Life In The Trenches
The agony and the ecstasy

Andrew Haley
Red Hat Open Source Java Team

January 2014

This is about you

- This talk is about what it's really like to work on OpenJDK, the good and the bad
- Please interrupt me as much as possible or I won't have much to talk about!

This is not a moan-fest

- Or, at least, I don't want it to be

This is not a moan-fest

- Or, at least, I don't want it to be
- OpenJDK is an amazing piece of software to work on: state-of-the art performance, several garbage collectors, two just-in-time compilers
- Enough good stuff to keep a CompSci PhD busy for years
- And it's all free software. Rejoice!

This is not a moan-fest

- OpenJDK is one of the largest ever donations of free software
- This is a very big deal, and I am grateful

This is not a moan-fest

- OpenJDK is one of the largest ever donations of free software
- This is a very big deal, and I am grateful.
- I don't know what I'd be doing now if it were not for OpenJDK, but I'm pretty sure it would not be so much fun

The interviews

- I signed up to do this talk without thinking it through
- I realized that, on my own, I didn't really have enough to talk about
- And also, maybe my experience isn't representative: I am a JCP representative, I'm on the OpenJDK Governing Board, and I'm the lead of Red Hat's Java team. That's not typical!

The interviews

- So what do you do if you don't have enough ideas? Talk to people and hope to find inspiration!
- I interviewed several OpenJDK developers, all members of the community outside Oracle
- That's not because the Oracle developers aren't community members, just that I wanted the experiences from “outside”
- These were all full-time developers. That might have been a mistake, but I don't know many people working on OpenJDK in their spare time (Shame)

The questions

- I wanted to ask a set of questions to everyone
- It's interesting to ask everyone the *exact same* questions: that way you can find out how people's opinions (and experiences) differ

The questions

- How long have you been an OpenJDK contributor?
  - The answer varied from 2-7 years
- What projects and groups have you worked with?
  - HotSpot, class libraries, build, Java 2D, Caciocavallo, AWT

The questions

- Do you think the rules for patch submission and joining a group are clear?
  - Opinions varied widely, all the way from “Yes, reasonably” to “No!”
  - I sometimes wonder if I am the only one who doesn't understand this stuff, but evidently I'm not

The questions

- How would you describe the process for becoming an OpenJDK contributor?
  - This was a badly-phrased question (*mea culpa*)
  - I got “It's not bad” and “Reasonable” and “The process is really convoluted”

The questions

- Would you encourage a friend to to contribute to OpenJDK?
  - Interviewees generally were very positive about this. “It's a cool project!” “It's important” “There's something for everyone”
  - However, they also noted that you have to be pretty serious because “It is really hard to become a contributor”
  - But on the positive side, “People on the lists are helpful”

The questions

- How transparent are the groups you work with? Can you tell what they're doing, and what their priorities are?

This one was mixed.

  - “By now we know what they do, but we can't tell what their priorities are.”
  - “Groups are reasonably transparent, but only after following them for a long time. You have to follow all the lists.”
  - “It depends on the group. Sometimes it's easy, but some groups are not nice. Some people really go the extra mile.”
  - “We need to know more about the roadmap.”

The questions

- Describe the process of finding a bug (or enhancement) through to a patch submission
  - Interviewees observed that this varies widely: it can take just one day, but other patches languish for months
  - More than one interviewee noted that important patches had been abandoned after a long time
  - Mailing lists seem to be a real pain point. Even fairly experienced contributors can struggle to find the right lists

The questions

- Should OpenJDK do things differently? If so, in what way?

  This is a very open-ended question, and was more to prompt discussion than get simple answers, but:

  - “The fixed rules about writing X patches before getting commit access to a project don't make sense.”
  - “We really need to make more people reviewers.”
  - “We need to sort out the HotSpot infrastructure. It makes no sense that external reviewers cannot push.”

And finally...

- Is OpenJDK a well-run Open Source project?
  - “It's more or less well-run, but not open source”
  - “The developer community is good, helpful, and well-run”
  - “Generally positive”
  - “Generally yes, but it's hard for others to pitch in”
  - “It's generally well-run, but it's more 'they let us contribute' than 'we're part of OpenJDK.'”

Any comments at this stage?

- Do you agree with the interviewees?

Matters arising

- Patch delays are a major pain point
  - One senior Oracle manager has pointed out that patch approval delays can be just as problematic inside Oracle
- So we in the wider community should remember that it's not just us who suffer delays
- The open-source model benefits everyone

Matters arising

- Even though we now (almost) have two community-contributed back ends in HotSpot and a fair bit of expertise, we still have to find a sponsor to push changes. This really won't do
- But working on the class library is much easier, and now relatively pain-free.

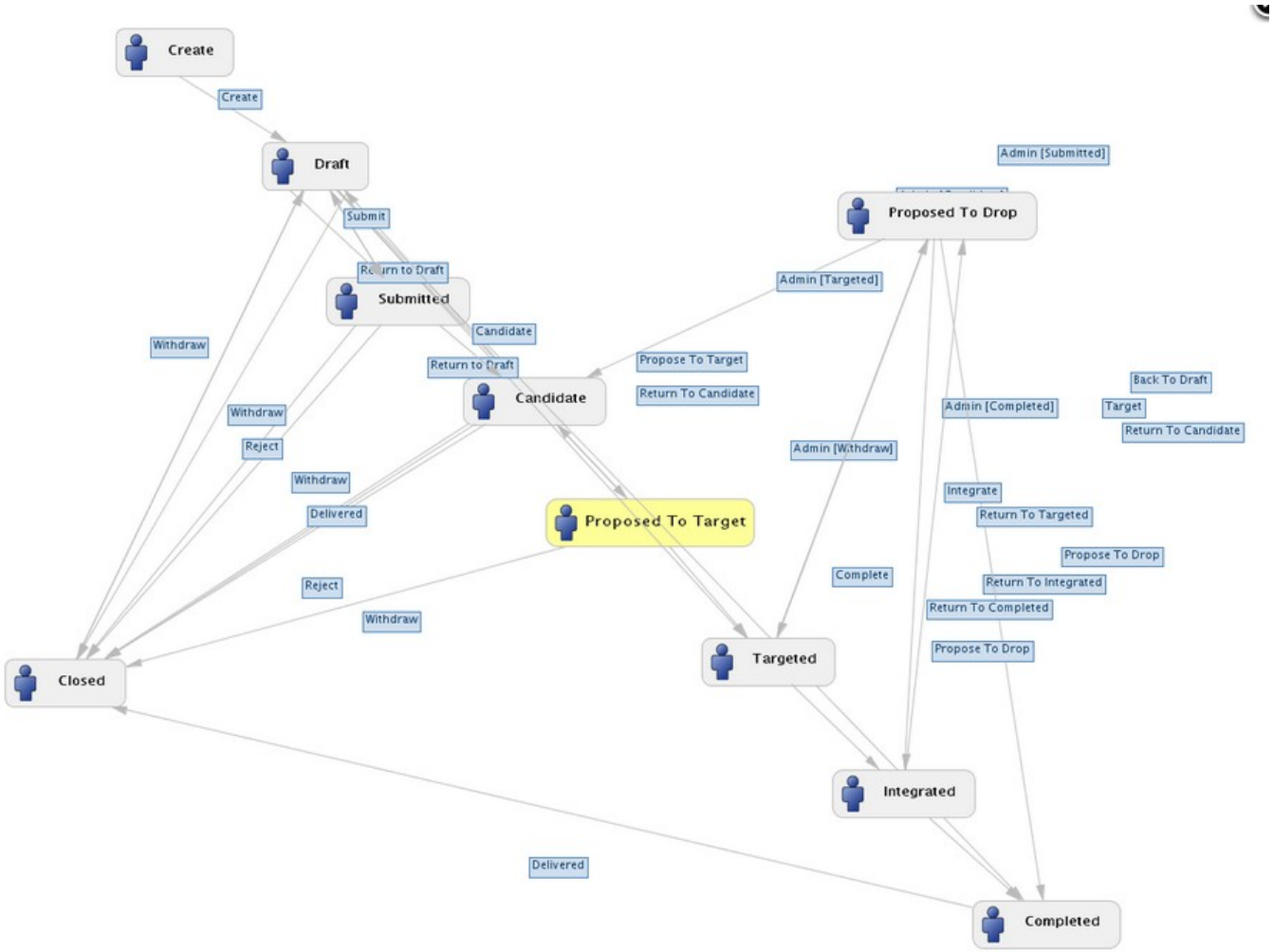
Matters arising

- One interviewee who has worked on the proprietary JDK (outside Sun/Oracle) observed that the OpenJDK development process is now much *better* than before:
  - “If you have a good argument it's much easier to work directly with the open-source developers than as an official licensee”
- It's easy to assume that things are different behind the proprietary wall, but it's not necessarily so!

Matters arising: Java Enhancement Process

- The JEP came up several times
  - JEP 2.0 is now pretty good
  - You create a JIRA database entry and fill in the sections
  - The process is complex, though:

Matters arising: JEP



Matters arising: Java Enhancement Process

- In general, though, JEP 2.0 is a great improvement. The real problem is that it can be hard to find someone already inside to review and endorse your proposal
- The long-term solution to that must be more people in the wider community who can review proposals
- For that to happen we have to keep writing good proposals and contributing good code

Matters arising: Java Community Process

- The JCP came up several times too
  - The JCP is the standards-setting mechanism
  - The relationship between the JCP and OpenJDK is somewhat mysterious
  - In some cases, though, it works well
    - JSR 335, *Lambda Expressions for the Java™ Programming Language,* is a standout example of a JSR developed using the OpenJDK process
    - 335 had open discussions with many participants. This is a great way to get people involved in a positive way without necessarily contributing any code

Matters arising: Java Community Process

- Sometimes, though, important decisions are made and discussed in the expert group's mailing list. That is not so good, but it is more, er, traditional

Matters arising: Bylaws

- The rules about how to work on OpenJDK aren't clear to people
- That doesn't mean that the rules themselves are ambiguous, but there's a heck of a lot of them!
- There is a web page with a description of how to become an author which seems to be different from the bylaws
- Adopt OpenJDK seems to have helped with hand-holding new contributors

Matters arising: onboarding

- The key issue, as with many open-source projects, is how we can grow the community with more contributors
- Apart from the procedural issues we have already discussed, there's the small matter of the sheer scale of the project

```
$ find . -name .hg -prune -o -type f -print | xargs cat | wc
10670268 41980596 421541396
```

  - Yes, that's right. 10 million+ lines of code. And it ain't easy code

Matters arising: onboarding

- The complexity and the need for deep technical knowledge seems to be most acute with HotSpot
- I don't think this is anything particular to HotSpot or OpenJDK: I've been through this before with GCC, which also took me a couple of years to get used to
- Industrial-strength software is hard. Textbook algorithms look very different when used in anger

Matters arising: onboarding

- We have jtreg regression failures in the test suite
- We could do a lot worse than getting new volunteers to work on some of them. They aren't all difficult
- There are also bugs in JIRA that have been languishing for a long time. I picked one of them up last week and was all done in a couple of hours

Matters arising: security updates

- Every six months (and occasionally more often) we get a drop of critical patches from Oracle
- We apply them to our local packages, test them, and on the embargo date push them out
- We back-port them to OpenJDK 6
- We usually have about ten working days to get all this done
- This is by far the most painful part of working on OpenJDK.
- Oracle have a new proposal for a Vulnerability Group. Yippee!

Matters arising: finding a buddy

- Programming in the large is a social activity
- Sometimes carried out by not-very-social people!
- We really need a way for contributors to find friends who already work in the areas they're interested in

And finally

- As I said at the start, this was never intended to be a moan-fest
- OpenJDK, as a project, is getting better all the time
- We have some way to go!

Discuss...

- Does anyone have anything to say?
- If not, I need a coffee...