Emerging Technologies for the Enterprise

March 28-29, 2007 Drexel University, Philadelphia, PA

# Apache Harmony

## Implementing Java SE in open source

Geir Magnusson Jr

Apache Software Foundation
Joost

# Presentation Goal

Teach you about motivation, history, status and future of the Apache Harmony project

# About the Speaker

- Geir has been a professional developer for 22 years.
  - Currently VP, Engineering, Joost (http://www.joost.com)
  - Previously at Intel, IBM, Gluecode, Adeptra, FitLinxx and Bloomberg
- Co-Founder of Apache Harmony, Chairman of the Harmony PMC
- Apache Representative to the Java Community Process SE/EE Executive Committee
- Other open source experience includes co-founding Apache Geronimo, Director of the ASF, major author of Apache Velocity

# Agenda

- Overview, Motivations and Goals
- Project History
- Project Status
  - Tour of the Project
  - Modular Architecture
  - How are We Doing?
- Relevance in the Age of OpenJDK
- Summary
- Q&A

# Overview, Motivations and Goals

# Overview

## Project of the Apache Software Foundation

- Founded May, 2005 in Apache Incubator
- Became Apache Harmony Project in Oct 2006

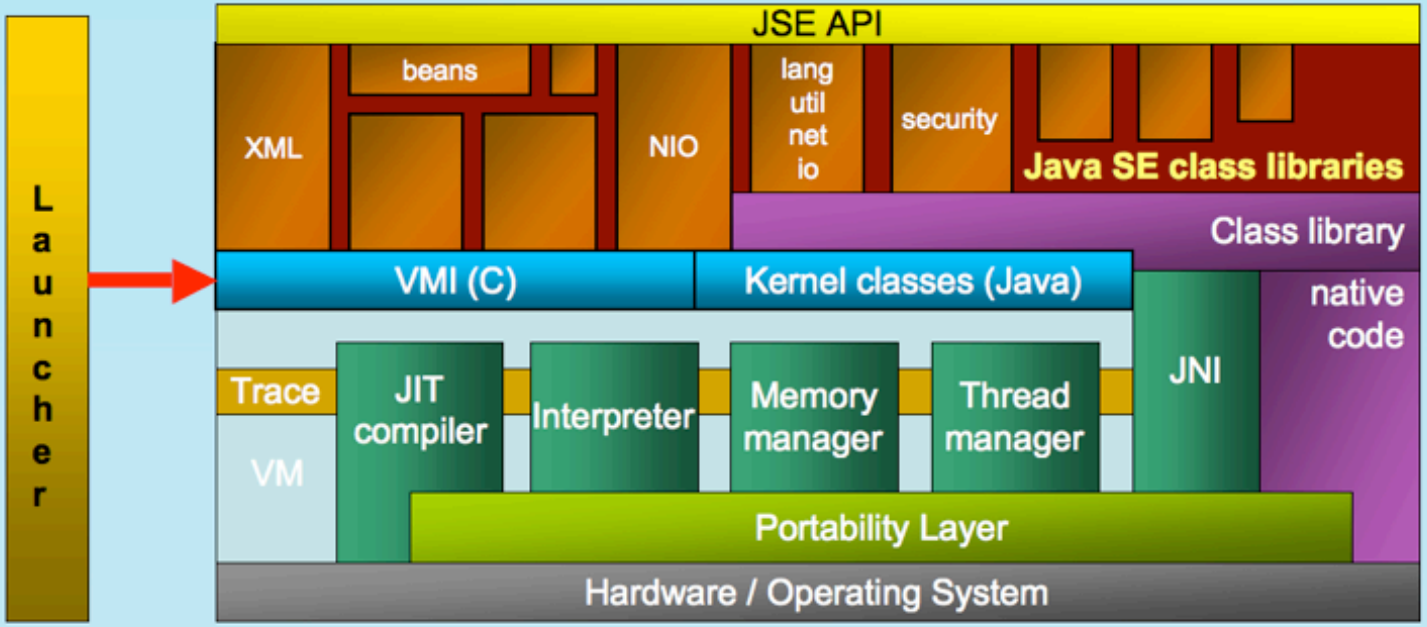
**Primary Goal - *full implementation* of Java SE**

- Compatible class library
- Competitive virtual machine
- Full JDK toolset

# What are we doing?

**Single launcher can switch between different JVM providers:**
*java -vm<vm_name> ...*

**Class library constructed from 20+ OSGi bundles!**




**Simple JVM interface to Harmony's class libraries**


**Portability Layer encapsulates OS functionality with platform independent interface**

# Motivations

Enable continued adoption of Java via portable code under a liberal license

# Motivations

## Create single open collaborative community

(That doesn't mean project consolidation - but inter-project communication, collaboration and cooperation)

# Motivations

Provide an acceptably-licensed Java platform for Linux and other FLOSS communities

# Motivations

Ensure Java can be a preferred platform for FLOSS-mandated IT

# Goals

Create implementation of Java SE 5

- *Available under the Apache License*
- *Independent of any encumbered code*
- *Complete, compatible, passes TCK*

# Goals

Community-developed *modular architecture*

- *implementations can share components*
- *enable innovation in components*

# Goals

## Protect IP rights of ecosystem

- *Sun's source was available under non-FLOSS license*
- *Ensure Sun's rights are protected*
- *Ensure our codebase stays independent of any source with incompatible license*

***As Sun delivers code to OpenJDK, we will re-evaluate.***

# Project History

Broad discussion beforehand

- *Inside ASF*
- *Outside in the community*
  - Corporate : Intel, IBM, Sun, Oracle, HP, BEA
  - FLOSS : GCJ, GNU Classpath, Kaffe, IKVM
  - Research  and academics

# Project History

May 18th, 2005 : Harmony accepted by Incubator

Summer 2005 : community discussion

- *Technical Issues - Defined modules, languages for classlib and VM*
- *Legal and IP Issues - License issues, contributor exposure to Sun's class library*
- *Project Governance Issues - Acceptable prior access, acceptable provenance for code contributions*

# Project History

Since then : code donations and development, community building

- *Three virtual machines*
  - DRLVM, JCHEVM, BootVM
- *Class library contributions*
  - Core classes, tests from IBM
  - Security, beans, Swing/AWT from Intel
  - Math & RMI from Cordoba Institute of Technology
- *IBM J9 and BEA JRockit VMs made available for project use use*
- *Build/test continuous integration infrastructure*

# Project History

## May 2006 : At JavaOne

1. Harmony demonstrates Swing/AWT

2. Sun announces
   - *Distribution License for Java*
   - *Intent to place their Java SE impl under OSS license*

# Project History

## July 2006 : Our first JRE snapshot

- *DRLVM + class library for Linux and Windows x86*
- *Can run Tomcat, Eclipse, other applications*
- *http://harmony.apache.org/downloads.html*

```
$ bin/java -version
java version "1.5.0"
pre-alpha : not complete or compatible
svn = r447761, (Sep 19 2006), Windows/ia32/msvc 1310, release build
http://incubator.apache.org/harmony
```

# Project History

August 2006 : More details from Sun

- *javac and hotspot VM in OSS by end of year, license unknown*
- *full source available 2007, license unknown*

October 2006 : Apache Harmony graduates incubator

- *top level project of the ASF*

November 2006 : Sun's OpenJDK launches

- *javac and HotSpot under GPLv2 11/06*
- *classlibrary under GPLv2+exception in 2007*
- *some Java ME under GPLv2*

# Project History

Feb 2007 : BEA JRockit eval works with Harmony

# Tour of the Project

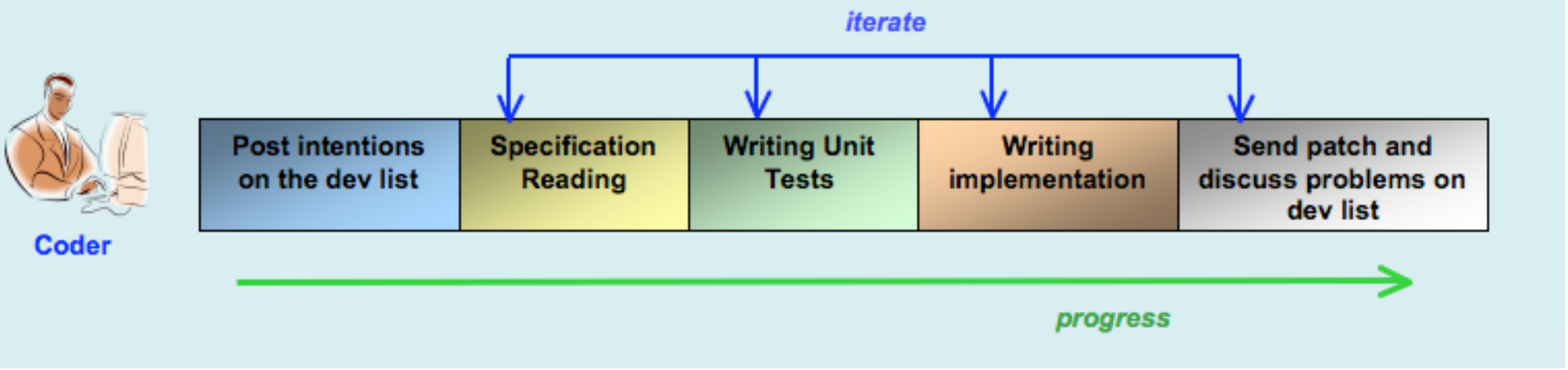
# Development Approach

We ensure an IP-clean codebase

- *Contributors detail prior access via questionnaire via the "ACQ"*
- *Developers can contribute in functional areas where they have not studied closed-source implementations*
- *Existing code being contributed to the project must provide pedigree information.*

**This is in addition to the standard Apache contribution processes**

# Development Approach



# Focus on Compatibility

We are following the Java SE specification via :

- *Java SE 5 javadoc*
- *Java Language Specification,*
- *Java Virtual Machine Specification, etc.*
- *Playing with the RI :)*

We're dedicated to testing via :

- *Our own functional/API tests*
- *Our own Integration tests*
- *Our own Unit tests*
- *Running real software*

***We are [still] trying to get the Java SE 5 JCK***

# Current Codebase

- /classlib - class library code
- /drlvm – modern VM w/ JIT - works with class lib
- /jdktools – our tools project (javac, javap, javah…)
- /trunk – federation point

# Current Codebase

| | LOC | Java | C | C++ | ASM |
|---|---|---|---|---|---|
| Classlib | 938,595 | 833,835 | 84,614 | 19,591 | 555 |
| DRLVM | 274,256 | 38,347 | 19,227 | 215,620 | 1062 |
| JCHEVM | 25,328 | 105 | 25,109 | | |
| Classlib-adapter | 4,007 | 3,634 | | 277 | |
| **TOTAL** | 1,242,186 | 875,921 | 128,950 | 235,488 | 1,617 |

(as of Sept 2006)

# /classlib

```
/classlib/trunk
   build.xml
   /deploy  -> HDK builds here or dropped in here
   /modules
      /accessibility
      /annotation
      ...
      /luni            -> lang/util/net/io module
         build.xml     -> module build script
         /src
            /main
               /java      -> all java source
               /native    -> all native source
            /test
      /jndi
```

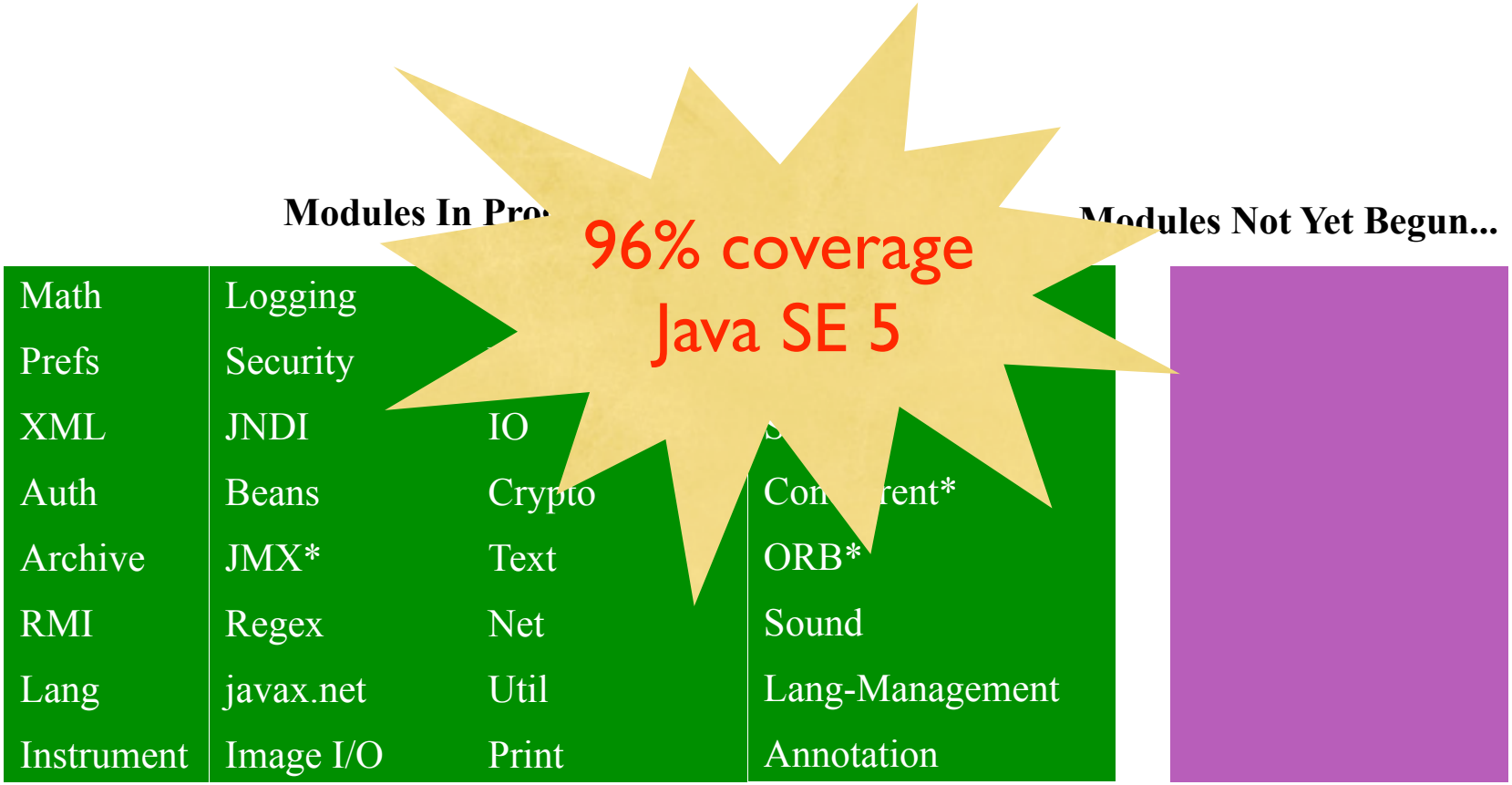
# /classlib - coverage status

**Modules In Progress...**

| | | | |
|---|---|---|---|
| Math | Logging | NIO-Charsets | Swing |
| Prefs | Security | NII-Channels | AWT |
| XML | JNDI | IO | SQL |
| Auth | Beans | Crypto | Concurrent* |
| Archive | JMX* | Text | ORB* |
| RMI | Regex | Net | Sound |
| Lang | javax.net | Util | Lang-Management |
| Instrument | Image I/O | Print | Annotation |

**Modules Not Yet Begun...**

# /classlib - coverage status



# /jdktools

```
/jdktools/trunk
  build.xml
  /deploy
    /jdk          -> where built artifacts go
      ...
  /modules
    /jpda         -> implementation of JDWP
    /tools        -> javac, javah, keytool, ...
    /launcher     -> VM launcher (aka “java”)
    /samsa        -> jdk/bin launcher
```

# /jdktools

Current toolset (javac, javah) written in java

```
/jdktools/trunk
   /deploy
      /jdk
         /bin
            java
            javah
            javac
            jarsigner
            keytool
            /data
               javac.dat
```

- How tools work :
- `jdk/bin/javac` is simple executable
  - finds on argv[0] (exe name)
  - looks up tool params in `/data`
  - invokes `jdk/jre/bin/java` with classpath and specifies class : `org.apache.harmony.tools.<toolname>.Main`

# /drlvm

```
/drlvm/trunk
   /build
      build.sh
      /deploy
         /jdk        -> where built artifacts go
            ...
   /vm
      /vmcore       -> core stuff
      /interpreter  -> bytecode interpreter
      /jitrino      -> JIT bytecode compiler
      /gc_cc        -> current garbage collector
      /gc_gen       -> next-gen garbage collector
      ...
```

# /trunk - federate this!

## Problem : How do build this thing?

(uses “trick” : `svn switch`)

```
/trunk
   build.xml
   /working_classlib  -> /classlib
   /working_vm        -> /<your favorite vm>
   /working_jdktools  -> /jdktools
   /current_resources -> /current_resources
```

- Running “ant” results in complete HDK, JDK and JRE
- Can work in any directory (`svn commit` does Right Thing)
- `ant -Dsvn.revision=X` gets you build on any SVN rev

# Modular Architecture

# Class Library Modularity

Conventional Approach

Harmony Approach

# Class Library Modularity

Conventional Approach


. . .

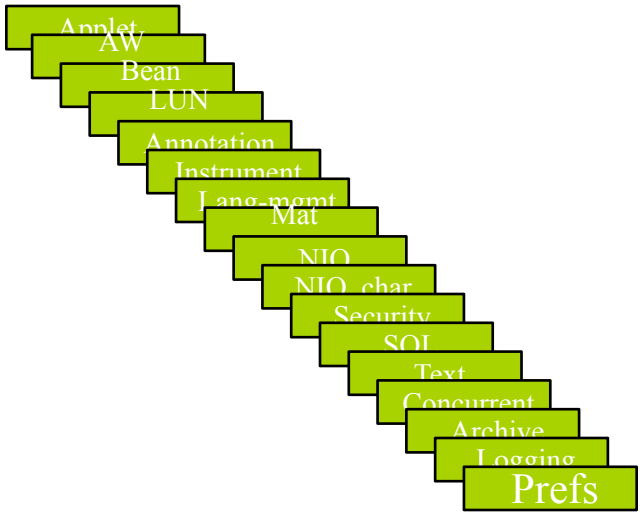
Harmony Approach

# Class Library Modularity

Conventional Approach


...

Harmony Approach

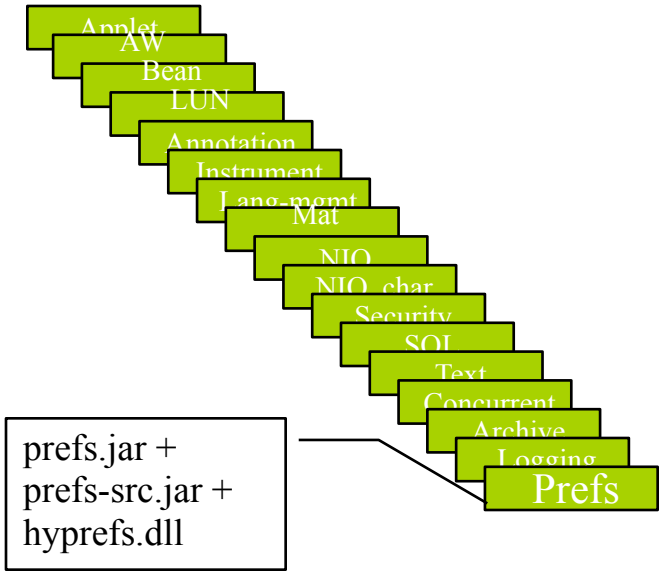


# Class Library Modularity

Conventional Approach


...

Harmony Approach



# Modularization Benefits

- Implementation componentized
- Manage prior exposure
- Freedom of assembly  (of modules)
  - *Freedom of choice for module consumers*
  - *Unit of replacement for fixes and updates*
- Facilitate contributions

# Modularization Benefit : The HDK

- Not replacement for JDK
  - HDK -> Harmony devs :: JDK -> Java devs
- Contains all files necessary for Harmony development and testing
- Enables fast rebuild of modules – Java and native
- Removes necessity to check out whole of trunk

# Modularization : Launcher

- Single program creates and invokes VM
- Allows VM choice w/ command line change

```
jre/bin/
        java.exe
        default/
                libharmonyvm.so
                harmonyvm.properties
                libhythread.so
                libhysig.so
                …
```

# Modularization : Launcher

- Can choose among multiple installed VMs :

```
jre/bin/
        java.exe
        default/
        drlvm-v1/
        drlvm-v2/
        ibm-j9/
```

```
$ java MyClass
$ java -vmdir:ibm-j9 MyClass
$ java -vmdir:drlvm-v1 MyClass
```

# How are We Doing?

# App-Oriented Testing

- Enabled by the JRE/JDK snapshot
- Goal : get the general user community to test their applications on Harmony and report what happened

http://wiki.apache.org/harmony/Application_Status

# What we test with

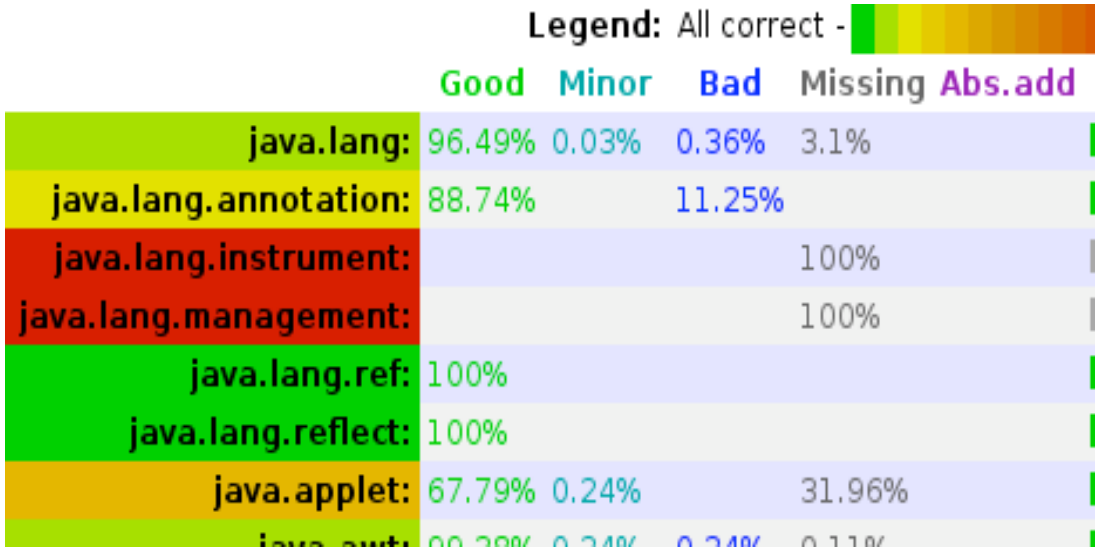
# API Coverage

- Apache Tomcat 100%
- Apache Geronimo - 96.2%
- Apache Tuscany SCA – 99.4%
- Apache Commons – passes 95+% of the 40,000+ tests
- JBoss
- Eclipse

# JAPI Reports

Produced by Stuart Ballard at:

**http://kaffe.org/~stuart/japi/**



**Legend:** All correct -

| | Good | Minor | Bad | Missing | Abs.add |
|---|---|---|---|---|---|
| java.lang: | 96.49% | 0.03% | 0.36% | 3.1% | |
| java.lang.annotation: | 88.74% | | 11.25% | | |
| java.lang.instrument: | | | | 100% | |
| java.lang.management: | | | | 100% | |
| java.lang.ref: | 100% | | | | |
| java.lang.reflect: | 100% | | | | |
| java.applet: | 67.79% | 0.24% | | 31.96% | |
| java.awt: | 99.28% | 0.24% | 0.24% | 0.11% | |

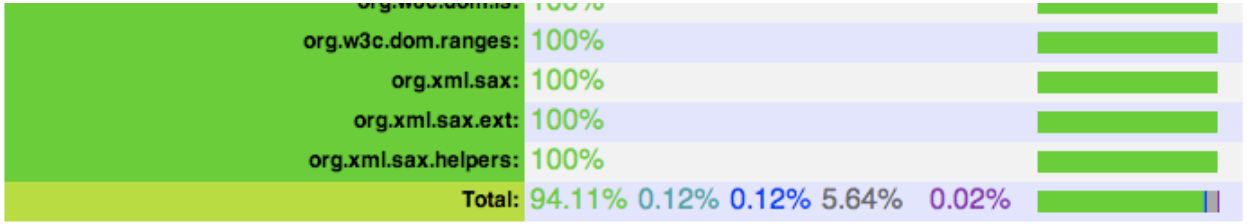
# JAPI Reports

API completeness map

# JAPI Reports

| | Good | Minor | Bad | Missing | Abs.add |
|---|---|---|---|---|---|
| java.lang: | 100% | | | | |
| java.lang.annotation: | 100% | | | | |
| java.lang.instrument: | 100% | | | | |
| java.lang.management: | 100% | | | | |
| java.lang.ref: | 100% | | | | |
| java.lang.reflect: | 100% | | | | |
| java.applet: | 100% | | | | |
| java.awt: | 100% | | | | |
| java.awt.color: | 100% | | | | |

And so on.....

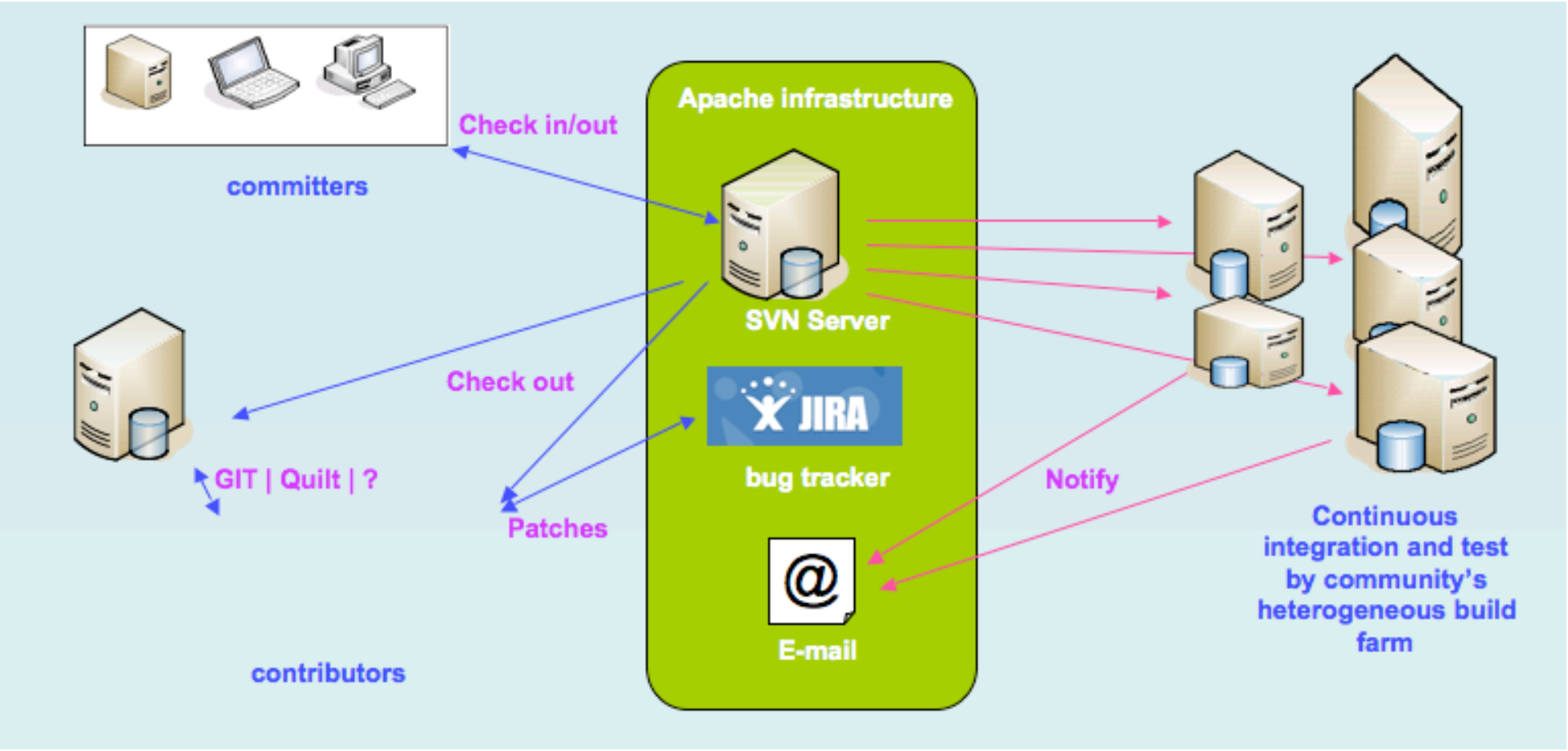


| | | | | | |
|---|---|---|---|---|---|
| org.w3c.dom.ranges: | 100% | | | | |
| org.xml.sax: | 100% | | | | |
| org.xml.sax.ext: | 100% | | | | |
| org.xml.sax.helpers: | 100% | | | | |
| Total: | 94.11% | 0.12% | 0.12% | 5.64% | 0.02% |

# Build-Test CI Infrastructure

- "CruiseControl" based automated builds and testing
- Community run  - runs on :
  - windows XP x86
  - Debian  x86
  - SUSE 9 x86, x86_64
  - RHEL AS 4 x86
  - Ubuntu 6 x86, x86_64
- Results reported back to project

# Build-Test Infrastructure



# Porting Status

- Currently producing snapshots for :
  - Windows XP x86
  - Linux x86
  - Linux x86_64
- Work is in progress for
  - Itanium Processor Architecture
  - PPC Architecture
  - OS X operating system
  - FreeBSD operating system

# Basic Roadmap

- Dec 2006 : 97% of Java SE 5 class library coverage
- Q1 2007 : 100% Java SE 5 class library coverage
- Q1 2007 : DRLVM is stable with reasonable performance
- Q1 2007 : begin testing with TCK
- Q1 2007 : work begins on Java SE 6 features
- Feb 2007 : Jave SE 5 JDK toolset complete
- Q2 2007 : TCK certification of Harmony completed
- Q2 2007 : Apache Harmony SE 5 JDK and JRE released
- Q4 2007 : Apache Harmony SE 6 JDK and JRE released
- Q? 2008 : Apache Harmony SE 7 JDK and JRE released.

# Relevance in the Age of OpenJDK

Caution : The following section contains political and other unfounded statements which may not be suitable for all viewers. It does not represent the position o my employer, the Apache Software Foundation, the Apache Harmony project or my friends or enemies. Sometimes I can't even believe I say this stuff. Your mileage may vary. Viewer discretion is advised. Offer not valid in CT, TX, CA or the Canary Islands. Supplies limited. We reserve the right to substitute an item of similar value and quality...

# Just to be clear....

- I applaud Sun for doing OpenJDK
  - Courageous as predictably internally disruptive
- It's a monumental event in the history of open source
  - I can't think of any contribution to FLOSS of this magnitude
- The adventure is just beginning for Sun and the greater community
- I look forward to find ways to have OpenJDK and Apache Harmony collaborate (hint : javac)

# Some History

- What significant changes happened to the Java ecosystem when Sun released Project Glassfish?

# Some History

- What significant changes happened to the Java ecosystem when Sun released Project Glassfish?

(nothing)

# The Java Ecosystem

Two critical facets of the Java Ecosystem :

- Specification-based technology

  *“Collaborate on specifications, compete in implementations” - me*

- Historically consisted mostly of proprietary software
  - Started that way by Sun
  - We’ve been slowly chipping away

# Moving Forward

- Harmony community members choose to participate for their own, independent reasons
  - “It’s fun” -dt
- Need for FLOSS implementation that permits freedom of downstream licensing persists
  - *Sun recognizes this - they will continue to license OpenJDK under proprietary, commercial terms*

# Moving Forward

- Competition/choice makes Java healthy
- Community matters
  - Apache has well understood, transparent model
    - Collaboration of peers - this doesn't work for everyone
  - OpenJDK community governance still to be defined

# Summary

# Accomplishments

In 21 months.... :

- ~96% of Java 5 class library API coverage
- Modern, Java 5-capable VM with JIT
- Community is growing
- Full Apache project
- Self-hosting codebase
- “Runs” major applications
- Modularity approach successful in classlib, ongoing in VM

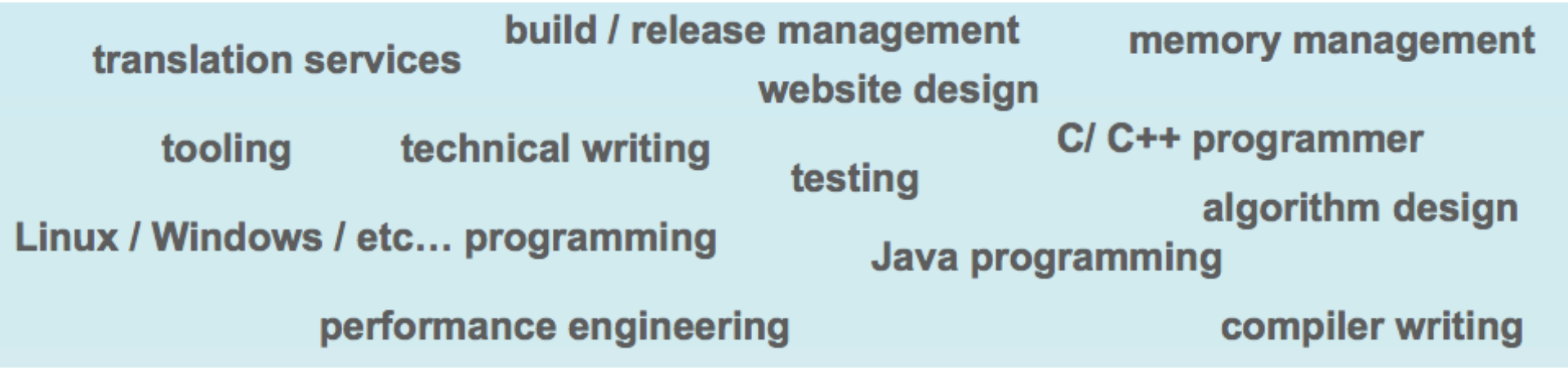
# Active Areas of Effort

- Class library completion
- VM performance work
- VM stability work
- Built/Test infrastructure
- Improved testcase coverage
- Real-world application testing
- Community growth

# Summary

- Goal is a compatible, open-source, impl of Java SE
- We invested time up-front getting the IP infrastructure right.
- The community is focused on building a first-class, modular runtime environment.
- We have come a long way in 21 months!

# Come Join Us




*Come and contribute in your area of interest!*

http://harmony.apache.org/

# Q&A