Open Source Software Development

Introduction to Open Source Software (OSS)

Thái Minh Tuấn - Email: minhtuan@ctu.edu.vn

Learning Objectives

- What is Free and Open-Source Software (FOSS)?
- Proprietary (Closed Source) Software
- OSS philosophical strains: Pragmatism vs. Idealism
- History of Open Source Software
- Reasons for Using Open Source Software
- Successful OSS Projects

What is Free and Open-Source Software (FOSS)?

- Free as in freedom to do something
  - NOT as in no cost
- Source code is made available
- With a license which provides rights (4 essential freedoms):
  - To run the program as you wish, for any purpose (freedom 0)
  - To study how the program works, and change it (freedom 1)
  - To redistribute copies (freedom 2)
  - To distribute copies of your modified versions (freedom 3)
  - Without restriction on the user's identity or purpose
- There is a multiplicity of licensing methods, falling into the two general classifications:
  - Restrictive (e.g., GPL-licensed software) vs. Permissive (e.g., BSD-licensed software)
  - (will discuss later)

Proprietary (Closed Source) Software

- Only the owners have full legal access to the source code
  - May grant inspection rights to trusted partners with non-disclosure agreements (NDAs).
- The owners may or may not be the authors of the code
  - The authors have been working under contracts giving up their individual rights
  - Many proprietary softwares are no longer even owned by their originators
    - Passed on through sales over time
- End users must accept a license restricting their rights
  - Through a confusing click-through box presented after a very long and dense description
- Price is not the point:
  - One can charge as much as one wants for an open product
  - One can provide a proprietary product for free
- The license differences have to do with redistribution, modification, reuse of code, etc.

OSS philosophical strains: Pragmatism vs. Idealism

- Idealism: There is a profound belief that all softwares should be open for ideological and ethical reasons, not just technological ones.
- Pragmatism: The primary considerations are technical ones, including faster and better development involving more contributors and review, easier debugging, etc.
- Ideological perspective also has strong technical imperatives, and in many cases, the objectives of both streams coincide
  - Should the software powering a life-saving medical device (such as a pacemaker, or an insulin pump) be secret?
  - Should the software powering voting machines be closed

History of Open Source Software (1/4)

- Open Source Software (OSS) has a long history, although the actual term only dates back to 1998:
- 1950s:
  - Software arose from researchers, both academic and corporate
  - Distributed openly and cooperatively
  - Source always distributed, binaries less often
  - Software not seen as separate commodity
  - Software bundled for free with hardware
  - Licensing was sloppy
- 1960s:
  - Major aspects of computer science and software rapidly developed both in academia (MIT, UC Berkeley) and industrial research labs (Bell Labs, Xerox)
  - 1968: ARPANET emerges, eventually leads to Internet, essential for developers and researchers to communicate, share, collaborate
  - 1969: UNIX given birth at Bell Labs (AT&T) and given for free to universities and research centers
  - 1969: IBM forced by US government to break software and hardware apart and sell/distribute separately, due to unfair business practices.

History of Open Source Software (2/4)

- 1970s:
  - 1976: emacs released by Richard Stallman and Guy Steele. Versions by other authors proliferate, including proprietary ones. GNU emacs not released until 1985
  - 1978: First version of TeX released by Donald Knuth, an open source typesetting system often used for publishing journal articles and books, still in widespread use, usually in the LaTeX version.
- 1980s:
  - 1980: Usenet begins as the ancestor of user forums and the World Wide Web
  - 1982: GNU project announced by Richard Stallman
  - 1984: X Window System released out of MIT, with X11 protocol released in 1987 (now run by X.Org)
  - 1985: Free Software Foundation (FSF) founded by Richard Stallman
  - 1987: gcc released (now known as Gnu Compiler Collection)
  - 1987: Perl released by Larry Wall.

History of Open Source Software (3/4)

- 1990s:
  - 1991: Linux begun by Linus Torvalds
  - 1992: Python released by Guido Van Rossum
  - 1992: 386BSD released
  - 1992: Samba developed by Andrew Tridgell in Australia
  - 1993: Debian first released by Ian Murdock, and still survives as the largest non-commercial Linux distribution
  - 1993: Red Hat is founded; while earlier commercial Linux distributions already existed, Red Hat was the first company built on open source to become very large
  - 1993: NetBSD released as a fork from 386BSD
  - 1993: FreeBSD released
  - 1993: Wine released to run Windows applications on Linux
  - 1994: MySQL development begins in Sweden, first release in 1995
  - 1995: PHP, GIMP, and Ruby released
  - 1996: Apache web server released
  - 1996: KDE released
  - 1997: GNOME released
  - 1998: Netscape open sources its browser, which will later become Firefox
  - 1999: OpenOffice released (eventually forks into LibreOffice).

History of Open Source Software (4/4)

- 2000s:
  - 2000: LLVM compiler project begun at UI-Champaign
  - 2002: Blender released as an open source project
  - 2003: Firefox released
  - 2004: Ubuntu releases its first version, which Canonical builds on top of Debian
  - 2005: Git released by Linus Torvalds
  - 2007: Android released, based on Linux kernel; first devices on the market in 2008
  - 2008: Chromium released by Google; basis of Google Chrome.
- 2010s:
  - In this decade, OSS becomes ubiquitous, dominating the smartphone market, supercomputing, worldwide networking infrastructure, etc.

Open Source Governance Models - Company-Led

- Company-Led (mostly, a closed process): Google Android, Red Hat Linux
  - Development is strongly led by one corporate or organizational interest
  - One entity controls software design and releases
  - May or may not solicit contributions, patches, suggestions, etc.
  - Internal discussions and controversies may not be aired very much
  - It is not definitively known what will be in the next software release
  - Upon release, all software is completely in the open

Open Source Governance Models - Benevolent Dictatorship

- Benevolent Dictatorship (strong leadership): Linux kernel
  - One individual has overriding influence in every decision
  - Project's quality depends on that of the dictator
  - Can avoid endless discussions and lead to quicker pace
  - As project grows, success depends critically on the dictator's ability to:
    - Handle many contributors
    - Use a sane, scalable version control system
    - Appoint and work with subsystem maintainers
  - The dictator's role may be social and political, not structural (forks can occur at any time)
  - Maintainers write less and less code as projects mature

Open Source Governance Models - Governing Board

- Governing Board (tighter control by smaller groups): FreeBSD, Debian
  - A body (group) carries out discussions on open mailing lists
  - Decisions about design and release dates are made collectively
  - Decisions about who can contribute, and how patches and new software are accepted, are made by the governing body
  - There is much variation in governing structures, rules of organization, degree of consensus required, etc.
  - Tends to release less frequent, but hopefully well-debugged versions

Exercise: Basic Facts about Open Source Software

- Which of the following statements are true?
  - 1. Once fees are charged for for the use or distribution of software, it can no longer be termed open source.
  - 2. If the control, content and timing of the official software releases is controlled by one entity (such as a company), it can no longer be termed open source.
  - 3. The first widely used OSS product was the Linux kernel, first released in 1991.
  - 4. If a program is written using an open source computer language (such as Python or Perl) it must be released under an open source license.

Reasons for Using Open Source Software - Collaborative Development

- Collaborative Development: Enables software projects to build better software
  - NOT everyone has to solve the same problems and make the same mistakes
  - Softwares can be made much faster and costs can be reduced
  - Stronger and more secure code
    - More eyeballs viewing code and more groups testing
  - It is often hard for competitors to get used to the idea of sharing
    - Competitors can compete on user-facing interfaces
    - End users still see plenty of product differentiation and have varying experiences

Reasons for Using Open Source Software - Security and Quality of Source Code:

- Code published openly tends to be cleaner
  - It is embarrassing to show ugly, sloppy code
  - Coding standards and styles tend to be cleaner and more consistent on community projects
  - More people have to understand and work on the code
- More eyeballs examining code and looking for security weaknesses
  - Before they are discovered by bad actors
- There is more input in original design to avoid bad ideas
- No *security through obscurity*
- No *just trust me*
- Potentially faster bug repair

Reasons for Using Open Source Software - Advantages for Various Stakeholders (1/3)

- Individual users:
  - Can mix and match software from different sources
  - Can save money on buying and leasing software
  - Can avoid vendor lock-in, maintaining choice
  - Can look under the hook (“trust, but verify”)
  - More funs
- Business:
  - Collaborative development
    - Lowers total cost of development
    - Speeds up time to market
    - Work is submitted to wider community for criticism, suggestions and contributions
    - Uses well-delineated application programming interface (API)
  - Marketing
    - Customers know what they are getting - They have confidence in quality, there are no secrets
    - Product is seen as part a large ecosystem of related products
    - More flexible, possibly modular construction
    - Adoption by larger community can help build customer’s confidence about product’s durability and stability

Reasons for Using Open Source Software - Advantages for Various Stakeholders (2/3)

- Education - Elementary, High school and Public systems:
  - Very large amount of available teaching resources at little and no cost
  - Very wide range of areas available for using, operating and system administration, and programming
  - Students do not become locked into vendor products
  - Generally lower hardware cost, and easier to use old hardware
  - Student are learning the skills they will need in the workforce
  - Unleashes student creativity: more fun!
- Education - University:
  - Students can study and work on the internal of operating systems, applications and libraries, system administrator utilities
  - Student are ready to enter to the workforce where they are most needed
  - Good habits are developed, including how to work with the open source community
  - Student work is easy for prospective employers to evaluate, since it is publicly accessible

Reasons for Using Open Source Software - Advantages for Various Stakeholders (3/3)

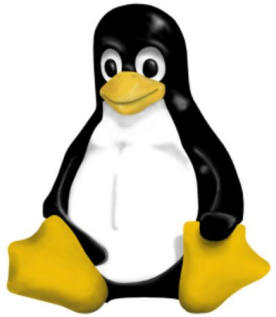
- Developer
  - No need to reinvent everything
  - Help to make good, early decisions on product design
  - More eyeballs on code can fix bugs faster
  - Suggestions and contributions are provided by a large group of developers
  - Great for finding next job
    - Code is readily available for evaluation
    - Can demonstrate how well you work and play with others
    - Can show how good you are at mentoring and maintaining projects and subprojects
    - Know you are not alone

OSS - Fear, Uncertainty and Doubt (FUD)

- Misinformation to influence recipients to avoid certain strategies, products or classes of products by appealing to fear
- Microsoft was widely accused of spreading FUD about Linux in the 1990s.
  - In present day, Microsoft has stopped doing so and is actually employing OSS widely
- OSS - FUD statements
  - OSS is a virus. If you include it in your product, all your source must be made available to everyone
  - OSS infringes on software patents, and the related claim that it forces you to grant patent rights to others
  - OSS products leave nowhere to turn when they break, or to get technical help
  - OSS requires a lot of legal help to avoid the above pitfalls, and is thus very expensive

Successful OSS Projects - Linux Kernel

- Has been an open source project since its inception in 1991
- Basis of almost all of the world's computing infrastructure
  - From the most powerful supercomputers to the largest number of mobile devices
- Become ubiquitous in an enormous range of devices and embedded products:
  - Personal fitness devices , DVR boxes and televisions, automotive systems, In-flight entertainment systems, medical devices, etc.
- Development community is large and mature; headed by Linus Torvalds
  - Well-formed organizational structure of subsystem maintainers and mailing lists, contributor guidelines and methods
  - Thousands of developers
    - From hardware and software companies; industrial collaborative consortia
- Released under GPL Version 2
  - Some sections that have dual licensing, with other open source licenses
  - New version is put out every 10-12 weeks



Successful OSS Projects - Git

- A distributed version control system
  - Used worldwide for an astounding number of collaborative products
- Created by Linus Torvalds in 2005
  - To handle the increasingly difficult task of coordinating and consolidating the work of thousands of contributors to the Linux kernel
- The basis of GitHub (https://github.com)
  - Hosts more than one hundred million of open source projects repositories
- Can be used in non-open source projects (like many other tools)
  - No way contaminates using it for OSS work

Successful OSS Projects - Apache

- Work on the Apache HTTP Server began in 1995.
  - The most widely used web server, with roughly 50% of the market share
- Operates under the umbrella of the Apache Software Foundation (ASF)
- Released under the Apache Software License
  - Has also been adopted by many other projects.
  - It is more permissive than the GPL

Successful OSS Projects - Python, Perl and Other Computer Languages

- Many computing languages are developed using open source methods.
  - Python is an interpreted, high-level, general-purpose programming language
    - The most popular programming language[1]
  - Perl is a programming language specially designed for text editing
    - Now widely used for a variety of purposes including Linux system administration, network programming, web development, etc.
  - Many others such as Ruby, GCC, and Rust

1. https://statisticstimes.com/tech/top-computer-languages.php

Successful OSS Projects - GNU: gcc, gdb, and More

- GNU (Gnu's Not Unix) Project has provided many essential ingredients for virtually all modern computer technologies,
  - Under various versions of the GPL (General Public License)
  - Started in 1983 by Richard Stallman
- Some of the most prominent products emanating from the GNU umbrella
  - GNU Compiler Collection (gcc): Compiler system supports various programming languages
  - GNU Debugger (gdb): Portable debugger that runs on many Unix-like systems
  - GNU C Library (glibc): GNU Project's implementation of the C standard library
  - GNU Bash (bash): Unix shell and command language used widely as the default login shell for most Linux distributions
  - GNU Core Utilities (coreutils): A package containing re-implementations for many of the basic tools, such as cat, ls, and rm, which are used on Unix-like operating systems

Successful OSS Projects - X and Desktop Managers

- Used to instantiate the GUI seen on any Linux laptop or workstation
- X Window System: The underlying software that handles basic screen, input device and other point operations
  - A newer and more secure alternative, Wayland, is moving into its current role
- GNOME, KDE, XFCE: Desktop Manager frameworks that control the operation of graphical interfaces, drag and drop between them, appearance of the desktop

Successful OSS Projects - OpenStack, Kubernetes and Other OSS Projects

- Many other large scale (as well as small) collaborative projects that are based on open source software