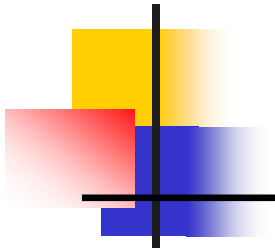
Applying Lessons from Open Source Development

Noel J. Bergman

DevTech®



Session Overview

The Apache Software Foundation has thousands of contributors in all time zones and most countries, collaborating to deliver world-class software. What lessons can you learn from such Open Source communities to apply within your company?

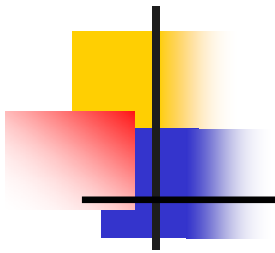
We will talk about decision making, communication, source control, community dynamics, and other aspects that enable communities of developers scattered across the globe to efficiently and effectively build software in today's global environment.

This session is intended to be interactive – we are to engage in a discussion, not a monologue – allowing us to discuss how the various processes and philosophies can be introduced into your own development organizations.

PLEASE ASK QUESTIONS! ☺

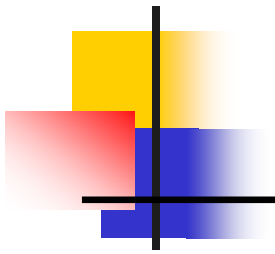
Project Structure/Governance Models

- Why is the model important?
  - Who participates?
  - How do you communicate?
  - How are decisions made?
  - How is oversight maintained?
- Typical Corporate Models
  - How are things structured and governed in your environment?
  - What happens when things don't go right?
    - Tester: "I submitted a bug to [the developer], and he hasn't fixed it in a year. Now we're supposed to roll out!"
- The "Despot" Model
- The Community Model
- Do all Open Source Projects *have* a model?



The ASF Approach

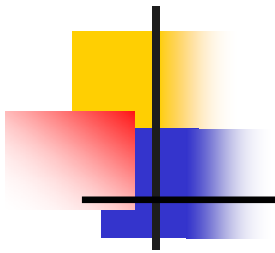
- The Apache Software Foundation (The ASF) has an approach to projects.
  - Emphasis on Community
  - Collaborative
  - Meritocratic
  - Consensus-based Decision Making
- This is the model we'll spend most of our time discussing.



Emphasize Community

"Great communities lead to great code. Great code does not necessarily lead to great communities."

- Collective ownership
- Collective decision making
- Collective development
- Collective maintenance

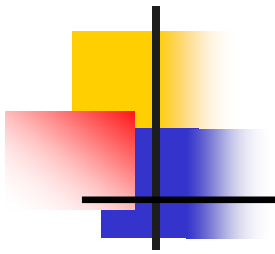


“Collective”?!

- “What is this, a software commune?”
- “This is the USA, not the USSR!”
- “We’re Capitalists, not Marxists!”
- “Hey, I want to be proud of **my** code!”
- …

Why Community Is Important

- People move on.
- Projects need to outlive individual involvement.
- Volunteers want to feel included in the process.
- Corporate participants want at least an equal say with their peers.
- Everyone wants control, and no one wants to give it up.
- We want a level playing field.
- Users want a voice, too. And developers need to hear them.
- We want to make the best decisions.
- Sad to say, but there are legal reasons, too.

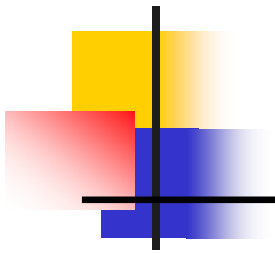


Community Members

- A community isn't a clique.
  - Your developer team isn't the community.
- The community consists of all the stakeholders:
  - Developers (of all sorts: Java, Web UI, *etc*.)
  - Testers
  - Administrators
  - Users
  - ...
- A community is connected and inclusive.

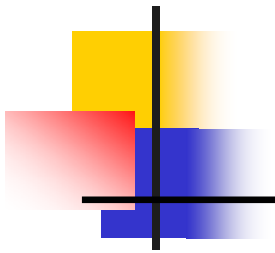
An Example

- Think about the possible Community involved in a single Portal project
  - Portlet Developers
  - UI Developers (possibly dealing with multiple markups)
  - Portal Administrators
  - Application Server Administrators
  - Application (model) Developers
  - Network Administrators
  - Database Administrators
  - LDAP Administrator
  - Security Administrators
  - Accessibility Experts
  - Language Translators
  - Documentation Writers
  - Business Analysts
  - Line of Business Managers
  - Managers for each of the above groups
  - The Powers That Be (holders of the purse strings)
  - ...
  - Oh, and those pesky creatures known as **USERS**!
- Does your model permit them to participate?
  - When?
  - How?
  - Timely?



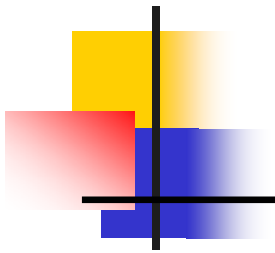
Collaborative Decision Making

- Meritocracy
- Meritocratic Egalitarianism
- Vote *vs*. Veto
- Policy *vs*. Technology
- Justification? Why?



Communication

- Mailing Lists and Archives
  - Asynchronous
  - Time zone-agnostic
  - Involves the community
  - Self-documenting
  - Preserves and communicates history
  - Allows non-native language participants the opportunity to review and comment at their own pace
- Newsgroups can also be used, so long as the server preserves a long-enough lifespan.
- What about IRC, Instant Messaging, *etc*?



Source Control

- Most of us already have decent source control.
- A few items, though, are crucial
  - Commit notices
  - The ability to re-factor code without losing history

Commit Notices and Code Review

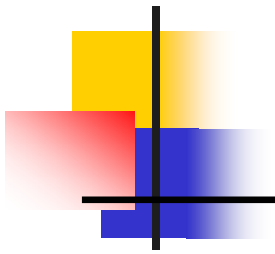
- Commit notices are absolutely vital to the process.
- Commit notices, including a diff, are automatically posted to a mailing list.
  - Generally either ${project}-commits or ${project}-dev
- The community is responsible for reviewing the changes.
  - Everyone can see what everyone else is doing.
    - Oversight and cross-training
  - Everyone can decide whether or not to keep the change(s).
    - Collective decision making. Who is responsible? **We** are.

Continuous Integration

- There are Open Source and proprietary tools that offer continuous building and testing of multiple projects.
- The ASF has two systems constantly building projects:
  - Apache GUMP – builds all projects from scratch in the order necessary to resolve dependencies.
    - Provides early warning of changes in upstream components that will cause problems later.
  - Apache Continuum – builds projects from their own latest source code based upon declared dependencies (specific versions).
- Some projects have their own systems to perform nightly builds with unit and integration tests.
- All such systems should mail notices to the community.

Unit Testing

- As with continuous integration, the key here is not doing unit testing, which is something that we should be doing anyway, but rather ensuring that:
  - these tests are done often, as in a nightly build environment
  - the tests results are posted to the community, keeping everyone in the loop regarding the current status



Avoid Silos

- Lots of great projects, but no one knows what anyone else is doing.
  - Lots of redundant effort and code
  - Inconsistent approaches
  - Incompatible solutions that should interoperate
  - Reinventing the wheel
  - Orphaned projects
- Silos are budget busters!
- One solution: DOAP

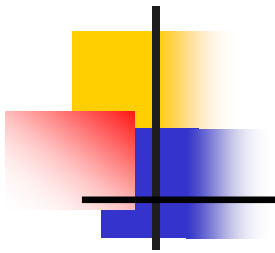
PROJECTS.APACHE.ORG

- Initiated by David Reid based upon the public DOAP specification
- Each project maintains its own DOAP file(s) in its own source control, and records the location(s) in a central file.
- Code regenerates http://projects.apache.org from the individual files.
  - Ontological indices are generated based upon:
    - Programming language
    - Standards (IETF, W3C, OASIS, JCP, vendor, *etc.*)
      - ✓ Implemented
      - ✓ Consumed
    - Project Management Committee
- Allows everyone to see who is working on/with what specifications.
- Not yet fully populated, but coming along.
- The same solution is now being investigated by major corporations to solve the Silo Problem within their organizations.

Why Open Source?

- Why you might want to Open Source your next project, like:
  - Apple, *BEA*, *Google*, *IBM*, *Intalio*, *Iona*, *JP Morgan*, Novell, *Red Hat*, *Sun*, *Sybase*, ... †
- "Open Source Communities As An Exercise In Enlightened Self-Interest"
  - Why should businesses want to participate in collaborative, Open Source, communities?
    - The effect of Open Source on the dynamics of Software Darwinism.
    - Does this affect corporate projects, too?
- Examples
  - Apache Derby: A Poster Child
  - JSR-168

† *Those in italics have contributed major projects to The ASF.*



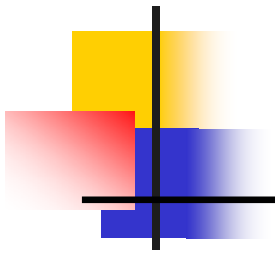
Some Interesting Statistics

- Developers world-wide contributing to open source: 83% (IDC 2006)
- Average experience of open source developer: 11 years
- US businesses are using open source: 87% (Optaros 2006)
- Organizations world-wide using open source software in production: 54% (IDC 2006)
- Average amount saved by US companies with > $1bn in revenue by using open source in 2004: $3.3m
- Saved by US companies with <$50m: $550k (Optaros 2006)

Statistics courtesy of Rebecca Hansen, Open Source Strategy Manager, Sun Microsystems

License Matters

- If you are going to participate in Open Source, you need to know about licensing.
- See Donald Smith's session "*Open Source Licensing for Developers".*
- Quick summary:
  - GPL – Requires all software that touches software licensed under the GPL to also be put under the GPL. Exists primarily to prevent closed source development based upon Open Source code.
    - Linus declared the Application Binary Interface to be a licensing firewall, removing the viral consequences for programs in the user space.
    - We have seen the GPL (arguably) abused by some companies, who require all contributions to be signed over to them, and then offer dual licenses: GPL for free use, and then as the sole provider of proprietary, and fee-based, variations under a non-free license.
  - Apache/BSD – Maximum freedom for developers to do whatever they want with the code, including making closed source versions. Exists primarily to prevent brand abuse and litigation against developers.
    - The ASF does **not** require that copyright be assigned. Rather, the ASF receives a grant from contributors, who maintain copyright ownership, to license the contribution under the Apache Software License.
  - Various other licenses require that changes to the Open Source code be contributed back to the owner, but are non-viral with respect to code that uses the open source project.
- When evaluating participation in an Open Source project determine:
  - Who owns the copyright to the (your) contributions?
  - What obligations/rights do you have with respect to the Open Source code?
  - What obligations/rights do you maintain with respect to your own code?



Related Sessions

- "Join the Conversation — Being a Good Open Source Software Citizen and Fostering Communities" – Donald Smith
- "Open Source Licensing for Developers" – Donald Smith
- "Managing Open Source" – David Moskowitz