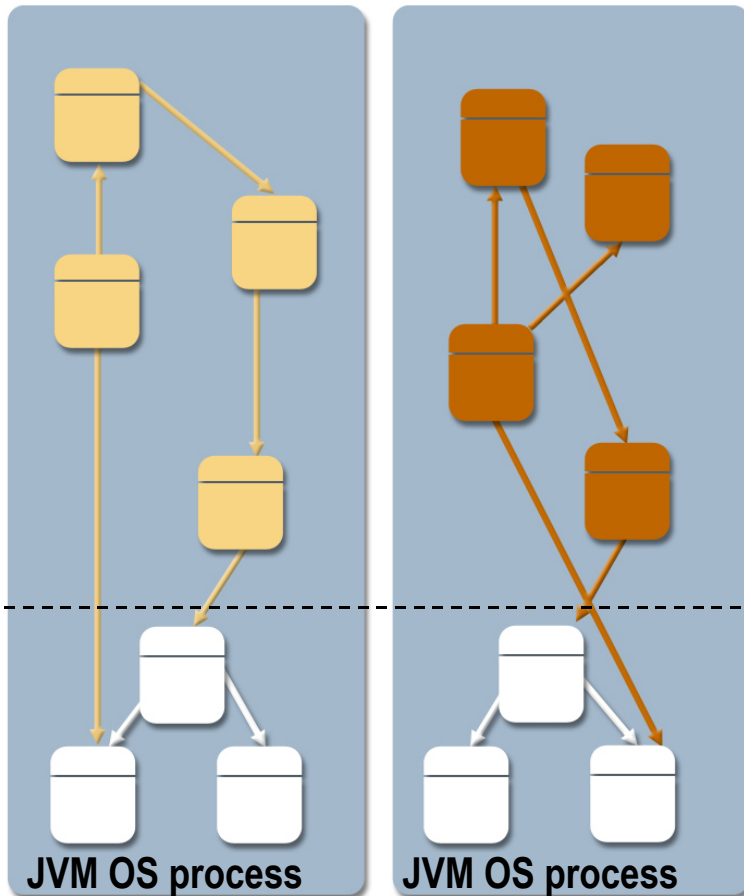
**Sun SPOT**



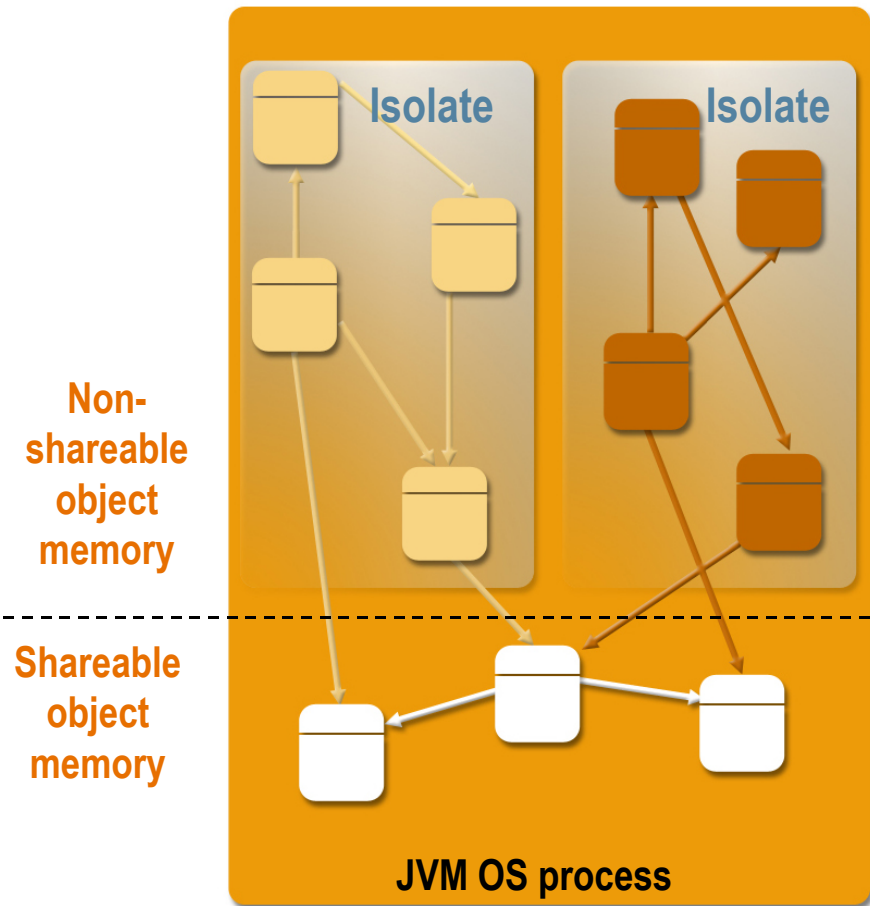
**SDWP/Wireless**

# Design Overview: Multiple Isolates on the One JVM

## Standard JVM

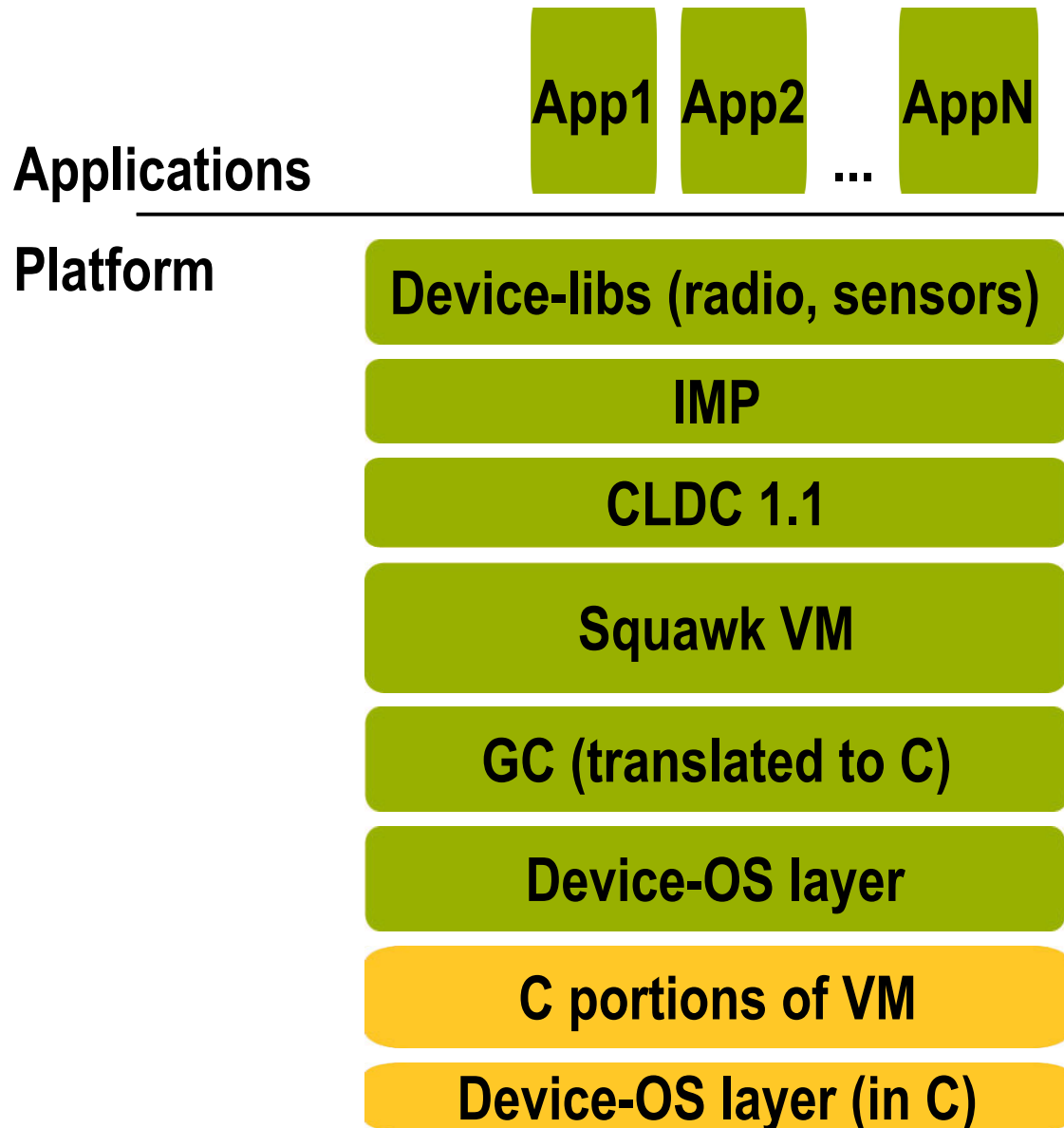


## Squawk JVM



# Design Overview: Trimming

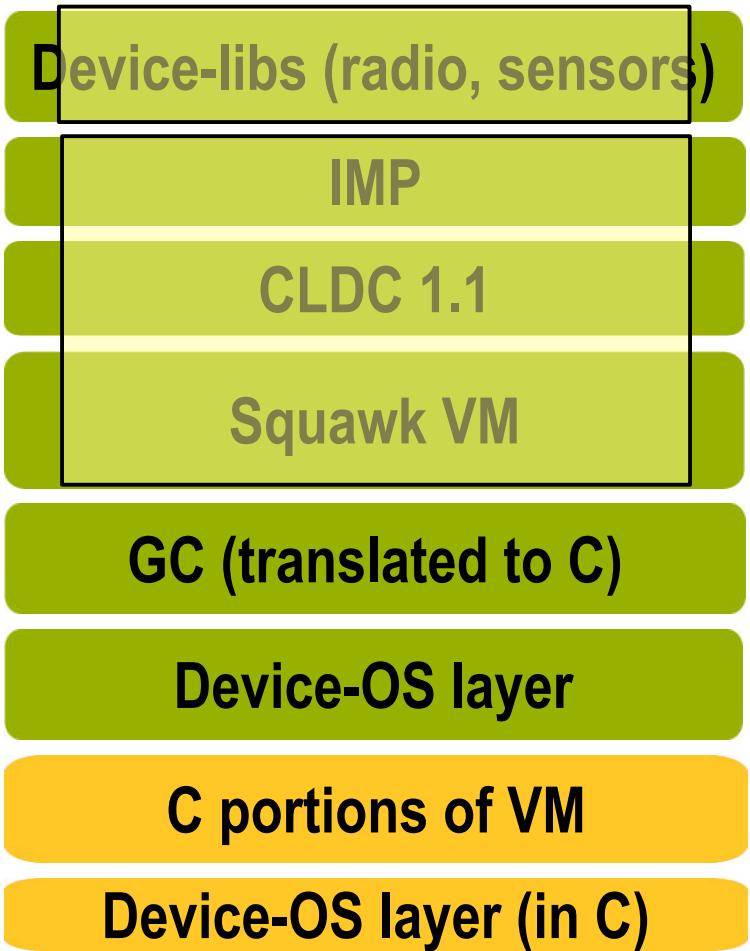
- We can close off set of classes at suite creation time
- Device is closed to new code until it is connected to a workstation
- This allows us to trim unused parts of an application



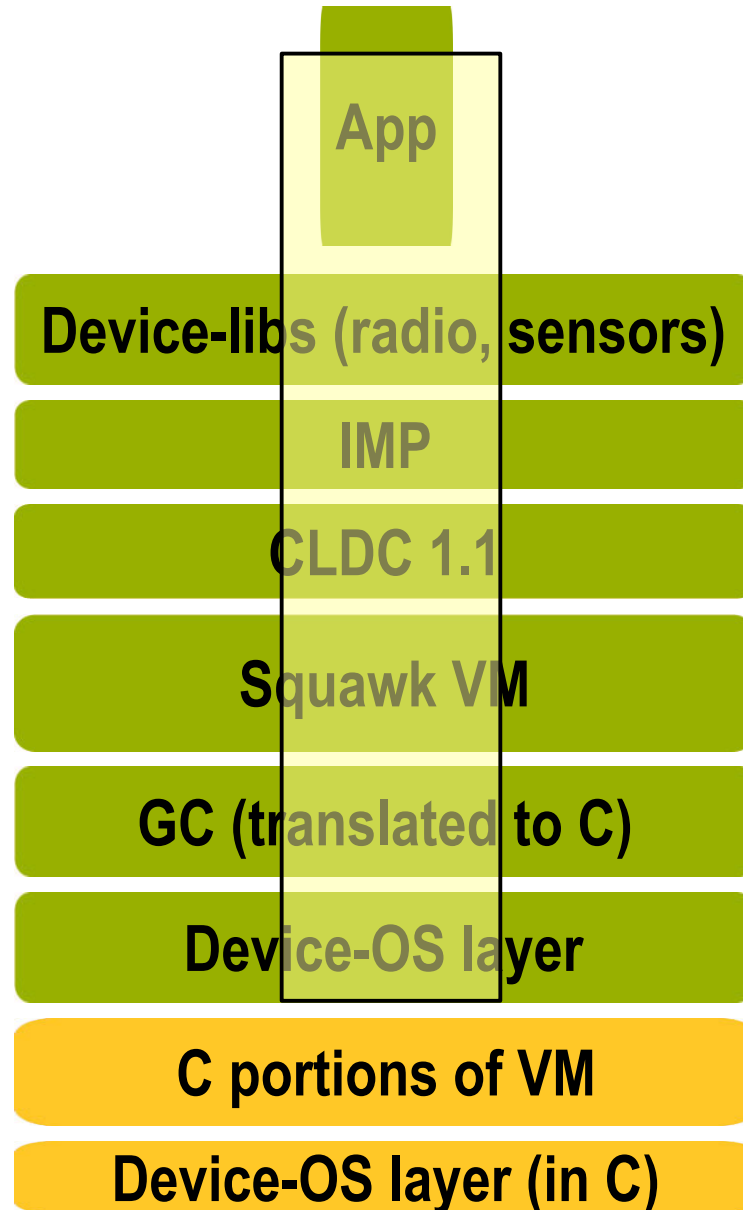
# Trimmed Applications



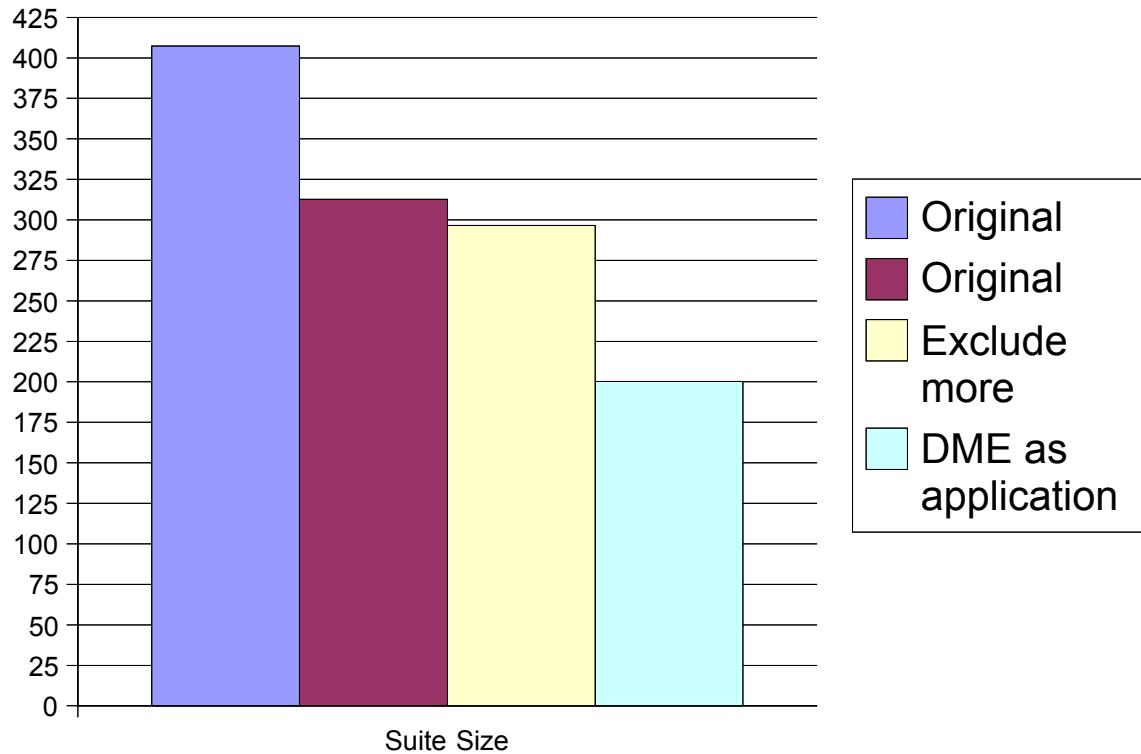
# Trimmed Platform



**Trimmed**



# Design Overview: Trimming Results



# Agenda

Design Goals of Squawk

Why Yet Another JVM ?

Design Overview

Progress

Future

# Progress: Platforms Ported To

- Solaris SPARC/x86
- Linux
- Mac OS X PPC/x86
- Windows
- Sun SPOT



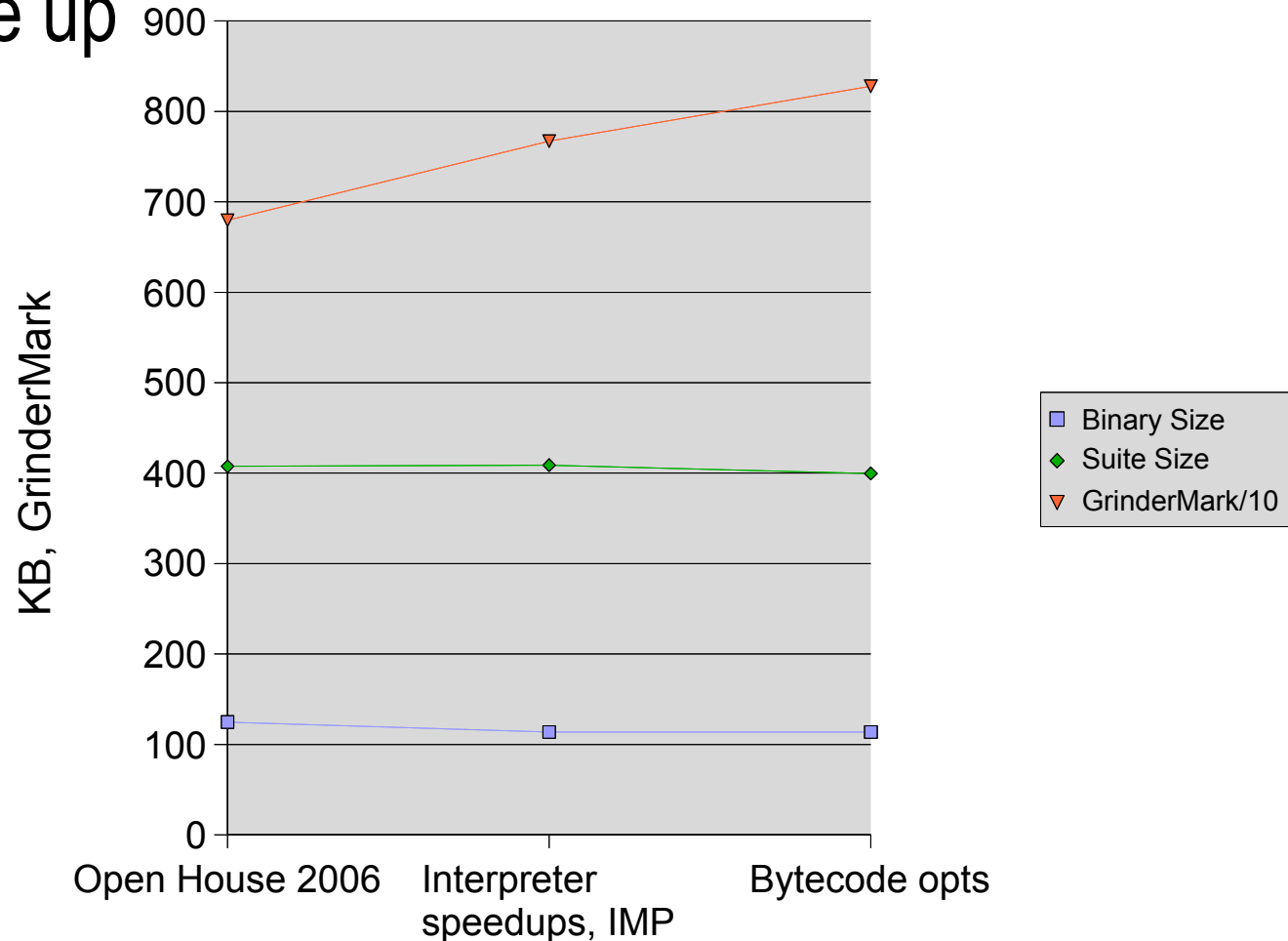
# Progress: This Year

- JME IMP compliance
- Universal Emulator Interface
- Dead method elimination
- Byte code optimizations
- Interpreter optimizations
- Improved debugger
- Most of the year spent focusing on single major client
  - > Project Sun SPOT



# Progress: This Year

- Performance up
- Size down



# Agenda

Design Goals of Squawk

Why Yet Another JVM ?

Design Overview

Progress

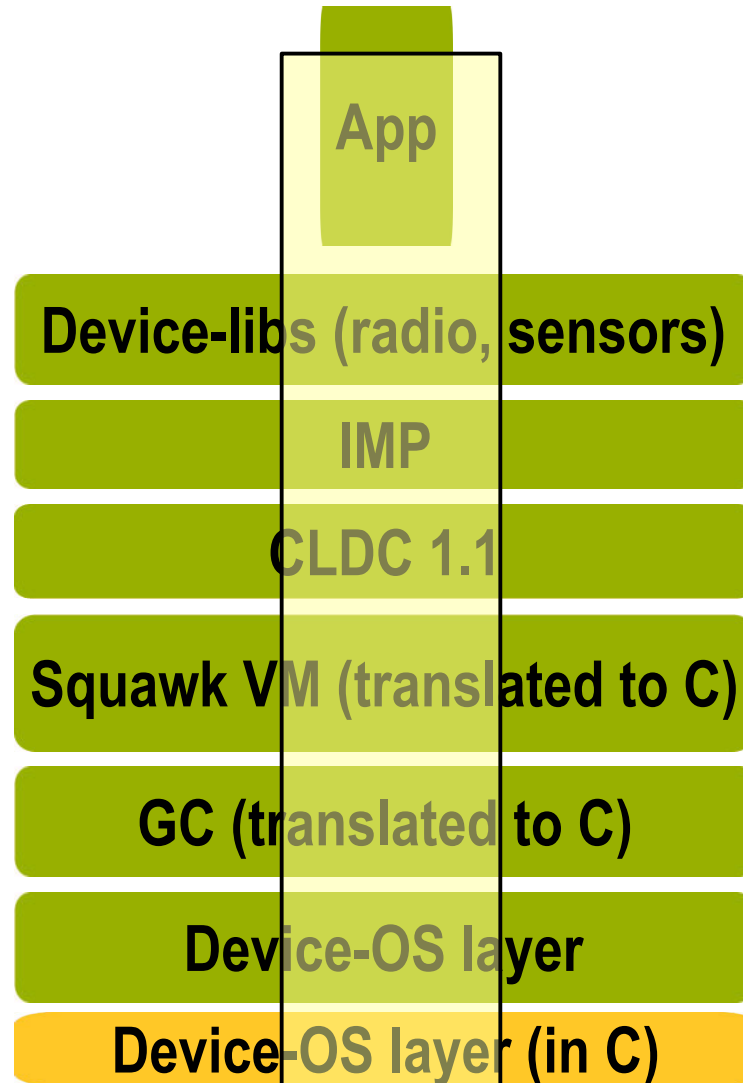
Future

# Future

- Complete Java™ in Java™
- Complete generation of VM
- Extend trimming
- Introduce ahead-of-time compilation to application code
- Subset of RTSJ
- Port to new embedded platform
- Work out licensing options

# Future

Trimmed



# For More Information

- Come and talk to us one on one at our demo station in back of room 1431 today and tomorrow
- <http://research.sun.com/projects/squawk>



**Eric Arseneau**  
eric.arseneau@sun.com



**2007  
Sun Labs  
Open House**

