Source code distribution in open source software communities

Topi Konttavaara, Teemu Moisanen, Aleksi Pöyhtäri

Abstract

There exist many ways to distribute and share the source code and binaries within communities in open source software development (OSSD). We assert that some of these ways must be better than others to share information with anyone who wants to take part in the projects. Everyone in open source communities want to share work in an easy way. This research wants to examine some of the ways that are used to share source code and binaries in the open source world. We have also included some comparison of the different methods that are used to share source code and binary formats of open source software. After the comparison we hope to have a better understanding of the various methods and also their advantages and disadvantages.

1. Introduction

Open source projects have multiple ways of distributing source code and binaries. One of these ways is through VCS(Version Control System). Our purpose is to find out, what open source software developers need to consider when they decide to distribute their software as open source. Version control system services are a popular way of providing your source code for everyone who wishes to see it or contribute to the project. Most open source projects that include more than one contributor use a version control system in their project. Version control is a handy way to keep track of contributions to the project.

Open source code has specific rules for how long source code needs to be available and exactly how it needs to be available. Also there are open source licenses that tell the developer for how long the developer needs to support their software..These rules are enforced in a project that uses a version control system. Even if development has ceased, the project will be available via the version control system so that people can obtain the source code and contribute, should they choose to.

The research question we aim to answer is: How open source source code and binaries are typically distributed. And what type of different methods are available.

One of the foundations of open source software is that the source code is publicly available for everyone who wants to obtain it. We aim to explore how source code is typically distributed. Most developers know of sources such as Github and SourceForge. Both of them are very popular but are they the most typical way of distributing open source software? VCS(Version control system) is a tool for developers, but they are also used to conveniently make source code available for those who wish to take part in the development process.

Previous research suggests that almost all open source projects use some kind of version control system. However it is not the only way to distribute source code. Our research could not conclude definitely how source code is typically distributed. We can however say with reasonable certainty what has been observed so far.

2. What is source code distribution

2.1 History of open source software distribution

“The history of open source is closely tied to the history of the hacker culture, since it is largely hackers who have sustained this movement. (Bretthauer 2001). Hacker as a word means a passionate hobbyist and a skilled programmer. Not to describe a person who uses their skills to break systems.

Most open source software is distributed through the internet. Internet is the same network that was originally designed to exchange information between universities. Now the internet has spread throughout the world and is used as a platform to spread software to name one of the few uses of internet.

First version control systems were made as early as 1970's. The main idea was that projects are easier to managing source code and multiple developers could work simultaneously with same code base. After the first version control systems appeared many parties have shown great interest in data, which is made in process. This data may contain things like evolution of data, snapshots from specific point of time and ways to recover/restore the past. Collaboration and groupware systems live from things like these.(Koc & Tansel, 2011)

First system that represented some kind of version control was Source Code Control System (SCCS), which was developed by Marc J. Rochkind at Bell laboratories in 1972. It was a very basic way to handle files. This contained some command for the OS/MVT and later for UNIX that manipulated files. SCCS handles every source code that is a file. This system generated two main attributes, which were inserted lines and deleted lines. These are so called deltas, which are born even if a developer changes one character from the line or moves line from one place to another. Rochkind the had inspiration for version control when he thought of basic challenges faced by software development life cycles. When something goes wrong the first question is "What changed?" (Koc & Tansel, 2011)

Walter Tichy created Revision Control System in 1982 (RCS), which improved the capabilities of diff and patch by automatically keeping track of the change history. Although he missed some crucial needs when he developed RCS. For example RCS uses locking mechanism in development style that blocks other core developers from modifying the code simultaneously. Further, RCS does not support network development. This essentially means that developers must share a developing machine. (Wu & Lin, 2001). Compared to SCCS, RCS used reverse delta method for the storage. This essentially means that the newest version is stored on the central server. There is instructions how one developer can go back to previous working version. That's why developer must do it manually.This new way of storing crucial data is making checking and reverting tasks more simpler and faster. (Koc & Tansel, 2011)

In 1986, Dick Grune created the Concurrent Versions Systems (CVS), which uses RCS's attributes and methods and makes them easier to use. Brian Berliner then rewrote CVS using C. (Wu & Lin, 2001). After C compiler was used to RCS and if only one file was under work, this new CVS could have be seen as a wrapper for the RCS. The biggest advantage for the CVS comes when the project consists of multiple files, because it allows the users see the whole project as a single file. This makes version control for multiple files much easier. For example a single command can manipulate the whole directory. CVS also introduced a client/server model for version control. This allowed multiple developers to work remotely from anywhere they wanted. Remotely working means that every developer can be use multiple devices to develop the code. The only need is to connect one's device to central server to get most recent version of the code. Version history of the project was also stored on a central server while developers worked on their own machines. When the developers worked on their own machines, they didn't have to lock files what they were working on. (Koc & Tansel, 2011)

After the CVS, CollabNet developed the next generation of the version control systems. It was called subversions. CollabNet advertised their product as “better CVS”. Their biggest outcome was developing system, which were using atomic commits. This kind of commits means that, if the connection disappears or something unexpected happens and commit fails. Then the central server has option to reject all the changes what has come in commit process. This kind of attribute prevents corruption between versions and between developers. The main reason for this method was that all the data that all the developers work on, lays on the central server. The another big step in this development process was renaming and moving files inside the project. This was also the beginning refactoring the code (Koc & Tansel, 2011).

Latest is Distributed Version Control Systems (DVCS). Basically this is the newest and will be futures way of distributing projects. This gives developer an opportunity to clone whole project for one’s local machine and start developing it. DVCS forgets the centralization and gives freedom of working without network connection. Advantage in this way is the more faster way of working and gives developer of change drafting before committing to whole project.(Koc & Tansel, 2011).

Three generations of version control systems(Sink, 2011)

Generation	Networking	Operations	Concurrency	Examples
First	None	One file at at time	Locks	RCS, SCCS
Second	Centralized	Multi file	Merge before commit	CVS, Subversion
Third	Distributed	Changesets	Commit before merge	Bazaar, Git, Mercurial

2.2 Source code distribution today

The transition from CVS to DVCS is very interesting. Larger projects have huge codebases so the transition is very hard. Still some companies do the transition to get some benefits for their projects. There are five different benefits from this transition, access for all the developers, atomic changes, automatic merging, experimental changes and support for the disconnected operation. Access to all developers means that all can get more easily commit access for the project. Atomic changes prevent more efficiently corruption in projects. Automatic merges on the other hand help projects maintain sufficient information in long living branches. This helps developers keep their branch up-to-date and releases some burden from the committers. The next benefit is the experimental changes, which basically means that every developer can work on their own experiments without breaking anything. After the change is seen as good it can be submitted for all. The last change is disconnected operation. This benefit can be seen as offline working. CVS requires the connection with server and DVCS doesn’t. This helps some developers to work on their and create a snapshot from their work on other developers. (Alwis, Silito, 2009).

Of course there is some challenges and changes when doing this kind of transition. One of the challenges is the change in a team's development process. Some teams got more integrated tools and processes. Also survey from Alwis and Silito brought up that some teams were fearing that their metadata from old VCS comes to haunt in the next one. Teams from NetBSD were telling that they want more meaningful tag to their commits. When they read their e-mails they know from the tag where the revision is done. Challenge here is that tags may not always be suitable for DVCS. Also they brought up some old law-case where the some certain source code file had been expunged from NetBSD repository. To do this in DVCS is basically impossible. The last challenge is that DVCS is new managing system for repositories for the developers. This makes marketing of the transition harder, because making people understand the change and making some tutorials are need to overcome the learning curve. (Alwis, Silito, 2009).

Today, projects developed for the high end commercial market uses second generation version control systems, among the most popular the Apache Subversion tool. Community developed collaborative projects however have embraced the third generation tools, the distributed version control systems. While there are many options, the three most popular DVCS systems currently used are Bazaar, Git, Mercurial (Sink, 2011).

Distributed Version Control Systems operate on a peer-to-peer model. They operate in a manner described best as distributed or decentralized. DVCS allows the repository to exist in more than one place. Each instance of the repository contains all of the project's history and metadata and every developer has his own private repository instance. Instead of a single central server, the repositories are often synced with each other. In practice, most projects operate a central server, even with DVCS, however this is not a requirement of the system but a decision of the project team. (O'Sullivan, 2009)(Sink, 2011)

While most open source projects have multiple ways to obtain the source code, many of them today offer a source code repository as the recommended source for the latest version. Many of these repositories are free and open cloud based services. These services offer the users free repositories and eliminate the need for the projects themselves to host a version control system server. Some of the most popular services are GitHub, SourceForge and BitBucket. However SourceForge has had almost all of it's projects moved to other service platforms because of their policy of acclaiming unmaintained project and the use of adware in the projects install files.

Alternatively, many projects offer the source code as a downloadable package, often packaged and compressed in for instance zip or tar format. Several projects offer these packages through own web servers, free services that offer download mirrors for open source projects, or one of the many content delivery services available through cloud based hosting servers. The main difference between offering the source code in a repository and a packaged format, is how up to date the code is depending on recent changes. The repositories often offer a current snapshot of the development branches of the project, whereas the packages distribution offers a version as of the last release of the software.

3. Distribution methods

3.1 Overview of various distribution methods

Given that a project's co-developers may be scattered across the globe, they must agree on a version control system to avoid development chaos. (Wu & Lin 2001.) Most open source projects use some form of version control system nowadays. One of the most popular version control platforms is Github. Github offers free repositories for users and also private ones for a fee. Open source projects make use of the free repositories as they can be viewed by everyone and it is a free and easy way to publish your project.

To facilitate concurrent software development and fast, controlled evolution, most open source projects implement some form of configuration management. They do this by using CVS (the Concurrent Versions System), other tools, or even an ad hoc solution using Web-based support. The need for version control stems from the global distribution of developers and cultures. Also version control systems provide security in case there is an accident or someone uploads defects to the code. By using VCS these blunders can be removed easily from the build and the project can be reset. (Gacek & Arief, 2004)

The previous research to the subject leads us to the conclusion that most project implement some sort of CVS as their primary distribution platform for source code. Most projects also have websites, discussion forums and email lists for coordinating their projects. The OSS has sprung from the invention of the internet and has spread throughout the world. We can also note that as the internet has spread, so has the open source software way of life, more and more people are joining up to develop software for everyone to use and contribute to, should they wish to do so.

A popular method of distributing the source code in open source projects is still through a packaged and compressed package, usually containing the source code version as of a major release. This allows the software's users to obtain a relatively stable and tested version of the code. Compared to version control systems, this method however is slow in releasing the newest fixes and patches to possible bugs.

3.2 Advantages and disadvantages of select methods

Using version control systems, the developers can easily publish their changes and avoid the chaos that ensues when people from all around the world work on the same project. Of course proper coordination between project members could help to mitigate the chaos but using version control makes this process much more lightweight. Version control systems also enable backtracking if something goes wrong with the project. Probable causes for backtracking the project could be defective code or some sort of disaster in the code itself that open security holes.

The codebase is the whole collection of source code used to build an OSS system. The project's codebase is typically stored in a source control repository. Source control systems used in the project under study included CVS, Subversion and Git. In general, developers modify a local copy of the project source code. This local copy is called a working copy, and resides on a developer's computer. When they want to make modifications to the project codebase, developers need to describe the changes in the form of patches(Sethanandha, Massey & Jones 2010).

A clear advantage of the using version control systems in open source projects is that the developers can get their own working copy and after they have tested their own solutions they can apply their patch to the codebase for others to use. The disadvantage of version control is that in the event of merge-clash or faulty code there can be some serious consequences for the project. This can mitigated through branching. Of course in the event of a faulty code submission, the project can always be reverted to the last working iteration. This is one of advantages of using version control systems in a project.

When comparing CVS and SVN to Distributed Version Control (DVCS) in version control CVS and SVN one main disadvantage is that you must be core developer when doing basic tasks to project. These task might be like reverting changes to a previous state, creating or merging branches, publishing and thing like that. Mainly this kind of weakness affects projects participation and authorship for the new developers. This DVCS systems includes the most used version control systems like for example Mercurial, Git and Bazaar (Rodriguez-Bustos & Aponte, 2012.)

One of the most used version control systems is Git. Git has made making small changes very easy. Unlike SVN where the project itself has a version history, Git uses version history for every file. This enables simple changes for one file and to merge it with the project. This also is a benefit in the case of an error that results in a bug in the codebase.

From technical point DVCS wins again. Developers can get their own copy and work their changes without being connected to server or any other client, in other words offline. Basically this means that every operation what is done offline is faster. Also the changes between versions isn't only a whole project's version history, every file has its own version history. Rodriguez and Aponte tells that this advantage makes branching and merging easier (Rodriguez & Aponte,2012).

Also there is structure view when comparing different ways to distribute OSS projects. In this context structure means structure of the team. DVCS gives an opportunity to create, implement or switch between different workflow models, because classic centralized model is no longer the only option. This basically means that every team can now easier difene intermediate roles for everyone. Multiple models to work on gives chance to record every change and contribution to project. This type of working gives any project big advantage on maintaining the projects.(Rodriguez & Aponte,2012.)

If a project is large enough, feature branching is a huge benefit. Branching has proven very useful when developing new features, because creating a branch enables the user to isolate their feature and all it’s effects on the codebase. Of course working offline can achieve the same thing, but using branches, the code can be uploaded to the repository to avoid possible losses of due to accidents. Also using branches enables other developers to participate in the development in an isolated working copy of the codebase.

Research is also one aspect of comparison. Nowadays versions control systems record so much data that it might help researchers to acknowledge how DVCS affect processes, products and people around the projects. There are two ways that might help researchers. The first is that nowaday branches are smaller but contains more crucial information about contributions and workflows. The second advantage is also related to reduced sizes. This way extracting data and repositories cloning are much faster than before. (Rodriguez & Aponte,2012.)

4. Conclusions

Most open source software projects offer multiple methods for users and developers for obtaining the source code of the project. Stable releases are offered through compressed source code packages, often through the website or some other possible download mirrors, and the newest developments and fixes to the code through some version control systems often hosted on free services online.

The most popular way of distributing source code of open source projects however is some form of version control system. Two of the most popular services currently used are Github and SourceForge. However lately SourceForge has become more menial for both developers and users and many of the projects previously hosted on it have moved on to Github. The most used forms of version control systems are SVN, GIT and Mercurial. Github itself uses GIT as its underlying system.

The distribution of open source software benefits greatly from the use of version control. The developers can perform analysis on the code before it is merged in to the codebase and they have a way to revert the codebase in case of an unexpected bug. The use of version control system also gives the developers the freedom to commit their contributions from anywhere around the world. New changes to the software can be released very quickly to users and other developers.

Open source software has brought people who are interested in development as a hobby together as well as a full time job for some. Open source software depends greatly on availability of source code for all who wish to contribute or modify it. This would not be possible without version control systems and the internet. Version control system repositories online have made sharing source code and relevant information of the project easy and efficient. As is the nature of open source software the source code is available. The spreading of the internet has brought new developers to the open source software community from all over the world where the internet is readily available, that means the whole world right now. While open source software has been around even before internet became a commodity, the internet has enhanced it tremendously.

The limitations of this research is that most studies in this field cover a very wide area. Many studies do not specifically consider distribution of OSS. However many of the studies we have taken into this paper have covered a wider area than what we originally intended. However all the papers we perused gave us the material to make our conclusion into how OSS is typically distributed. At this point, we can with certain safety say that OSS is being typically distributed through VCS like github or sourceforge. We noticed that while research has been done in open source software the research has been more of a general in nature or the research has focused on the open source communities and process.

More research is needed in the field of source code distribution methods to say with absolute certainty what is the most common way of distributing open source software. However our research suggests that a common thing for most of open source project that include more than one person is using a version control system. We suggest that more research be conducted in this area of open source software projects. Some researchers have done an excellent job of pointing us in to the direction of the answer but still we cannot say that we are certain. We can however make the assertion that most projects do in fact use version control systems in some form.

References

Open Source Initiative. Frequently Asked Questions. Retrieved from http://opensource.org/faq

Raymond, E. (1999). *The Cathedral and the Bazaar.* O'Reilly Media

Gacek, C., & Arief, B. (2004). The many meanings of open source. *Software, IEEE*, *21*(1), 34-40.

Srinarayan Sharma, Vijayan Sugumaran & Balaji Rajagopalan (2002). A framework for creating hybrid-open source software communities

W.Scacchi (2002). Understanding the requirements for developing open source software systems

Bretthauer, D. (2001). Open source software: A history.

Ming-Wei Wu & Ying-Dar Lin (2001), Open Source Software Development: An Overview.

Sink, E.. (2011). Version Control by Example.

O'Sullivan, B. (2009). Making Sense of Revision-control Systems.

Sethanandha, B. D., Massey, B., & Jones, W. (2010, July). Managing open source contributions for software project sustainability. In *Technology Management for Global Economic Growth (PICMET), 2010 Proceedings of PICMET'10:* (pp. 1-9). IEEE.

Christian Rodriguez-Bustos & Jairo Aponte (2012), H*ow Distributed Version Control Systems impact open source software projects.*

Ali Koc & Abdullah Uz Tansel (2011), A Survey of Version Control Systems.

Brian de Alwis & Jonathan Silito (2009)*, Why Are Software Projects Moving From Centralized to Decentralized Version Control Systems?*