OpenJDK

Mark Reinhold
Java SE Chief Engineer
mr@sun.com

2007/2/25

Mark Reinhold	Overview
Igor Nekrestyanov	2D graphics
Peter Kessler	HotSpot
Peter von der Ahé	Compiler
David Herron	Quality

Why?


Ubiquity

debian

redhat.

ubuntu

linux for human beings


ORDEM E PROGRESSO

Sun’s big goals

- A vibrant OpenJDK community
  - Transparent governance
  - Non-Sun committers
- A fully free OpenJDK
  - Conformant (and usable!) free replacements for encumbered modules
- An OpenJDK-derived distribution in Debian Main and Fedora Core

Past & future
Facts of life
Code, tools, & processes
Governance

Past & future

Early launch: November 2006

- GPL v2 + “Classpath” exception
- JDK 7 HotSpot VM & Java compiler
- Source bundles & read-only Subversion
  - Updated at each promoted build
- Current java.net infrastructure
  - Static web pages, e-mail lists

Past & future

Full launch: 1HCY2007

- Fully-buildable JDK 7
  - With “binary plugs” for encumbered modules
  - Interim measure until free replacements can be developed
- Source bundles & read-only Subversion
- Current java.net infrastructure
  - Sorry.

Past & future

Why is the full launch taking so long?

Source preparation ...

- Source & build architecture
- Carving out encumbered code
- Unit/regression-test audit
- Binary audit
- Trademark audit
- General cleanup

Past & future

Following full launch, over time

- Improved java.net infrastructure!
  - Dynamic content, wiki, source browser, ...
- Public Mercurial repository
  - With support for external committers
- Processes & tools fully externalized
- Automated build-&-test service

We’re probably going to get some stuff wrong

Please tell us how we can do better!

Past & future
Facts of life
Code, tools, & processes
Governance

The Java Community Process

- Governs the standard Java specifications
  - Established, well-defined, open process
- Open source is about implementations, not specifications
  - We didn’t open-source the JCP
  - We didn’t open-source the specifications

The Java Community Process

- Java Specification Requests (JSRs)
  - Fundamental unit of specification
- JSR deliverables
  - Specification
    - What does it do?
  - Reference implementation
    - Can it be built?
  - Conformance test suite
    - Is an implementation complete & correct?
    - For Java SE, this is the JCK

The Java Community Process

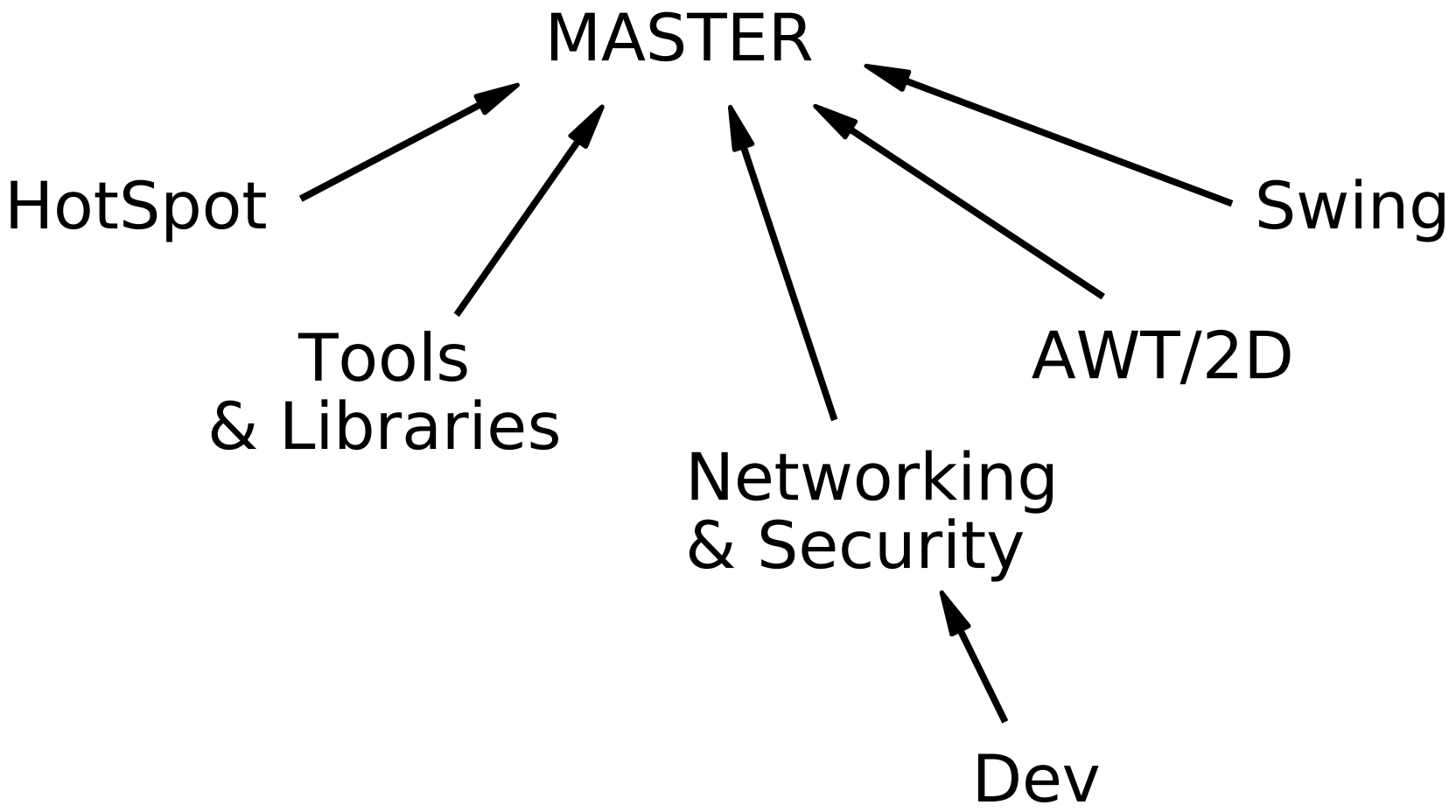
- OpenJDK is (just) one implementation of the Java SE platform
  - Must conform to all JCP requirements in order to be called “Java compatible”
  - In particular, must pass the JCK
- Experimentation and exploration is great!
  - New APIs, language features, bytecodes, …
  - You can call it “based on code from OpenJDK”
  - Just don’t call it “Java compatible”

Quality, stability, & compatibility

- Quality
  - Takes priority over schedule
  - Millions of people depend upon the JDK
    - We must take this seriously
- Stability
  - Five nines (today), seven nines (tomorrow)
- Compatibility
  - If existing code runs on the current release then it must run on the next release
  - No matter how stupid the code might be
  - Except for security issues

Past & future
Facts of life
Code, tools, & processes
Governance

Code flow


MASTER
HotSpot
Tools
& Libraries
Networking
& Security
AWT/2D
Swing
Dev

Source-code management

Present: TeamWare

- Fully-distributed SCM
  - No need for a scalable, centralized server
    - Or a connection to one!
  - Experiments are cheap
- Supports complex code flows
- But ...
  - Sun-proprietary
    - Yeah, we could open-source it, but ...
  - Does not scale well (relies upon NFS)
  - Old and creaky in other ways

Source-code management

Future: Mercurial

- Most mature and performant of the modern distributed SCMs
  - Others: Arch, Bzr, Darcs, Git, Monotone
- Scales well across the net
  - HTTP & SSH, not NFS

http://selenic.com/mercurial

Types of changes

Implementation change

- No API or specification changes
- Bug fixes, performance work, *etc.*

1. Write code and test
2. Get code review (webrev tool)
3. If in release endgame: Get approval
4. Integrate change

Types of changes

Specification change

- New API, changed API, or spec clarification
- Might not include code change

1. Write code and test
2. Get code review (webrev tool)
3. CCC review (online tool)
4. If in release endgame: Get approval
5. Integrate change

Types of changes

New feature

- Might not include specification change
- Might be a whole JSR

1. Submit in jplan (online tool), get approval
   After convincing appropriate JSR EG if needed
2. Write code and test
3. Get code review (webrev tool)
4. CCC review (online tool)
5. Integrate feature (before endgame!)

Bug tracking

- Currently an internal system
  - Cannot be open-sourced
  - Cannot be externalized as-is
- Looking at open-source alternatives
- Complex problem!
  - Tracking bugs across different projects
    - Not just at Sun, but upstream (*e.g.*, GNOME)
  - Sun's internal needs (gotta pay the bills)
    - Service calls, customer escalations, *etc.*

Any suggestions?

Governance

- Where we are today
  - Sun in control
  - External contributions submitted as patches
    - Via e-mail
    - Handled by a Sun sponsor
- Where we want to go
  - OpenJDK community in control
  - External contributions integrated directly by committers

How do we get there?

Governance

Possible structures

- Benevolent dictator (Linux)
- Highly structured republic (Apache, OpenSolaris)
- Loosely structured republic (GNOME)
- Something else?

Thoughts?

Governance

Some sort of republic seems most likely

- Will need a governing board
  - With significant non-Sun representation
  - Ultimate point of dispute resolution
  - Hopefully rarely needed!

Q&A

OpenJDK

Mark Reinhold
mr@sun.com