

Open Source Methodology

Course: Software Engineering Principles

Lecturer: Dr Petra Malik

FOSS Development

- Development artifacts are publicly available and development activity is openly visible
- Usually no formal project management regime, budget or schedule
- Oriented towards growth of a community of developers and users
- Shown to produce high quality and sustainable software

Open Source and Money

- “The Magic Cauldron” looks at how money is currently made in “traditional” system
 - Software has two values: use value and sale value
 - Are software companies driven by sale value?
 - Is sale value appropriate?
- Raymond claims a “manufacturing delusion”

The Factory Model

- Two premises:
 - Most developer time is paid by sale value
 - Sale value is proportional to development cost
- Raymond shows these premises to be false
 - Most developer time is spend on maintainance and in-house code
 - When vendor goes out of business, their software is worth almost nothing
- Service industry, not manufacturing industry

Open Source Models

- Emphasise/maximise use value by
 - Expanding user base by providing zero-cost products
 - Building a community and motivating members to participate
- But resources are required to build community
- How can companies gain economic value to cover these costs and make profit?

Loss-Leader/Market Positioner

- Indirect Sale-Value Model
- Use open source to create or maintain market position for proprietary software
 - Open-source client software to increase sales of server
- Example: Netscape (Mozilla)
 - Open source browser used to combat Microsoft's Internet Explorer and thus prevents Microsoft to control server market

Widget Frosting

- Indirect sale-value model for hardware manufacturers
- Hardware is profit area but software needs to be written and maintained for the hardware to be accessible
- Open source provides pool of debuggers and developers and future support
- Example: Darwin, the core of the Mac OS X

Give Away Recipe, Open Restaurant

- Use open-source software to create market position for services
- Example: Cygnus Solutions
 - configure script to unify build process for GNU tools
 - sold support services and binaries bundled with their version of GNU tools
- Other Examples: Red Hat and other Linux distributors

Other Models

- Free the Future, Sell the Present
 - Source with closed license and expiration date
 - Example: Aladdin Enterprises' Ghostscript
- Free the Software, Sell the Brand
 - Sell brand that certifies their implementation is compatible with others using the brand
- Free the Software, Sell the Content
 - Open source software but sell subscription to content
 - Example: Real Player

Example: Pentaho

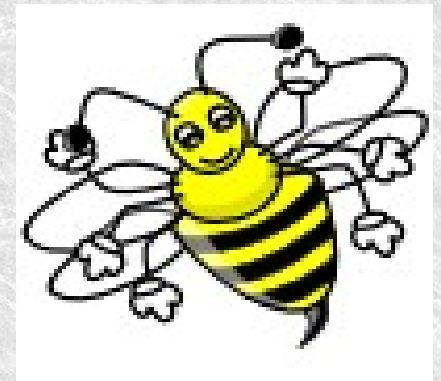
- Founded in 2004
- commercial open source alternative for business intelligence
- leads and supports open source project
- provides support, service, and product enhancements via an annual subscription
- The bee keeper
 - <http://wiki.pentaho.com/display/BEEKEEPER/>

The Bee Keeper Model

- Bee keeper (company) creates an environment that is attractive to the bees (community)
- Bees do what they do naturally and make honeycombs
- Bee keeper turns effort of bees into products that the customers desire (honey and wax)
- Money from selling products is used to grow his bee farm

The Bee Keeper Analogy cont.

- Each individual bee makes a small contribution
 - A large number of bees is required for success
- Bee keeper has little control over bees
 - Bees may desert (project forking)
 - Bees can sting
- Bee Keeper must grow both his bee population and his customer base at the same time
- Bees and honey came first



Software Lifecycle and FOSS

- Requirements
 - Usually no formal requirements analysis
 - Based on earlier releases
 - Projects may start proprietary
- Design/Implementation
 - Code reuse
 - Refactoring and rewrites

Software Lifecycle and FOSS cont.

- Testing, debugging, maintenance
 - Continuing evolution and incremental advancements
 - Testing distributed between users
 - Source code continually available for modification
 - Code reviews act as quality control

FOSS Development Tools

- Revision control (CVS, SVN)
- Tools to automate testing (xUnit), compiling (tinderbox), and bug reporting (Bugzilla, GNATS)
- Tools for communication (mailing lists, IRC, blogs, wikis)
- Software development management systems (GNU Savannah, SourceForge, BountySource)
- Package management (RPM, APT)

Pros and Cons

- Reliable, high quality software quickly and inexpensively
- Reuse
- Security Issues
- Lack of Documentation and Warranties
- Risk management
- Relies on informed and capable user base

FOSS in New Zealand

- New Zealand Open Source Society
 - <http://nzoss.org.nz/>
- Catalyst
 - “Specialists in Open Source Technologies”
 - Based in Wellington, 80 staff
- SilverStripe
 - Open Source Content Management application
 - Based in Wellington

Summary

- FOSS development is a different approach to the development of software systems
- Studies and research have just begun
- Many open questions:
 - Can FOSS be used to produce better software more efficiently?
 - Why do some FOSS projects succeed and others fail?
 - Why do people participate in FOSS projects?
- FOSS is no silver bullet