Open Source Software

EBU5304 Software Engineering
2018/19

Dr Matthew Huntbach
matthew.huntbach@qmul.ac.uk

Free Software

- The word “free” in English has two meanings:
  - Available without requirement of payment
  - Able to do whatever you want
- Conventional software is produced to meet a customer need, either it is produced for an individual customer who has asked for it, or it is built for open sale to customers who see it advertised and purchase it
- Those who produce conventional software expect to get paid for it
  - In order to protect their income, they will impose restrictions on how it can be used, which will usually mean it cannot be passed on to anyone else, and that it cannot be modified in unauthorised ways
  - As part of this protection, usually the source code is not made available, what is distributed is the machine executable code produced by compilers, which is difficult for humans to understand and modify
- The phrase “open source” derives from the idea of the source code being publicly available, but the phrase “Open Source Software”(OSS) is usually used to mean software which is free of charge, and free of legal restrictions on how it can be used

Why work for free?

- Many people enjoy computer programming so much, they would do it as a hobby even if it wasn't their job
- Early OSS was sometimes hobby code
- However, many complex software products in day-to-day use are OSS, it is not just a small-scale thing
- Many of the early developers of OSS were people who had strong beliefs about personal freedom, and saw distributing their free software as a way of giving people freedom instead of being dependent on commercial companies
- However, the success of OSS products has led many to support its use just because they see it as providing good reliable products
- OSS is particularly common for software systems which are used to support software development, such as the Linux operating system and the Eclipse development environment, the Spring application framework
- In some of these cases it is software produced for personal use that was spread more widely because it was found to be effective

OSS Business Strategies

- Some who produce OSS may be motivated entirely through altruistic reasons
- Some may like the fame that comes from producing a successful and widely used product, even if it doesn't come with a fortune
- However, others may see OSS as a "loss leader" (in business terms, a product sold at a low price to stimulate sales of more profitable goods or services)
- There are companies which make profit from installation of OSS products, from offering training in the use of OSS products, and from providing consultancy advice to businesses about their use.
- One example is Red Hat, which now has an annual revenue of over one billion US dollars. See: http://www.redhat.com/en/open-source
- The Java programming language has many of the aspects of OSS, although it does not fully fit the definition because modification and redistribution of Java's support code is banned by its developers (Sun Microsystems, now owned by Oracle Corporation)
- One of the reasons for the success of Java was because it was distributed free of charge

OSS Development Methodology

- Software produced by volunteers cannot be produced under the forms of management used for commercial software
- Large scale OSS systems have to be produced by teams, like any other large scale software products
- There needs to be an overall co-ordinator, but assignment of individual tasks is done through volunteering
- A large scale OSS system will be developed by people working across the world, communicating electronically
- It has some of the aspects of Agile software development, but the Agile methodology emphasises close personal interaction
- With OSS systems there is typically no developer-customer distinction, the developers are also customers as they will make use of the system themselves
- One of the pioneers of OSS, Eric S. Raymond, described the distinction between development of software by a central business organisation and by a community of volunteers as like that between “The Cathedral and the Bazaar”: http://catb.org/~esr/writings

The Cathedral and the Bazaar

- A cathedral is a large building built to a single plan under close control, which typically takes a long time to build
- A bazaar is an informal collection of stalls, which can change quickly and seems to have no central control, relying on stallholders co-operating, but each working to their own goals
- Commercial software, and early OSS systems were developed like a cathedral
- Development as a bazaar means customers act as testers and debuggers, good quality stalls stay, bad quality stalls go through lack of business or are improved through customer suggestion
- A bazaar may have a co-ordinator who manages the recruitment of stall-holders, but does not have a strong overall plan and leaves the stall-holders free to work as they want

Many Eyeballs Tame Complexity

- In OSS systems, the developers are typically also users of the software
- Users who are also developers can report bugs if they find them and suggest corrections based on being able to see the source code
- The result is that there are a large number of users who also act as testers
- As the software is source code rather than compiled machine executable code, it can easily be updated to correct a bug, it does not have to wait until a formal release of a new version
- Typically, a large OSS system will be divided into many small parts, with correction of a bug in one part not having an impact on development of other parts

Open Source and Business Risk

- If you run a company which needs a software product, you could develop your own, but this would be expensive to do and to maintain as your requirements change
- You could purchase a product from a closed-source supplier, but then you are tied to that supplier: you are forced to carry on making use of them and paying what they ask for further development, because only they have knowledge and control of the code
- If you use an OSS product, you have the source code, and there is likely to be several service companies who will help adapt it to your needs
- So using an OSS product means there is no risk that a company supplying the software to you goes out of business, or makes unreasonable charges because it knows you have no choice but to carry on using it

Software Freedom

- The strongest form of “free software” insists on the following four freedoms:
  - The freedom to run the program, for any purpose
  - The freedom to study how the program works, and change it so it does your computing as you wish. Access to the source code is a precondition for this.
  - The freedom to redistribute copies so you can help your neighbour.
  - The freedom to distribute copies of your modified versions to others. By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this.
- These were developed by Richard Stallman founder of the GNU project, which was set up to produce an OSS version of the operating system Unix
- The work co-ordinated by Stallman was combined with the work on a Unix kernel initiated by Linus Torvald to produce the system which is generally called Linux, but Stallman prefers to call GNU/Linux
- Stallman is know for his strong personal views, and has established a distinction between Free Software meaning with all the freedoms above, and “Open Source Software” which may have more limited freedoms

Copyright

- Copyright is a legal term meaning legislation which gives protection to creators of an original work, originally it referred to writing, and was developed when printing made copying written work easy to do
- Now copyright law covers sound and pictures, and also software
- The idea is that only the producers of the original work have the full right to produce copies, and create new work based on it, but they may grant permission for others to make copies of the work and adapt it
- Permission to make copies and adapt work will generally be made with the imposition of obligations on those doing it towards the original author
- Such obligations would usually include making a payment to the original author, but could also include restrictions on the form of the copy and and any adaption
- Details of copyright law vary from country to country, but most countries have signed to international agreements which establish standards for copyright law
- The idea is that people and organisations will be more willing to produce new work, if they are guaranteed to keep profits made from it

Copyleft

- The supporters of Free Software have developed a concept they call copyleft to mean using the copyright laws in a way that guarantees the freedom of others to copy and use their work rather than stopping unauthorised or unpaid copying
- It involves a free software license, which is associated with free software, use of that software is granted only to those who agree to the terms of the free software license
- The terms will generally state that users of the software may pass it on to others only if when it is passed on, those it is passed to also agree to the terms of the same license
- The license will include the requirement that the source code is made available, and also state the extent to which the code may be modified and how the modifications should be acknowledged
- There are many variations on this form of license, the most commonly used form is the GNU General Public License, developed by Richard Stallman
- If the terms of the license do not insist that copies and adaptions are passed on with the same license, it is not copyleft

Free Software and Open Source Software

- Richard Stallman makes a distinction between the practical reasons for using Open Source Software and the principles that led him to develop it, and is critical of those who think of it only in terms of its practical benefits
- He uses the term “Free Software” for software that meets his principles
- From the GNU Manifesto http://www.gnu.org/gnu/manifesto.html:

  “I consider that the Golden Rule requires that if I like a program I must share it with other people who like it. Software sellers want to divide the users and conquer them, making each user agree not to share with others. I refuse to break solidarity with other users in this way. I cannot in good conscience sign a nondisclosure agreement or a software license agreement.”
- From http://www.gnu.org/philosophy/open-source-misses-the-point.html:

  “Some of the supporters of open source considered the term a ‘marketing campaign for free software,’ which would appeal to business executives by highlighting the software's practical benefits, … The term ‘open source’ quickly became associated with ideas and arguments based only on practical values, such as making or having powerful, reliable software. Most of the supporters of open source have come to it since then, and they make the same association.”

Hacker Culture

- Richard Stallman and Eric Raymond regard themselves as “hackers” as the term was used in the 1970s and 1980s
- It meant someone who loves to program and often also has other features, such as enjoyment of tricks with language, dislike of authority, unconventional personal behaviour
- It was associated in particular with a group of people at the MIT AI lab, and other top university computer labs
- The word “hacker” is now often used to mean someone who deliberately tries to break the security of computer systems, but the original hackers call such people “crackers” and do not agree with them because their focus is not on programming
- From Eric Raymond’s “How to become a hacker” http://catb.org/~esr/faqs/hacker-howto.html:
  1. The world is full of problems waiting to be solved
  2. No problem should ever have to be solved twice
  3. Boredom and drudgery are evil
  4. Freedom is good
  5. Attitude is no substitute for confidence

e-voting

- Richard Stallman is a prominent opponent of the use of computers to assist in the casting or counting of votes in elections (e-voting), he advocates keeping to paper ballots and manual counting
- Many other software experts agree with this position
- This often comes as a surprise to others, who assume software experts would be in support of “modernising” voting procedures by introducing computer technology
- The reason why software experts tend to oppose e-voting is that they are particularly aware of how software can perform incorrectly, either through unintentional error or through deliberate manipulation by those who wish to gain unfair or illegal advantage
- This is an example of how new technology is not necessarily an improvement on established old-fashioned techniques

Requirements for Voting Technology

There are particularly strong security and accuracy requirements with any system for voting, it must ensure:

- That each person who is entitled to vote may vote at most once
- That no-one who is entitled to vote and chooses to do so is blocked from doing so
- That those who vote are presented with the full range of choices
- That the choice of each person who votes is correctly counted and cannot be changed by anyone else
- That no-one who does not have the right to vote can vote, either by being given that right incorrectly, or by pretending to be someone who has voting rights
- That the total votes are counted correctly, and cannot be changed
- In most cases, that no-one else can identify what choice any individual voters made

The Diebold Voting Machine Controversy

- Diebold was a company that manufactured machines for direct recording electronic (DRE) voting, widely used in the USA
- In a DRE system, voters are issued with a card for their individual use, presenting the card to a machine causes the voting choices to be displayed, once the voter has made a choice and confirmed it, the choice is recorded and the card invalidated so it cannot be used again
- An official starts the machine when the election is started, stops it when it has finished, and the data recorded by the machine is sent to a central server to calculate the final election result (there are generally many machines for an individual election)
- The source code for Diebold's voting machines was private to the Diebold company, but a researcher found a copy of the code that had been mistakenly placed on a public website
- Analysis of the software showed there were many ways in which the security and correctness of the voting system could be broken
- This was written up in a report by a team led by Johns Hopkins University Computer Science professor, Avi Rubin

The Rubin Report

- The report written by Avi Rubin and his team was published as a paper in the *IEEE Symposium on Security and Privacy 2004*
- Among the problems noted in the report were:
  - The system did not use cryptography, so voting cards could easily be forged
  - Voting cards could be programmed to allow a single voter to vote more than once
  - The administrator cards used by officials to start and stop the machine could easily be forged
  - The software recording the votes could be accessed and changed
  - The software recording the votes could be accessed to find out what choices were made by individual voters
  - The software displaying the voting choices could be accessed to change the choices or link the display to the wrong choice
  - An unauthorised voting machine could be linked to the server which collects the voting information
  - The code was of a poor standard, so could easily go wrong without that being detected
  - The code was poorly structured and commented, making it difficult to be sure it would work correctly

E-voting in the Australian Capital Territory

- The country of Australia is divided into several states or territories, which have their own Parliament which deals with most internal issues
- The capital city of Australia, Canberra, is a separate territory with its own Parliament for city issues, it is called the Australian Capital Territory (ACT)
- In 2001 the ACT introduced e-voting for elections for elections to its Parliament
- Although a private Australian company designed the system used, it was based on specifications set by independent elections officials, who posted the code on the Internet for all to see and evaluate
- A separate verification and validation company was hired to audit the code
- Members of the public were free to check the software themselves, and identified some bugs which were corrected
- There was confidence in the system because it was Open Source
- Note, Richard Stallman does not agree that OSS solves concerns about voting machines, arguing we cannot be sure the software that is displayed is the software that is actually used

Core and Community

- In successful Open Source projects, there will generally be a small core of people (10-15) who are active in controlling the code base, and deciding on new functionality
- A larger group of people will have some involvement in correcting defects in the software
- A larger group than this will be actively involved in reporting problems with the software
- Volunteers may submit code to implement proposed new features, the project may benefit from picking the best from several implementations
- This system works best when, as with GNU/Linux, the overall product naturally divides into many semi-independent parts
- A fork is when there is a split, and a separate group starts maintaining a separate version of the product
- In most cases, developers will prefer to contribute to an established product rather than fork it and develop their own new version
- So enabling volunteers to contribute to an OSS product helps maintain one standard product rather than seeing it split into many versions

Open Source Governance

- Proprietary software is owned by the organisation which produces it, and controlled by a business management structure
- Open Source software development generally has a control structure which means an identifiable official version of the product exists
- In some cases, this is a “benevolent dictatorship”, meaning the person who first proposed the project always retains the final say, examples include Linus Torvalds with Linux and Guido van Rossum with the Python programming language
- In others, there may be a committee structure involving some form of election, an example is the Apache Software Foundation, which is responsible for the Tomcat Java servlet system, the Struts Java web application framework, and many other projects
- The mobile device operating system Android is open-source, but controlled by the private company Google
- The OSS nature of Android makes it easy to write applications for it, and has led to its widespread adoption, but Google have kept control by working internally on each new version, publishing the source code only when the version is released

The Android Open Source Project (AOSP) People and Roles

From http://source.android.com/source/roles.html:

- A Contributor is anyone making contributions to AOSP source code, including Google employees and external developers
- A Developer is an engineer writing applications that run on Android devices
- A Verifier is responsible for testing change requests
- An Approver decides whether to include or exclude a proposed change
- A Project leads oversees the engineering for individual Android projects and is “typically employed by Google”
- Verifiers and approvers are chosen by Project Leads “after demonstrating their skills by submitting a significant amount of high quality code to the project”
- Users are invited to report bugs, see:
  http://source.android.com/source/report-bugs.html
- Users are invited to submit patches, see:
  http://source.android.com/source/life-of-a-patch.html

How open source is Android?

- The analyst firm Vision Mobile have developed a way of measuring the openness of OSS projects, and claim Android shows the least openness of the major OSS projects they investigated, see: https://upload.wikimedia.org/wikipedia/commons/5/5f/VisionMobile_Open_Governance_Index_report.pdf. The key aspect of their claim is their suggestion that an open governance model is as important to a project being truly open as the licensing terms:
  - Licensing gives rights to use, copy and modify the source
  - Governance is about how open are the processes which decide on the future development of the project
- Vision Mobile claims that Android is closely controlled by Google, which gives priority access to specific developer groups and organisations, and does not make public its internal decision-making processes
- Google does not provide public information regarding meetings held and decisions made on the development of Android
- Google owns the Android trademark, and will not permit it to be used for devices which use Android code but do not pass a set of tests, the Compatibility Test Suite (CTS) which it controls

Open Source Software: Summary

- Open Source Software (OSS), has developed a long way from the time when it was seen as a few eccentrics putting together code as a hobby
- Most standard software development tools are now OSS, and OSS is moving into other areas
- The key to its success is that the community of users of the code also provides a community of testers and debuggers
- OSS develops rapidly in requirement to need because anyone who wants to improve it can do so
- OSS products avoid the risk of control of code that is vital for a business being in another business's hands
- OSS products maintain a single identity because it is in the interests of their users and developers to contribute to one standard product rather than "fork"
- OSS products maintain their OSS nature through a use of copyright legislation to enforce free distribution rather than prevent it
- Control of the development of OSS products ("governance") may now be with companies operating on a commercial basis rather than individuals whose motivation is enjoyment of software and/or personal freedom