INFO212

Induvidual Assignmnet

Open-Source Development

Candidate number:

176

December 14, 2016

Contents

1 Introduction

Open-Source software is among the most important software in use today. It powers close to 86% of all mobile phones[1], and its estimated that about 75% of the worlds top websites are run by open-source software[2]. In this essay we will take a look on how open-source code is written, the tools used and the strategy, along with a few examples of them in use. We will also touch on some of the debate around open-source software versus free software.

2 Open-Source

Open-source has come a long way inn general. This essay is written on largely free and open-software, with a few exceptions. The computer runs Linux, one of the largest open-source projects inn the world. The essay is written inn Vim, descendants from one of the oldest editors in the world. It's written with latex, which uses a open-source toolchain and compiles down too the open PDF standard. Some of these tools are older then me, some are barley younger. With the rise of social coding platforms, we have seen an increase inn code written openly on the web.

However, redistribution of code isn't a new thing. It's been prominent inn the academia for many years prior to the 1980. As well as the rise of the homebrew computer club inn Palo Alto during the 1970s. Arguably, the world wide web as we know it today is built on the very idea of a collaborated effort to share ideas and code. CERN gave the original web server and web browser away for free[3]. They wrote their own license for this that later became one of the most used licenses today, the MIT license. We will take a look at it later.

Richard Stallman established the Free Software Foundation and the GNU Project inn 1983. The GNU project had a goal to write a complete open and free operating system, as most operating systems during this era was proprietary and not open-source[4]. A popular example is your printer does not work, you could fix it if the source was open. But with a proprietary system this was not possible. Leaving you with a broken system with no

other way to fix it but to contact the manufacturer. Stallman was out to change this. They created many of the tools commonly used for software development today. The concept of open-source was still in its infancy during the 1990s, it was not before Linux arrived in 1991 that the GNU project was able to fulfill their goal of creating a complete open and free operating system.

2.1 Free versus Open-Source Software

There are two large camps in the world of Free and Open-Source software. On one side you have Richard Stallman with his Free Software Movement, and on the other side Eric S. Raymond with the Open-Source initiative. The Free Software Movement redefines the term "free" to the word of "libre"[4], you can modify the code, but you have to redistribute the code and make it open. They reject the notion that the use of computers should prevent people from co-operating. This leads to the rejection of proprietary software that Stallman very heavily argues against.

The Eric S. Raymond however defined the term Open-Source. Following his essay "The Cathedral and the Bazaar" Netscape released their source code of the netscape browser. The advocacy and the philosophies of the Free Software Foundation didn't appeal to companies. So they wanted a better term that could incorporate the ideas, yet be friendly to companies. The Open Source Initiative was established. They published guidelines inspired by the Free Software foundation, yet more open and less hostile[5].

It is important to realize that the terms "open-source" and "free software" have a very rigid definition. These definitions are heavily represented by the licenses they push forward. GPL for the Free Software Foundation, and the MIT/BSD/OSI approved licenses from the Open Source Initiate.

2.1.1 Licenses

Licenses are important in the world of open-source. Normally the writer retains the copyright of the code written, making any open code written without a license on shaky ground as you are at the mercy of the writer, and not the license.

There are two types of licenses. Free, or libre, licenses like the GPL, and the permissive licenses like BSD and MIT. And depending on what license you use, you might not be able to use the code for your software.

Stallman invented the widely used GPL license. GPL allows the end use to modify and redistribute the code. It is also a copyleft license which means derivative work has to be redistributed with the same license. This goes both ways. This makes it harder to developers to figure out what licenses are compatible with each other. Microsoft for instance has been very harsh critics of GPL, as any libraries written with GPL licensed code has to be redistributed as such.

On the other side of the specter, we have the BSD and MIT licenses. Both licenses put a very limited set of restrictions on reuse, and in turns makes it compatible with other licenses such as GPL. MIT and BSD allows anyone to do anything with the provided software, but the originally writer can not be hold liable for any damage cause by the software. The simplicity of the licenses makes it the most widely used licenses today.

The interesting part of this is that for you to call your project "Open-source", you would need to use a OSI (Open Source Initiative) approved license[6]. The Free Software Foundation and the Open Source Initiate both publish a list of licenses they approved[7]. Both the GPL and the MIT license are both approved by the FSF and OSI alike. So anyone could use either licenses and mix them in their project, but you would be forced by the GPL to redistribute the code. There is also a concept about dual-licenses, where as you can licenses one part of your code as MIT, and another as GPL. But we wont dive to much into this in this essay.

The discussion about using a permissive license of a libre license is today a very big and political discussion. Some people believe a license like GPL protects the freedom of the user, because of redistribution. While other choose something like the BSD or the MIT because they equal "true" freedom. This is largely a ongoing argument in the deep ends of the Internet.

Today MIT is the most widely used license for open-source work. However, several large projects use the GPL as its more reasonable when you want contributions in return. One example of this is Linux.

2.2 The Cathedral and the Bazaar

One of the largest influences on the Open-Source community inn the early 2000 was the essay "The Cathedral and The Bazaar", referred to as "CatB" written by Eric S. Raymond[8]. The essay looks at the development of the Linux kernel, which we will take a look at later in this essay, and his own experiences developing fetchmail. CatB was a large influence, it helped Netscape open-source their code for the Netscape Webbrowser, which formed Firefox and started the Mozilla project. It gave rise to several prominent projects.

The essay proposes that "Given enough eyeballs, all bugs are shallow", which means that given enough developers and testers, all bugs are easy to fix and correct. This could be in the form of code reviews where several developers look at the written code, and tests it. Thus its easier to fix and find flaws inn the code early on. This has later been formulated as Linus's Law.

The essay explains the difference between two models of developing free and open software, The Cathedral and The Bazaar model of development.

2.2.1 The Cathedral

The Cathedral model is where the source code is available between every release of the software. The system is thus only shared between a number of developers in private. This means you can't look at code in between releases. You could look at a log of all the changes and not really know what code went where. This style has largely fallen away with the rise of version control systems, but is still used in some software, notably TrueCrypt, a now deprecated disk encryption software, and Putty, a widely used software to managed servers with SSH.

2.2.2 The Bazaar

The Bazaar models is how most of the open-source work today is done today. The idea is to make development transparent and open. You should be able to read the changes, see the changes and follow the development of the project. With the spawn of social programming websites like bitbucket, gitlab and github this is easier then every before. This ranges from the Linux kernel, several prominent programming languages such as Golang, Rust, Clojure, Python and Ruby. Among compilers for C and C++. It's overall a very successful model of development and with the rise of social programming platform it become easier to start contributing.

This model allows for what Eric S. Raymond dubbed Linus's Law. As the code is always available, at any state during the development. It is easier to debug, test and get the needed reviews for the code.

3 Git

Version control software, VCS for short. Is a program that keeps track of changes inn a software project. It keeps track of added and deleted lines of code in a folder, and enables you to commit these changes to a repository. These repositories then lets you have multiple parallel sets of changes into different branches. This enables developers to keep track of features, changes and version their software to a greater extent. This is crucial in todays software development.

Git is a distributed version control system created by Linus Torvalds[9]. It was mainly created to replace the proprietary solution used by the Linux developers. The goal of the project is to create distributed version control system. By being distributed developers could stay offline for an extended period of time, working on their own local copy of code, and merge it back together when needed, instead of having to stay in-sync with the current status of the code. This is a interesting feature compared to SVN, which is another often used VCS. SVN only allows commits to the trunk if you can reach the source repository. This prevents developers from working on the project away from work, if the repository is behind a firewall. With the distributed nature of Git, this is not a problem.

4 Social Coding Platforms

Social coding platform has a been a game changer for the free and open source software world. More software is getting released now then ever, and its largely because of websites and services supporting these activities. One of the largest of these sites is Github. Github allows you to create free open repositories for your code. They also introduce an option to fork other project, simply to clone and host a copy on your own profile, and send back pull-requests. These pull-requests lets you contribute with changes to a project you don't have direct access to. This way you can propose changes, figure out bugs or errors early on.

This adheres to the Bazaar model discussed earlier. Everyone is able to review the code and changes. The changelogs for the project could reference these commits, and pull-requests which as well doubles as a discussion platform for the proposed changes. A lot of large projects use this for their discussions. Docker is a software allowing developers to create lightweight containers. Rust is a up and commit low-level programming langauge, and Golang, a new programming language from google. All these projects use Pull-Requests as a discussion platform.

Github started out inn 2007 as a part-time project by Tom Preston-Werner. Today it hosts around 49 million projects, for its estimated 14 million users[10]. It has also spawned a pawed way for a lot of the projects and conferences we see today. There has also been created alternatives for Github with different models.

Github allows users to create unlimited code repositories. To create private ones you have to pay. In contrast, bitbucket allows anyone to great private repositories, but limits the amount of contributors that share any repository. This enables students with projects to utilize bitbucket better then github, as you want your projects to not be visible.

There are also open-source github clones, like gitlab. They allow you to self-host your own gitlab instance. This is great for a company that can't use any external services for their code. There are also several smaller projects that enables you to just display information from git without any extra features like bug trackers.

4.1 Development Methods

Git itself does not enforce any way of how development should be done. A lot of people working alone on projects commits straight to the master branch. While some projects utilize other models on how code should be commited.

A few popular examples are the git-flow model[11], and feature-branches. Feature-branches is essentially a way of saying that one branch should contain one feature, or one new addition to the project. This enables you to manage each merge as a feature, and easily revert changes that should not work out. This also enables you to expand into more complex workflows, like git flow.

Git flow is a structure where you have 2 long lived branches. One master, used as a production branch. All code going into this branch should be vetted, and deployable.

4.2 Tools

There is a large variety of tools available to compliment the development of open-source software. This could be in the form of supporting testing and integrations. Keeping track of bugs, and issues. There are also numerous systems to help developers and projects to deploy into production systems. We will take a look at different tools that support developers of open-source projects and companies in this section.

4.3 Bugtracking

Bugtracking is an essential part of a open-source project. It enables you to label, track, and assign bugs, issues or planned features of a project. Linux uses Bugzilla with tight integration to the mailing list to track bugs. This enables the developers to keep their main attention to the mailing list. Github includes a bugtracker, but some projects feel it's bad to centralize everything and choose other providers. This helps a lot if it would ever happen that github goes down. The Tor Project creates privacy enhancing tools for the Internet and relies on the open-source trac bugtracker for their issues and bugs.

4.4 Testing & Build tools

Testing and building is essential for any projects, not only open-source. It enables the project to verify and test the code they are writing.

Buildbots is a open-source build automation software[12]. It enables the project creator to keep build created along with the code commits to the repository. This is heavily used by projects like chromium, googles open-source web browser which chrome is built on. This lets chromium to build and offer experimental builds straight from their git repository, without having the user to manually clone and build the webbrowser them self, which can be a complicated task.

Travis is a proprietary product that integrates heavily into github[13]. It enables both public and private repositories to create multiple testing suites for their project. This is great as feedback can be directed into the pull-request on the github website, instead of being a separate process. This makes it easier for the individual contributor to receive feedback and check the errors.

Jenkins is a widely used open-source tool[14]. It allows you to setup testing and deployment for your systems. Since it can be self-hosted it is also heavily used internally by companies for testing and deployment. It is written in java and has a great open-source ecosystem for integration between different systems. You could setup git integration between github and bitbucket. Do the testing using docker container and scale across a cloud. And when done, you could deploy your product to the cloud provider of choice; google, amazone or azure.

There is also an own category of testing tools for code. These are called linters and are used to catch coding mistakes, and code not conforming to a given coding standard. These can be used along with tools like jenkins to make sure the code submitted is within whats accepted of the community.

4.5 Containers and virtual environments

Inn recent years technology like containers and virtual environment has emerged from open-source communities. Technologies like Vagrant and Docker enables developers to not care about the underlying operating system they develop on. But can initiate thin layers to emulate different operating systems. This enables rapid prototyping and easier testing as the environment does not have to be created, but can be emulated with configuration files.

Several tools like Travis and Jenkis has started to rely on this technology as it scales easier between machines and enables you to create small clouds. This can make testing for complex software that needs multiple concurrent test more efficient.

4.6 Package management

Package management is important in the development of software[15]. Packages are one application, its dependency list, its source files and usually its compiled form. This enables you to keep track of the installed software, libraries and needed dependencies on your computer. They are usually stored in a centralized repository. This is crucial in today open-source world as installing this by hand is hard, and often complicated. A single dependency for your project could contain 3 dependencies, who might have 12 other dependencies together. A package manager simplifies this process. The only thing required by the author is to make there is a updated list of dependencies for the project.

You can divide between two package managers. System package managers and application-level package managers. System package managers are often the package manager used in your operating system. This depends on the Linux distribution, or if its macOS or windows. Neither macOS nor Windows was designed to be used with a package manager, but solution has been developed. Respectively homebrew and chocolatey.

Application-level package manager support packages for their own application. This is commonly used to manage dependencies for popular programming languages like, Python, PHP, NodeJS, Ruby and Perl. These are usually called by the system package

managed so they don't conflict with the different operating system.

5 Communication

Communication is important in the open-source world. There has to be a place for the developers to discuss features, the road ahead and how things should progress. Even with the rise of social coding platform, two things has largely stayed the same; Mailinglists and IRC.

Mailinglists is used a lot inn projects. This is where the announcements go, and general discussion of features. Patches can be sent here, help and guidance is also often here.

IRC is one of the oldest chatting services in the world. It was created inn 1988 by a finnish student. It has been one of the most important chatting platform through the years. However, it has declined inn more recent years. But developers and IT people still enter and actively discuss on these networks[16]. Larger projects use these channels to casual chitchat, discussions and meetings. The Tor project heavily use these channels to plan development.

Conferences is also an important part of the open-source ecosystem. Since large parts of the developers do not work together physically, conferences is where they meet and discuss. One of the largest is called FOSDEM[17], and currently hosts 8000 people from the open-source community inn a free today event. Here you can attend workshops, talk with contributors from other projects, and join sprints if you want to try and get your hands on some new projects.

6 Linux

Linux is the largest open-source project ever created. The project was started in 1991 by Linus Torvalds to create an open clone of the MINIX kernel, only available for educational use[18]. As Linus is only an operating system, it is supported by the GNU project,

providing the user tools to interact with the operating system. This has created some controversy around the naming of the operating system as Stallman with the Free Software Foundation claims it should be named GNU/Linux as linux alone is not in reality an operating system.

Linux has so far more then 3000 developers, and over 400 companies making it one of the most successful open-source projects ever. Linux is behind the largest market share of mobile phones, and web servers, driving the world wide web.

6.1 Development Method

Linux is developed by using a distributed version control system called git. The community discussion is centered around mailing lists and IRC channels, a distributed messaging protocol. Linux has a lot of maintainers and developers working on the project, 13% are volunteers, and rest is from companies like Google, IBM and Redhat. The release cycle lasts about 2 months, where the patches has a review period of 2 weeks. All patches are generated and sent to the mailing, and reviewed there[19].

Linux does not use a social coding platform for its code. There is a mirror available on github, but they do not accept changes there. They host their own git repository on their own website.

Because of the large codebase, linux has a lot of maintainers spread out over different subsystems of the kernel. One person could manage the file system drivers, while another would maintain the wireless drivers. During a normal 2 weeks period developers can receive as much as 490 patches, and the 3.8 release of linux had 6-7 changes per hour applied. The most recent release of linux, version 4.9, had the most changes ever. 16,216 in the 4.9 release cycle. Linux release a new version every month, this means that there is roughly 270 changes a day to the linux kernel at its current state[20].

6.2 Distributions

Linux and the GNU project itself alone is a boring thing. It would largely only give you a shell on a black screen with a cursor to type inn commands. It's the distributions of Linux that really makes it. A distribution is essentially Linux+GNU+More. They together create a graphical environment along with a package manager. Depending on the distribution used, different settings and default packages could be installed. Some distribution only give you the bare minimum needed for you to create your own tailored system, Arch Linux and Gentoo are good examples here. While others like Ubuntu, Debian or Linux Mint create a very complete feeling of an operating system.

They can provide a installation CD, guiding you through the process of installing the system, with a default set of configurations. This makes linux largely accessible for a wider audience. The minimal distributions also suits more experienced developers and power-user better.

6.2.1 Debian

Debian is one oldest distributions of linux. It was created by 1993 by Ian Murdoc, and the first stable release appeared in 1996. Today its one of the most widely used distribution of Linux.[21]

Debian largely inspired the works of the Open Source Initiative by Eric S. Raymond. They defined "Debian Free Software Guidelines" which is part of their social contract. This determines if a license is an Open Source license, and can be used inside the Debian project, The Open Source Initiative adopted these guidelines when they later defined the open-source license approvals.

Debian is largely built by people inn their free time. It is estimated that the work put into the project is worth around 8 billion US Dollars.

It powers home computers, server providers and even the international space station. NASA changed from Windows too Debian for their space station inn 2013. Largely because of stability and ease of management and updating[22].

7 VLC

VLC is a free and open-source video player. It supports a large number of codecs and medias to play from. It's multi-platform and supports Linux, Windows, macOS, Android and iOS among others. It is widely used for being free and supporting the most used video formats today.

It started out as way for the campus students to stream from satellites over a campus network, but today is a more generalized media playback service. People use the software today for media playback. It has been downloaded more then 2 billion times

7.1 Development

VLC uses a workflow very similar to the workflow of Linux. They are based around mailing lists, use an external bugtracker, in this case trac. And utilize git for its version control. The code is hosted on an external git server, and they do not relay on external services such as github. There is also a wiki website to collect developer information, and misc information about the project. It is released under the GPL license.

8 Tor

Tor is an anonymity network created and maintained by The Tor Project. The network consists of nodes hosted by anyone, worldwide and through cryptography keeps the origin of the connection to the destination a secret. It was originally developed to be used by the US armed forces, but since been open-sourced and improved upon. It is today widely used by journalists and outspoken activities to hide their identity[23].

Tor also allows people to host "Hidden Services", which is webpages only accessible from the Tor network. This enables the services to hide their IP, and the services to never know who accessed the websites.

Tor has gained a lot of controversy as people have been using the system for distribution of illegal items. Silk Road, an infamous drug dealing website was funded on the Tor network and gave the concept of "dark nett" to the public media.

8.1 Development

The development of tor is slightly different from what we have seen. They do host their own git server, with a large collection of projects that belong to contributors. The main discussion happens on mailing lists, but patches and proposed code changes are never submitted there.

Instead they utilize the trac bugtracking software for the submission of changes. This differs from Linux and VLC which we went through earlier. This choice makes it easier for people not following the wast mailinglists of the project, and it easier for new people to contribute. A lot of the discussion and the community effort is done over IRC, they have several channels dedicated for Tor users to talk inn. They also provide own developer channels for new people that want to contribute to one of the projects under tor.

9 Chromium

Chromium is a open-source webbrowser released by google. It is used as the development version to googles stable webbrowser, chrome. It is considered to be among the most secure webbrowser today, and there is a yearly competition to find bugs, or zero day exploits inn the software. The bounty is currently at 63,000 US Dollars[24].

9.1 Development

Chromium is written on a rolling release basis. That means every commit into the project is built and handed out to users. This is know as Continuous Integration, along with using Buildbot to build and create available version, they also use a Continuous Build

system. This enables the chromium to adapt quickly whenever there is a bug, or much needed feature available for the user.

Large parts of the chromium discussion is on their IRC channel, along with mailinglists. They also provide a self-made codereview tool for submitted patches from the community.

10 Summary

In this essay we have taken a look at important aspects of the open-source community, and the development of open-source software and the history behind it. Free and Open Source Software is crucial and important in todays technology driven world. The projects fueled by the work of volunteers and corporations alike is important for the growth of the internet.

References

[1] IDC Research. Smartphone os market share, 2016 q2. https://www.idc.com/prodserv/smartphone-os-market-share.jsp, 2016.

[2] Pingdom. 75% of top 10k websites served by open source software. http://royal.pingdom.com/2012/05/22/75-percent-top-10k-websites-served-by-open-source-software/, May 2012.

[3] Francois Fluckiger Tim Smith. Licensing the web. https://home.cern/topics/birth-web/licensing-web, December 2013.

[4] Free Software Foundation. About. http://www.fsf.org/about.

[5] Open Source Initative. History. https://opensource.org/history.

[6] Open Source Initative. Licenses standards. https://opensource.org/licenses.

[7] GNU Operating System. Various licenses and comments about them. https://www.gnu.org/licenses/license-list.html.

[8] Eric S. Raymond. The cathedral and the bazaar. http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/, September 2000.

[9] Github. Git. https://github.com/git/git.

[10] Github. About. https://github.com/about.

[11] Vincent Driessen. A successful git branching model. http://nvie.com/posts/a-successful-git-branching-model/, January 2010.

[12] Buildbot. Buildbot. http://buildbot.net.

[13] Travis. Travis. https://travis-ci.org.

[14] Jenkins. Jenkins. https://jenkins.io.

[15] Ian Murdock. How package management changed everything. http://ianmurdock.debian.net/index.html%3Fp=437.html, July 2007.

[16] Google Summer of Code. Irc: Internet relay chat. https://developers.google.com/open-source/gsoc/resources/irc, February 2016.

[17] FOSDEM. Fosdem. https://fosdem.org/2017/.

[18] Wikipedia. History of linux. https://en.wikipedia.org/wiki/History_of_Linux.

[19] Greg Kroah-Hartman. I don't want your code. https://github.com/gregkh/presentation-linux-maintainer/blob/master/maintainer.pdf, March 2013.

[20] Linux Torvalds. Linux 4.9. https://lwn.net/Articles/708766/, March 2013.

[21] Debian. A brief history of debian. https://www.debian.org/doc/manuals/project-history/ch-intro.en.html#s1.1, April 2015.

[22] The Linux Foundation. Linux foundation training prepares the international space station for linux migration. https://training.linuxfoundation.org/why-our-linux-training/training-reviews/linux-foundation-training-prepares-the-international-space-station-for-

[23] The Tor Project. About. https://www.torproject.org/about/overview.html.en.

[24] Zero Day Initiative. Pwn2own contest rules 2016. http://zerodayinitiative.com/Pwn2Own2016Rules.html, 2016.