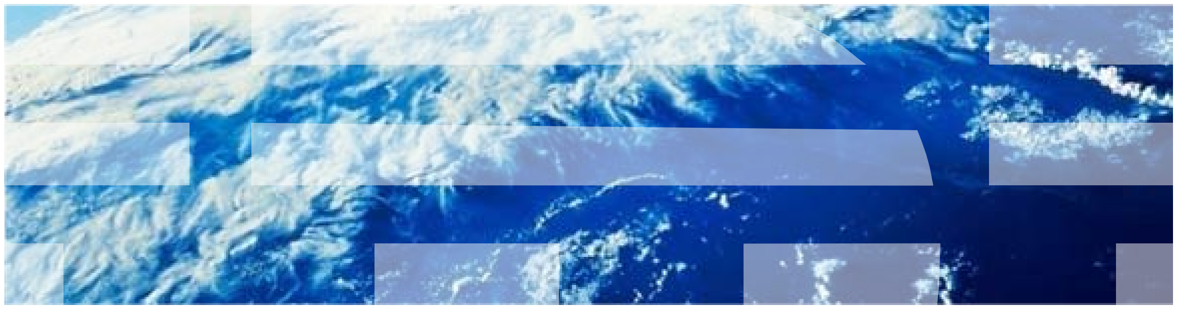
Open Source Software



What is Open Source Software?

- Open Source Software (OSS) is distributed with its source code. The Open Source Definition has three essential features:
  - It allows free re-distribution of the software without royalties or licensing fees to the author
  - It requires that source code be distributed with the software or otherwise made available for no more than the cost of distribution
  - It allows anyone to modify the software or derive other software from it, and to redistribute the modified software under the same terms.

Examples of Open Source Software

- Operating Systems
  - Linux
  - FreeBSD, OpenBSD, and NetBSD: The BSDs are all based on the Berkeley Systems Distribution of Unix, developed at the University of California, Berkeley. Another BSD based open source project is Darwin, which is the base of Apple's Mac OS X.

Examples of Open Source Software

- Internet
  - Apache, which runs over 50% of the world's web servers.
  - BIND, the software that provides the DNS (domain name service) for the entire Internet.
  - sendmail, the most important and widely used email transport software on the Internet.
  - Mozilla, the open source redesign of the Netscape Browser
  - OpenSSL is the standard for secure communication (strong encryption) over the Internet.categories.

Examples of Open Source Software

- Programming Tools
  - Zope, and PHP, are popular engines behind the "live content" on the World Wide Web.
- Languages:
  - Perl
  - Python
  - Ruby
  - Tcl/Tk
- GNU compilers and tools
  - GCC
  - Make
  - Autoconf
  - Automake,etc..

Open Source Software sites

- Free Software Foundation www.fsf.org
- Open Source Initiative www.opensource.org
- Freshmeat.net
- SourceForge.net
- OSDir.com
- developer.BerliOS.de
- Bioinformatics.org
- see also individual project sites; e.g., www.apache.org; www.cpan.org; etc.

Choose a tracker and you can browse/edit/add items to it.

Bugs ([illegible])
Bug Tracking System

Feature Requests ([illegible])
Feature Request Tracking System

Patches ([illegible])
Patch Tracking System

Sub-projects ([illegible])
[illegible]

Support Requests ([illegible])
Tech Support Tracking System

History of OSS

- 1984 – The Free Software Foundation (FSF) is Formed - Goal: to develop a free version of a UNIX-like OS, this was called the GNU project
- 1989 – FSF releases the GPL v1.0
- 1991 – the first code for Linux is released
- 1994 – Linux 1.0 is released
- 1994 – RedHat and Slackware versions released (C++, TCP/IP, Server)
- 1995 – Work starts on Apache
- 1996 – Work starts on KDE
- 1997 – “The Cathedral and the Bazaar” is published
- 1998 – the term “Open Source” is coined
- 2001 – Linux 2.4 is released

Main characteristics of F/OSS

- Free Software - Open Source Software: used interchangeably [F/OSS]
- Software development paradigm: collaborative and distributed development
- Licensing models: proliferation of “open source licenses (GPL, LGPL, BSD, etc..)
- F/OSS: potential benefits for interoperability and standards

ISSUES RELATING TO THE USE OF OSS

- Competition: adding competition to the market
- Lock-in: strong customer dependence on vendor
- Cost: total cost of ownership
- Reliability: software errors
- Maintenance: update cycle
- Sustainability: F/OSS as a lasting mechanism
- Capacity building: more S/W producers, small market segments better served

ISSUES RELATING TO THE USE OF OSS

- Innovation: large base of developers
- Product liability: a general problem; “good governance”
- Security and trust: unwanted functions; branding
- Education: educational tools
- Empowerment: customer has more decisive power and access to tools
- Equity: wider access to software

Software Development in Practice

- In the real world, software development is totally different and is more chaotic
  - Software professionals make mistakes
  - The client's requirements change while the software product is being developed
  - A software product is a model of the real world, and the real world is continually changing.

Conventional Models of Software Development

- Waterfall
  - from requirements to code without a backward turn
  - historically used for large military and corporate software productions; originally used because computing time was expensive
- Spiral
  - iterative cycles of requirements, development, testing, redrafting of requirements, etc.

Conventional Models of Software Development

- Rapid Prototyping
  - is a working model that is functionally equivalent to a subset of the product.
  - The first step is to build a rapid prototype and let the client and future users interact and experiment with the rapid prototype.

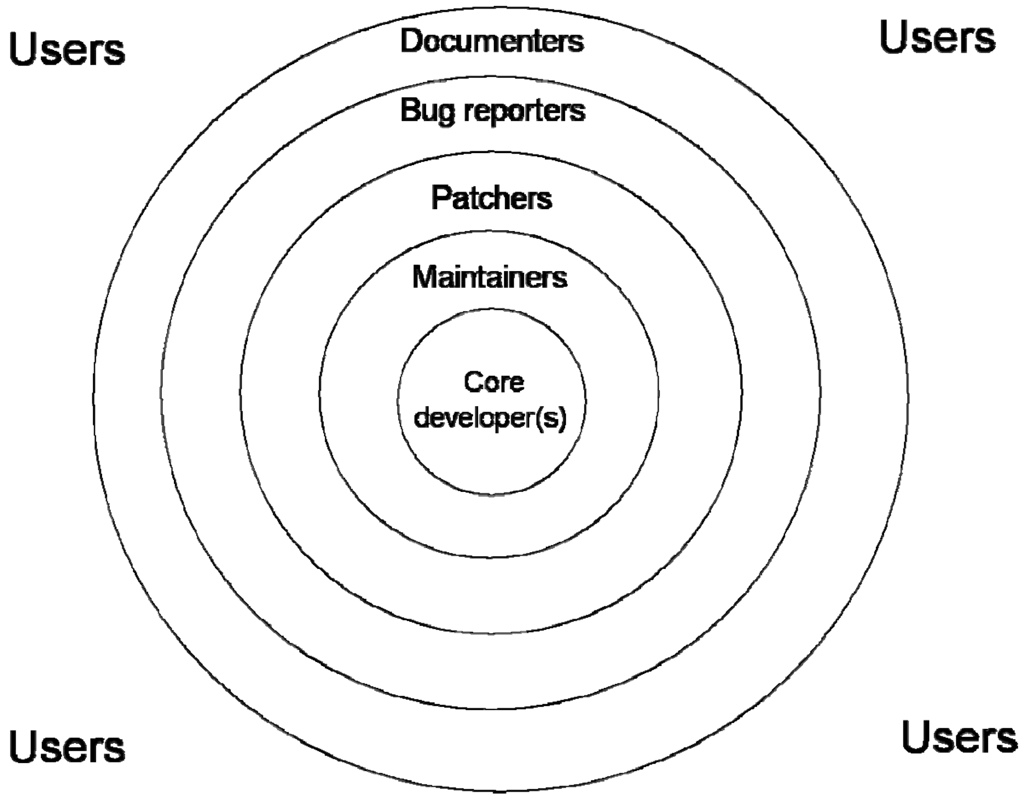
Open Source Software Development

- Open Source” is a software development and distribution methodology
  - Source is provided
  - Source can be distributed
  - Source can be modified
- The basic idea behind open source is very simple: When programmers can read, redistribute, and modify the source code for a piece of software, the software evolves.
- People improve it, adapt it, and fix bugs.

Open Source Software Development

- This can happen at a speed that, if one is used to the slow pace of conventional software development, seems astonishing.
- Free Software - Open Source Software: used interchangeably [F/OSS]

Open Source Software Development



Open Source Vs. Closed Source Software

CSS

- Developed by Companies and developers work for economic purposes.
- Centralized, single site development
- Users may suggest requirements but they may or may not be implemented
- Release is not too often. There may be only yearly releases.

OSS

- Developed By Volunteers work for peer recognition. People know that recognition as a good developer have great advantage
- Decentralized, distributed, multi-site development
- User suggestsadditional features that often get implemented.
- Software is released on a daily or weekly basis

Open Source Vs. Closed Source Software

CSS

- Market believes commercial CSS is highly secure because it is developed by a group of professionals confined to one geographical area under a strict time schedule. But quite often this is not the case, hiding information does not make it secure, it only hides its weaknesses

- Security cannot be enhanced by modifying the source code

OSS

- OSSD is not market driven; it is quality driven. Community reaction to bug reports is much faster compared to CSSD which makes it easier to fix bugs and make the component highly secure

- The ability to modify the source code could be a great advantage if you want to deploy a highly secure system

Open Source Companies

- IBM
  - uses and develops Apache and Linux; created Secure Mailer and created other software on AlphaWorks
- Apple
  - released core layers of Mac OS X Server as an open source BSD operating system called Darwin; open sourcing the QuickTime Streaming Server and the OpenPlay network gaming toolkit
- HP
  - uses and releases products running Linux

Open Source Companies

- Sun
  - uses Linux; supports some open source development efforts(Forte IDE for Java and the Mozilla web browser)
- Red Hat Software
  - Linux vendor
- ActiveState
  - develops and sells professional tools for Perl, Python, and Tcl/tk developers.

Licensing

- A license is basically an agreement between the user and the developer on how that software can be acquired and used.

- “Open Source” has specific meaning, tied to licenses endorsed by the Open Source Initiative (OSI)

- Software licensed to users with the freedoms:
  - to run the program for any purpose,
  - to study and modify the program, and
  - to freely redistribute copies of either the original or modified program (without royalties, etc.)

Licensing

- Grants permission to use a copyrighted work
- Can grant any or all of the rights associated with copyright
- Can impose other restrictions, such as type or place or usage, or duration of the license
- Does not transfer ownership of the copyright

- An open source licensor must give the licensee certain rights to be considered open source
- Basically, the licensee has the right to use, modify or distribute the software, and the right to access the source code.
- Open source software is software that is subject to an open source license.

What are the OSI and the OSD?

- The Open Source Initiative (OSI) is the de facto standards body for open source software. It determines what open source means, and approves licenses as being open source

- The Open Source Definition (OSD) is a set of criteria that a license must conform to be considered open source. The OSI maintains the definition and changes it from time to time.

Licensing

- The distribution terms of Open-Source Software must comply with the following criteria:
  - Free Redistribution
  - Source code
  - Derived Works
  - Integrity of the Author's Source code
  - No Discrimination Against Persons or Groups
  - No Discrimination Against Fields of Endeavor
  - Distribution of License
  - License Must Not be Specific to a Product
  - License Must Not Restrict Other Software
  - License Must Be technology-Neutral

The distribution terms of Open-Source Software

- Source Code: “The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well publicized means of obtaining the source code for no more than a reasonable reproduction cost, preferably, downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.”

The distribution terms of Open-Source Software

- Derived Works: “The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.”

- Integrity of the Author’s Source Code: “The license may restrict source-code from being distributed in modified form only if the license allows the distribution of ‘patch files’ with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.”

The distribution terms of Open-Source Software

- No Discrimination Against Persons or Groups: “The license must not discriminate against any person or group of persons.”

- No Discrimination Against Fields of Endeavor: “The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used in genetic research”

The distribution terms of Open-Source Software

- Distribution of License: “The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.”

- License Must Not Be Specific to a Product: “The rights attached to the program must not depend on the program’s being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program’s license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.”

The distribution terms of Open-Source Software

- License Must Not Contaminate Other Software: “The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.”

- License Must Be Technology Neutral: “No provision of the license may be predicated on any individual technology or style of interface.”

Open Standards

- Availability
  - Open Standards are available for all to read and implement.
- Maximize End-User Choice
  - Open Standards create a fair, competitive market for implementation of the standards.
  - They do not lock customer into a particular vendor or group
- No Royalty
  - Open Standards are free for all to implement with no royalty or fee.
  - Certification of compliance by the standards and the organization may involve a fee.

Open Standards

- No Discrimination
  - Open Standards and the organizations that administer them do not favor one implementer over another for any reason other than the technical standards compliance of vendor's implementation.
  - Certification organizations must provide a path for low and zero-cost implementations to be validated, but may also provide enhanced certification services.

- Extension or Subset
  - Implementation of Open Standards may be extended, or offered in subset form
  - However, certification organizations may decline to certify subset implementation, and may place requirements upon extensions.

Most Popular OSS Licenses

- The GNU “General Public License” (GPL)
  - No standard open source license, but GPL most widely used (roughly 85% of open source software);
  - Terms include:
    - User freedom to distribute and/or modify.
    - Requirement that original and modified source code be always made available to the world under the terms of the original license.
    - Must retain copyright notices and warranty disclaimers.
    - Does not include grant of patent licenses.

Most Popular OSS Licenses

- The Mozilla Public License
  - Developed by Netscape for the Mozilla browser
  - Terms include:
    - Very similar to the GPL but,
    - Can charge royalties for modified versions;
    - Can include source code within larger works licensed under different license types, thus license does not ‘infect’ all downstream projects;
    - Must retain copyright notices and warranty disclaimers;
    - May provide additional warranties to downstream users but may have to indemnify original developer for any claims arising as a result;
    - Includes grant patent licenses;

Most Popular OSS Licenses

- The IBM Public License
  - Terms include:
    - User freedom to distribute and/or modify;
    - No requirement for source code availability in downstream distribution;
    - The program can be distributed in executable form thus allowing downstream users to develop, sell, and install customized software packages without having to make all customizations available to the world;
    - Must retain all copyright notices and warranty disclaimers;
    - Includes grant of patent licenses.

Most Popular OSS Licenses

- Open Software License
  - Terms include:
    - User freedom to distribute and/or modify;
    - Viral license, source code is always made available to the world;
    - Must retain copyright notices and warranty disclaimers;
    - Requires indemnification for attorney’s fees incurred as a result of potential claims or litigation.

Most Popular OSS Licenses

- The Apache Software License
  - Governs the Apache web-server software.
  - Terms include:
    - User freedom to distribute and/or modify;
    - No requirement for source code to be made available to the world in downstream distribution;
    - Must retain all copyright notices and warranty disclaimers.

- The FreeBSD License
  - Unrestrictive license:
  - Only requires preservation of copyright notices and warranty disclaimers.

Community

- What is a community and why do open source projects want to build them?
  - Communities are a group of individuals sharing common interests.
  - Both closed and open source projects have communities of users, most of whom will be relatively passive in terms of their interactions with other community members.
  - On the other hand, either type of community may have members who decide to take on more active roles

Community

- To participate in an open source software community
  - Prepare
  - Get to know your community
  - Engage and give back

Get to know your community

- Understand how the community communicates
  - Most communities have an accepted way of engaging with the community. This can be as simple as joining a developers' email list, and/or adopting a particular code of practice. Most important, perhaps, is to learn how questions are asked and answered in this particular development community.
- Understand how the community is governed
  - Some communities, for example, that of the Linux kernel, are hierarchies with clear chains of command
  - Understanding how decisions are made and conflicts resolved will help you understand how to best engage with the community.

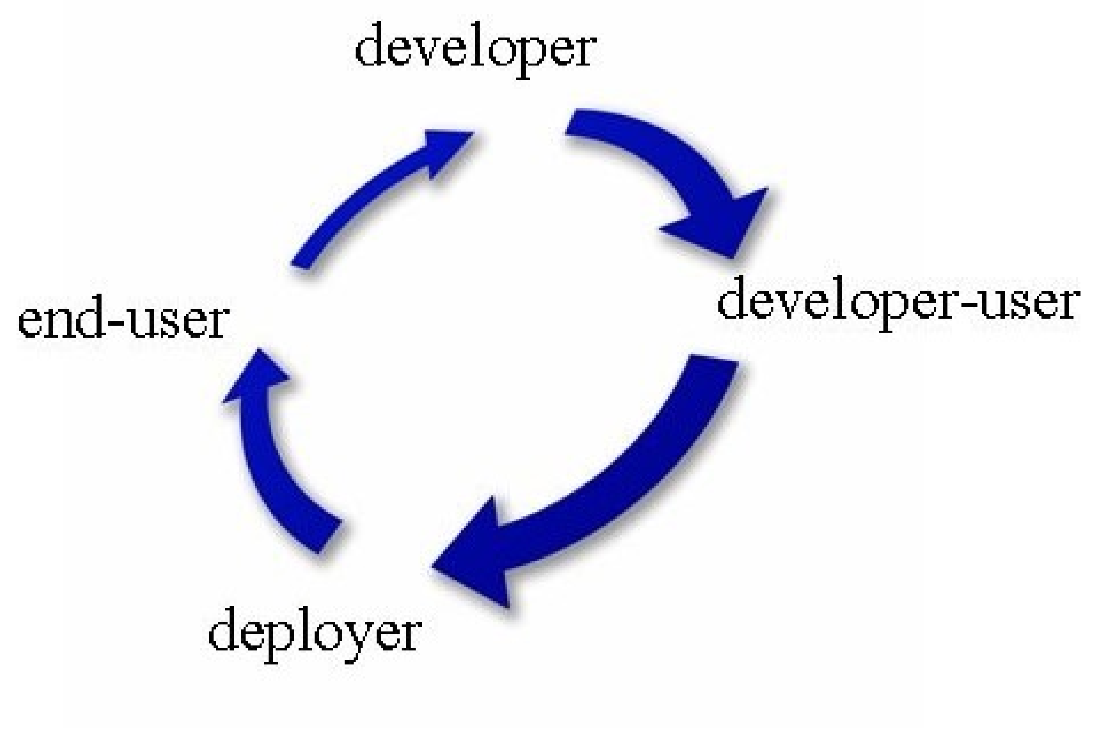
Get to know your community

- Understand the communications channels
  - Open source projects typically use chat sessions (typically IRC or jabber), mailing lists, websites, blogs, wikis, and version control repositories as their primary means of communication. Get to know your project's communication style and preferred tools.

Engage and give back

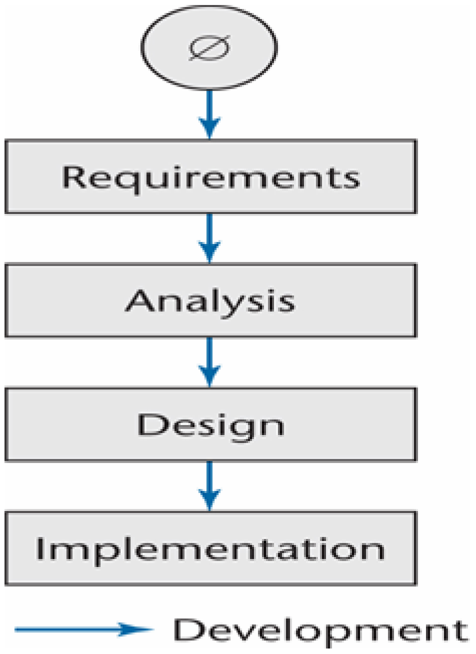
- Give back
  - A healthy community depends on the principle of individuals giving back to that community.
- Plan an exit strategy
  - Have a plan for what happens to your contributions to the open source project when your commitment to the project changes.
- Retire gracefully
  - if your open source participation is waning due to work, family or other commitments, inform other community members of the problem, so that they can take up the slack and/or take steps to lower your workload.

OSS People Cycle



Software Development in Theory

- Ideally, software is developed as described in the last lecture
  - Linear
  - Starting from scratch



Open Source vs. Closed Source

- **Closed Source software** is maintained and tested by employees
- Users can submit failure reports but never fault reports

- **Open Source software** is generally maintained by unpaid volunteers
- Users are strongly encouraged to submit defect reports, both failure reports and fault reports
- Core group: Small number of dedicated maintainers with the inclination, the time, and the necessary skills to submit fault reports (“fixes”); They take responsibility for managing the project; They have the authority to install fixes
- Peripheral group: Users who choose to submit defect reports from time to time

Open Source vs. Closed Source

- New versions of **closed source software** are typically released roughly once a year
- After careful testing by the SQA group

- The core group releases a new version of an **open source product** as soon as it is ready
- Perhaps a month or even a day after the previous version was released
- The core group performs minimal testing
- Extensive testing is performed by the members of the peripheral group in the course of utilizing the software
- “Release early and often”

Some popular bug trackers include:

- Bugzilla - a sophisticated web-based bug tracker from the Mozilla house.
- Mantis Bug Tracker - a web-based PHP/MySQL bug tracker.
- Trac - integrating a bug tracker with a wiki, and an interface to the Subversion version control system.

Some popular bug trackers

- Request tracker - written in Perl. Given as a default to CPAN modules - see rt.cpan.org.

- Fossil -written in C, and uses SQLite database. Apart from bug tracking, it also provides Wiki.

- EventNum -This was developed by the MySQL team, and written in PHP.

Documentation

- One way to ensure basic initial documentation gets done is to limit its scope in advance. That way, writing it at least won't feel like an open-ended task.
- A good rule of thumb is that it should meet the following minimal criteria:
  - Tell the reader clearly how much technical expertise they're expected to have.
  - Describe clearly and thoroughly how to set up the software, and somewhere near the beginning of the documentation, tell the user how to run some sort of diagnostic test or simple command to confirm that they've set things up correctly.

Documentation

- Give one tutorial-style example of how to do a common task. Obviously, many examples for many tasks would be even better, but if time is limited, pick one task and walk through it thoroughly
- Label the areas where the documentation is known to be incomplete. By showing the readers that you are aware of its deficiencies,

The Cathedral and the Bazaar

- Eric Steven Raymond
  - December 4, 1957
  - Fetchmail, gpsd, emacs editing modes
  - "The Cathedral and the Bazaar", published in 1997 .
  - Became a prominent voice in the open source movement
  - Co-founded the Open Source Initiative in 1998

Origins

- First version of the paper written in 1997.
- Several revisions appeared up to 2000.
- The author discovers a \development model" through the history of the Linux kernel and an own tool.
- This model is presented as revolutionary, since it is useful to build large systems without apparently any or few organization at all.

Models

- The Cathedral: The “classic” model.
  - Closed environment.
  - Small group of leaders/developers.
  - Only \stable" releases on, in some cases, \betas".
  - Used both in classic development models, such as waterfall,
  - spiral etc; and in classic OSS projects.
  - Examples: GCC, GNU Emacs.

Models

- The Bazaar: The model introduced by Linus Torvalds.
  - Open environment, almost any person can participate.
  - There are no clear leaders, undefined number of developers.
  - However, there is a benevolent-dictator figure.
  - “Release early, Release often”.
  - Examples: Linux.

Models

- The Bazaar style of development:
  - with a community seemed to resemble a large babbling bazaar of diverse agendas and approaches
  - with archive repositories where anyone can propose a modication
  - but from this, a stable and coherent large system emerges.

- This was surprising:
  - Why Linux world did not fly apart in confusion?
  - and why Linux seemed to go from strength to strength at a speed barely imaginable to cathedral-builders?

Eric Raymond Principles

- Every good work of software starts by scratching a developer's personal itch.
  - Most successful free software projects have been started by developers with needs addressed by their \pet" project.
  - In the world of proprietary software, programmers spend their time building programs that they neither need nor want.
  - This motivation could explain the high quality of results given by Linux.

Eric Raymond Principles

- Good programmers know what to write. Great ones know what to rewrite (and reuse).
  - Linus Torvalds did not try to write Linux from scratch.
  - Instead, he started by reusing Minix code and ideas.
  - Although today all reused Minix code has been removed or rewritten, while it was there, it provided scaffolding for the infant that would eventually become Linux.
  - The source-sharing tradition of the Unix world has always been friendly to code reuse.

Eric Raymond Principles

- If you have the right attitude, interesting problems will find you.
  - Eric's problem was that he needed a POP protocol client to work with.
  - And he found an abandoned one.
  - The problem was the continuation of the abandoned client, and Eric took it over and started to coordinate it.

Eric Raymond Principles

- When you lose interest in a program, your last duty to it is to hand it off to a competent successor.
  - Before abandoning the development of a free software, we should find another person to continue its development.
  - Fortunately, in the bazaar world, some other hacker will find your abandoned work soon, and will start to develop it for his own needs.

Eric Raymond Principles

- Treating your users as co-developers is your least-hassle route to rapid code improvement and effective debugging.

  - In Linux, many users are hackers too.
  - Because source code availability, these users can be e effective hackers.
  - This can be useful for shortening debugging time.
  - These users will diagnose problems, suggest fixes, and help in improvements.

Eric Raymond Principles

- Release early. Release often. And listen to your customers.
  - This is another Linux characteristic: during very active development periods, lots of versions were released.
  - Sometimes, more than one in a day.
  - This maintains the hackers constantly stimulated and rewarded:
  - stimulated by the prospect of having an ego-satisfying piece of the action.
  - and rewarded by the sight of constant (even daily) improvement of their work.

Eric Raymond Principles

- When the code is getting both better and simpler, that is when we know it is right.

- At this moment, the software maintained by Eric was, not only very different, but also simpler and better. It was time to change its name and give it its new identity: “fetchmail” instead of “popclient”.

- Q’s
- Feedback