The Anatomy of A FOSS Project

We heard the historical/political side

- What does it mean then to say that something is a FOSS project?

- What do we think of when we hear that term FOSS project?

- What are expectations with regard to project organization and working style?

- What is the infrastructure of a FOSS project?
FOSS project
“requirements” (?)

• License
• Inclusive philosophy
• Open communication
• Meritocratic model
• Bottom-up architecture
• Distributed work model

How to support a distributed work model?

• Communication tools
  – IRC
  – Mailing lists
  – Web/wiki pages
• Coordination tools
  – Bug trackers
• Coding tools
  – CMS (git, hg, SVN, CVS)
• Dissemination tools
  – Web
How to support a distributed work model?

• Communication tools
  – IRC
  – Mailing lists
  – Web/wiki pages
• Coordination tools
  – Bug trackers
• Coding tools
  – CMS (git, hg, SVN, CVS)
• Dissemination tools
  – Web

• Missing anything?

---

Communication tools

IRC (Internet Relay Chat)

– Lowest common denominator for real-time communication
– Divided into #channels
– Non-persistent channel with limited flow control
– Used by developers to chat, coordinate and get answers to quick/simple questions
– Mix of social and technical content
– Extremely noisy in many cases
– Get a good IRC client
IRC (cont.)

- Huge difference between raw text communication and formatted text

- Anyone with a ~&@%+ is a special member/moderator

- Spartan and often direct style of communication

- Can sometimes be archived, but often frowned upon

- Often intimidating, sometimes on purpose

Communication tools

Mailing Lists

- Simple redistribution and archiving mechanism

- Strict rules about attachments and formatting to facilitate automated processing

- Used by developers and users to coordinate and
  - Get answers to more involved questions
  - Hash out designs/requirements/direction
  - Submit/vet patches

- Etiquette really important
  - Follow formatting guidelines
  - Figure out who to ask
  - Check to see if it has been discussed already

- Set up filter rules
Mailing Lists

• Almost always archived (public)
• Side conversations (off list) often frowned upon
• Some lists protect posters’ privacy (anonymize email addresses)

• Start by ‘lurking’
• Learn the difference between regulars and newbies
• ID the important people
• Learn how to talk the talk

Mailing Lists

• The role of mailing lists: Official vs. unofficial project communication
  – A lot of announcements
  – A lot of FYI info, not necessarily open for discussion
  – Anything ever posted to the list should be taken as read and understood
• As a newbie, avoid project leaders if possible
  – Most overworked project members
  – Most likely to have seen your question too many times
  – Statistically most likely to flame you
  – Some of them **are** very nice
Communication tools

Web/Wiki pages
- Projects’ public face
- Required info
  - Project status info
  - Key membership info
  - IRC channel info
  - Mailing list info
    - Link to archives
  - Design docs (?)
  - Code repo
  - Bug tracker
  - Code download
  - Executable download (?)
  - Help wanted ad (?)
  - FAQ (?)

What do you need to know?

What did you want to know about the project?

1. Purpose & goals
2. Design decisions
3. Community Org.
4. Contribution ideas
5. How to get started
7. Software dependencies
8. Activity level
9. Development stage
10. Working practices
Problems?

• Where and how are goals documented?

• Where and how is design documented?

• Where and how is “community” documented?
Bug Trackers

• What are they?
  – Essentially a database front-end for tracking:
    • Outstanding issues
    • What is known about them (what causes bug)
    • Who is working on them
    • Current status
    • Current priority
  – Essential for ensuring things don’t fall through the cracks in a distributed project

• More correct term could be “Issue Trackers”
  – Many projects mix bug reporting with feature requests
  – Not a problem per se., but can cloud statistics

Open Bug Reporting Key to FOSS

“Given Enough eyeballs, all bugs are shallow” – Linus’ Law

• Lacking dedicated QA teams and market researchers, FOSS projects depend on large numbers of bug contributors and end-users to help identify bugs and drive development direction
• Open bug reporting means anyone can contribute a bug report
• Requires an easy-to-use system
Bug Lifecycle

Can be complex, lengthy, requires tracking

Bug Trackers

Examples
• Bugzilla
• Trac
• Redmine
• Jira
• IBM Rational team concert
• MS Team Foundation Server

Anyone work with any of these?
• Likes?
• Dislikes?
Problems - Usability

• Contributing a bug report – is it easy enough?

Problems - Usability

• Is using reports easy enough?
Problems - Usability

• Duplicate bug reports – Jen Davidson
  – Up to 26% of all bug reports duplicate
  – Duplicates can take months to detect
  – Not always seen as bad

• Most people only ever report 1 bug

Other tools?
Documenting Design: Diagrams

- Diagramming difficult, but common in FOSS
  - Poor tool support: Difficult to share, track changes, merge
  - Many examples of workarounds: ASCII art, scanned images

- Practices not always what we expect them to be
  - Diagrams considered “Static” part of conversation, not updated
  - Great appreciation for auto-generated diagrams
  - Diagrams more commonly used to explain than focus of discussion

Documenting community

Can visualization tools help us understand how code and community “work”?
Resistance

- FOSS communities often adopt a minimalist philosophy
  - Can’t/won’t require use of non-FOSS tools
  - Can’t/won’t require use of non-standard tools

- Automatic documentation tools not always accurate
  - Experts often hung up on details
  - Newbies often happy with an approximate picture