Open Source Development

Dr. James A. Bednar

jbednar@inf.ed.ac.uk

http://homepages.inf.ed.ac.uk/jbednar

Dr. Chris Walton

cdw@inf.ed.ac.uk

http://homepages.inf.ed.ac.uk/cdw

Traditional Commercial Software Development

Producing consumer-oriented software is often done in much the same way as for tangible mass-produced products, such as furniture or cars. E.g., a company:

- Designs the product
- Produces copies of it
- Provides a copy to any consumer willing to pay for it

Most products from e.g. Microsoft fit into this model.

Traditional Assumptions

Implicit assumptions applying to tangible goods like furniture:

1. Making each copy costs money, and so it makes sense for customers to pay for each copy
2. Customers and other companies cannot easily make their own copies, so they have an incentive to purchase from the company
3. Development costs can most efficiently be recouped (and profits made) by charging per-copy fees
4. It's reasonable to expect improvements to the product to come only from the company

Intangible Goods

Software and other intangible goods like music and text can be duplicated for essentially zero cost, by consumers and companies alike, and so any restrictions on copying must be enforced artificially.

Software costs are for development and for support, independent of the number of copies produced, and so are not well-matched to per-copy revenue models.

Unlike tangible goods, software can be modified by users to meet their needs better and then redistributed, without needing the original company to produce the result.

Open Source/Free Software

The Open Source Software (OSS) and Free Software movements are focused on how software can be distributed in a way more suited to intangible goods, with:

- Free redistribution: both free of cost (free beer) and free of restriction (free speech)
- Freely available source code: to allow fixes and changes
- Freedom to redistribute modified versions

Open Source vs. Free Software

Roughly, the difference between the two movements is:

- Free Software is based on a firm principle that all software should be free (see gnu.org)
- Open Source focuses on the practical, business-friendly advantages to having freely distributable and modifiable code (see opensource.org)

Projects in either category can be understood as examples of highly distributed large-scale software development methodologies.

Advantages of OSS: For Users

- No restrictions on how software can be used: copying to new machines, sharing with co-workers, porting to new platforms, etc.
- Can trust the software to do what it says, not to have spyware, etc. – because source code is visible
- Can count on their data created by the program being readable in the long, long term, on future platforms, although they themselves may have to maintain it
- Because development is driven by users, features are likely to match what users actually want

Advantages of OSS: For developers

- No need to worry about piracy
- Users act as legions of beta-testers:
  Linus' Law = given enough eyeballs, all bugs are shallow
- Security holes can be found quickly, tested rigorously
- Huge number of users working in parallel can help make development faster (but can also be overwhelming)
- Companies can pool resources on a common good (e.g. Linux, Java, GCC)

Disadvantages of OSS

- Difficult to make money from OSS
  (but can sell support, customization, services)
- Less interesting tasks tend to get less work,
  e.g. documentation, newbie usability
- Hard for upper management to control
  (but is that necessarily a problem?)
- Difficult to coordinate corporate and user efforts
- Users don't have anyone to hold accountable
  (but in reality they can never do that anyway)

Cathedral vs. Bazaar

Major SW development methodologies (Raymond 2001):

Cathedral: Master plan controlled by one person, all implementers follow it. Suited to small projects, large projects run by one organization, and/or projects with only one customer.

Bazaar: No master plan. Many users contribute ideas, changes, features. Often the principal maintainers' role is just to make sense of it all.

Successful OSS projects of both types exist, but bazaar is particularly suited to OSS.

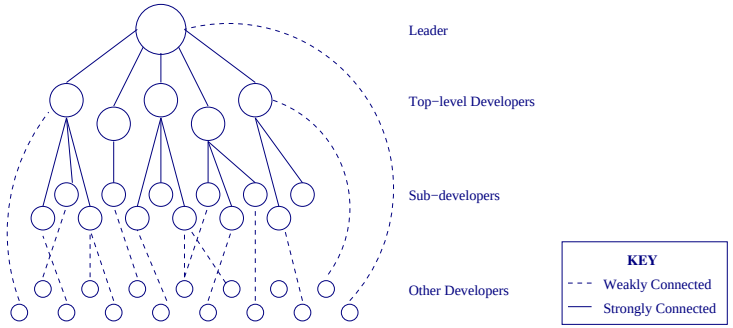
OSS Success Stories

- The Internet/WWW (Microsoft did not invent it!)
- Apache — the most popular web server in the world
- Linux and *BSD
- Sun Open Office – released to encourage purchase of hardware
- SourceForge (approaching 100,000 hosted OSS projects as of 1/2005)
- Mozilla/Firefox?
- GCC, Emacs, GNOME, KDE, AbiWord, Python

Linux

- Implementation of a UNIX-like OS kernel
- Main competition: MS Windows
- Final code is controlled completely by Linus Torvalds who operates a "benevolent dictatorship"
- No code revision tools are used - patches are submitted and controlled by Linus
- Two versions of the code-base are maintained at the same time: stable and development branches
- Developers synchronise entirely by e-mail, mailing lists

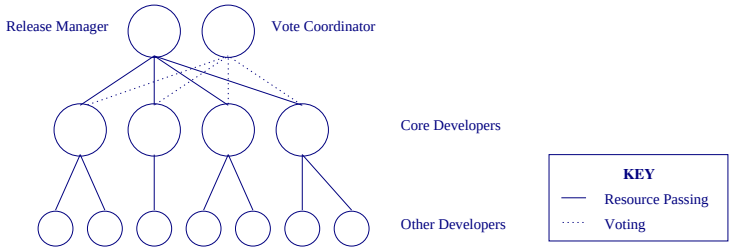
Linux Development Environment



Apache

- Extremely successful web server product (1/2004: 67%)
- Main competition: MS IIS (1/2004: 21% share)
- Development coordinated by team of core developers
- No central leader
- All major decisions voted upon by the core developers: 3+, 0- required
- Selection of core-developers is a form of meritocracy
- Code revision is controlled by CVS
- Bug tracking via GNATS database (not often used)
- Developers synchronise via e-mail, mailing lists, web and newsgroups

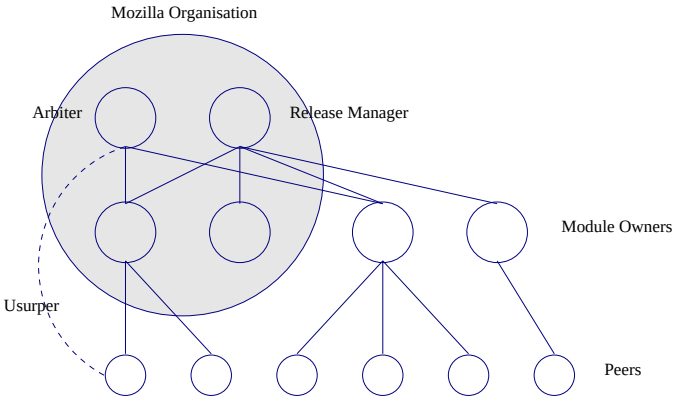
Apache Development Environment



Mozilla Firefox/Thunderbird

- OSS version of Netscape Navigator/Messenger
- Main competition: MS Internet Explorer/Outlook
- Difficult transition to OSS; years before stable release
- Didn't seem to help Netscape the company survive
- Development is co-ordinated by the Mozilla Organization (MO), (see www.mozilla.org/mission.html)
- MO operates a "benevolent dictatorship" like Linux
- Each module has an owner, designated by MO, but owner can be usurped
- Code revision controlled by CVS, bug-tracking by Bugzilla

Mozilla Development Environment



GCC

- GNU Compiler Collection (C/C++ and related compilers)
- Started in 1984, released yearly or so since 1987
- Part of Richard Stallman's GNU project
- Experimental branch forked in 1997 as EGCS, later usurping to become official maintainers
- Guided by GCC Steering Committee
- Many ports are funded or supplied by hardware industry
- Large test suite, nightly regression testing
- Nightly builds, snapshots available to all
- Code revision by CVS, bug-tracking by Bugzilla

Summary

- Software is not like tangible goods
- Difficult to devise appropriate revenue models
- Open Source projects are examples of highly distributed, parallel, user-driven development
- Both cathedral, bazaar styles can work

Required reading:

`http://www.catb.org/~esr/writings/cathedral-bazaar/`

Recommended: Mockus et al. 2002

References

Mockus, A., Fielding, R., & Herbsleb, J. D. (2002). Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology*, *11* (3), 309–346.

Raymond, E. S. (2001). *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly.