Open Source

Software Architecture, Process, and Management

Open source projects once seen as idealistic playgrounds are now producing some of the software that we depend upon. The rise and popularisation of more lightweight development methodologies can arguably be attributed to the rise of open source software, which necessarily have little in common with heavyweight development methodologies. Open source projects provide the evidence that gradual evolution of a product can work. The causal relationship is still hotly debated.

Economies of Scale

- Our whole economy is mostly based on the economy of scale
- This first occurred during our ancestors' phase as hunter-gatherers
- Humans need plant matter to survive but meat gave our ancestors enough energy to power their massive brains
- It's more efficient for two people to cooperate in this endeavour
- Because, gathering for two isn't twice as difficult as gathering for one, and similarly hunting for two isn't twice as difficult as hunting for one

Economies of Scale and Services

- This is true even if it cannot be produced in bulk, for example a service
- A plumber, artist, doctor etc are all people who have learnt to do something and are then selling that expertise

Economies of Scale

- Large scale production of tangible goods works because even if you tell the consumer how to build the product, they cannot trivially copy it, and so additional customers have incentive to purchase from the original manufacturer
- Note though, that it doesn't prevent someone else from working out how to mass produce a similar competing product
  - This is the traditional motive behind patents, to provide incentive for people to innovate, since their innovation can be protected from competition

Economies of Scale

- There is a continuum of products from those that are difficult to re-use to those whose benefits can be usefully transferred to additional customers without recourse to the original manufacturer
  - Ovens, cars, etc.
  - Power sander, tennis racquet, etc.
  - books, DVDs, etc.

Traditional Commercial Software Development

- Producing consumer-oriented software is often done in much the same way as for tangible mass-produced products, such as furniture or cars. E.g., a company:
  - Designs the product
  - Produces copies of it
  - Provides a copy to any consumer willing to pay for it
- Most software products with widespread name recognition, e.g. from Microsoft or Adobe, fit into this model

Intangible Goods Revenue Models

- Note that this is true of other intangible goods as well such, movies, literature and music
- Broadcast television had a similar problem that creating one copy essentially made it available to everyone
- This has been solved in three ways, the licence fee, advertising revenue, and technological solution allowing pay-per-view

Intangible Goods Revenue Models

- Advertising is a good way to obtain revenue, however it only works for popular software/programmes etc
- It also means that what gets produced is dictated by advertisers rather than consumers
- It is more difficult to generate revenue for important but not so popular material
  - The licence fee has to some extent helped with that for television and radio, for example the shipping forecast

Modding Software

- Modding software is much easier if you have the source code
- If you have the source code, you can make multiple copies of the software **and** your modification
- This is a tremendous opportunity, but it also means the original company is less willing to give their software in source code format
- This doesn't make a lot of sense, since you have to artificially prevent the customer from copying it anyway
- But publishing the source code may make it easy for a customer or competitor to disguise a copy and break your no-copy-licence

DAYZ

- DAYZ is a very popular multi-player zombie survival game
- It is actually a “mod” of a tactical shooter “ARMA 2”
- It is more popular than ARMA 2
- When ARMA 2 was released on Steam, many thousands of people were believed to have bought ARMA 2 with no interest in the game but simply to play DAYZ
- So it can be in the original company’s best interest to allow modifications to their creations
- However, people wouldn’t have bought ARMA 2 if it were possible to distribute the modded DAYZ game separately

Richard Stallman

- In 1980, Stallman and some other hackers at the MIT AI Lab were refused access to the source code for the software of a newly installed laser printer, the Xerox 9700
- Stallman had modified the software for the Lab's previous laser printer (the XGP, Xerographic Printer), so it electronically messaged a user when the person's job was printed, and would message all logged-in users waiting for print jobs if the printer was jammed
- Not being able to add these features to the new printer was a major inconvenience, as the printer was on a different floor from most of the users
- This experience convinced Stallman of people's need to be free to modify the software they use

Free Software Foundation

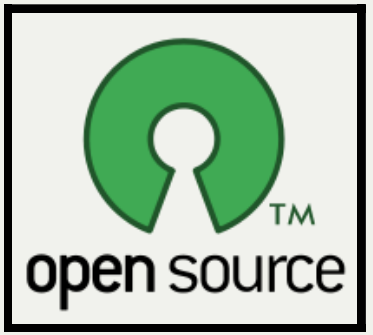
and Gnu's Not Unix

- He founded the "Free Software Foundation"
  - The "Free" refers to freedom not price; it means that once you have the software you are free to do with it as you please
  - This involves providing the source code of the software such that it can be modified
  - There is no restriction or even recommendation to provide the software for free
  - However, it tends to go hand-in-hand since a person that paid for your source code is now free to do with it whatever they wish, including give it away for free
- And Gnu's Not Unix
  - An attempt to build a restriction free clone of the Unix operating system

Four Specific Freedoms

- The Free Software Foundations specifies four freedoms:
  1. The freedom to run the program, for any purpose
  2. The freedom to study how the program works, and adapt it to your needs
  3. The freedom to redistribute copies so you can help your neighbour
  4. The freedom to improve the program, and release your improvements to the public, so that the whole community benefits

The Open Source Initiative



- Created in 1998, largely prompted by announcement that Netscape were to publish the source code for their Netscape Communicator product
- Partly in response to several software licences which all advocate the publishing of the source code of the licensed software
- Partly to evade the *"moralizing and confrontational attitude that had been associated with 'free software' in the past"*
- Provides an open source definition and certifies licences as open source licences
- Definition at: **http://opensource.org/docs/osd**

The Open Source Definition

1. Free Redistribution
2. Source Code
3. Derived Works
4. Integrity of The Author's Source Code
5. No Discrimination Against Persons or Groups
6. No Discrimination Against Fields of Endeavour
7. Distribution of License
8. License Must Not Be Specific to a Product
9. License Must Not Restrict Other Software
10. License Must Be Technology-Neutral

Open Source vs Free Software

- Roughly, the difference between the two main movements is:
  - Free Software is based on a firm principle that all software should be free, see **gnu.org**
  - Open Source focuses on the practical, business-friendly advantages to having freely distributable and modifiable code, see **opensource.org**
- Projects in either category can be understood as examples of highly distributed large-scale software development methodologies

Licences

- Licences are highly significant here:
  - GPL(Gnu Public Licence) is viral: affects derivative works, including packages using libraries
  - Many producers prefer a weaker licence which allows companies to link, for example, machine-generated parser source
  - Hence even the FSF have the *Lesser GPL*
  - Dual licensing: many products (e.g. Berkeley DB) are dual licensed, so use in commercial software presents the user with a choice between going open source themselves, or paying a commercial licence

Disadvantages of OSS

- Difficult to make money from OSS
  - but can sell support, customization, services
- Less interesting tasks tend to get less work, e.g. documentation, new user usability
- Hard for upper management to control (but is that necessarily a problem?)
- Difficult to coordinate corporate and user efforts
- Users don't have anyone to hold accountable
  - Most open source licences start with: *"This software is provided with ABSOLUTELY NO WARRANTY implied or explicit"*
  - But then again, most EULAs say something to that effect, they might just be more sneaky about it

Advantages of OSS for Users

- No restrictions on how software can be used: copying to new machines, sharing with co-workers, porting to new platforms, etc.
- Can trust the software to do what it says, not to have spyware, etc. because source code is visible
- Can count on their data created by the program being readable in the long, long term, on future platforms, although they themselves may have to maintain it
- Because development is driven by users, features are likely to match what (those) users actually want

Advantages of OSS for Developers

- No need to worry about piracy
- Users act as legions of beta-testers:
  - If you are lucky may provide the fixes, or at least find the cause
- Huge number of users working in parallel can help make development faster, but can also be overwhelming
- Companies can pool resources on a common good (e.g. Linux, Java, GCC, X.org, Android)

Tragedy of the Commons

- A term used when each individual's rational economic behaviour is to the detriment of the group
- It is therefore very difficult to get each to act within the best interests of the group as a whole
- Examples include:
  - Climate change
  - Over fishing
  - Traffic/congestion/parking

Tragedy of the Commons

- A more subtle case of this is where the tendency is to *mooch* because it is difficult for any one participant to benefit themselves without benefiting others
- Fungible goods such as milk often go unadvertised, because it is difficult to advertise for a particular producer's milk without generally advertising the benefits of milk
- Therefore each producer simply hopes the other producers will advertise, resulting in few milk advertisements and most people drinking bottled water, or soft drinks
- Hence often such goods are taxed to pay for generic advertising

Tragedy of the Commons

- The republicans successfully blocked a tax on independent real Christmas trees as a “tax on Christmas”
- But it was nothing of the sort, it was designed to provide common advertising for real Christmas trees and was subsequently inacted
- Open source can provide a means by which separate companies can overcome their instinct to mooch

Tragedy of the Commons

- At first it might appear as if open source would *encourage* participants to mooch, since whoever provides the original code is generally doing so for free
- However, open source has been tremendously useful in overcoming the tragedy of the commons between companies with different vested interests
- Part of the reason is that each can company can provide only a portion of the resources required
- Additionally, there is an obvious exit-strategy for any company that finds the project aims to be at odds with their own

Cathedral vs Bazaar

- Major SW development methodologies (Raymond 2001):
  - Cathedral:
    - Master plan controlled by one person all implementers follow it
    - Suited to small projects, large projects run by one organisation, and/or projects with only one customer

Cathedral vs Bazaar

- Major SW development methodologies (Raymond 2001):
  - Bazaar:
    - No master plan
    - Many users contribute ideas, changes, features
    - Often the principal maintainers' role is just to make sense of it all
- Successful open source projects of both types exist

OSS Success Stories

- The Internet/WWW
- Apache; the most popular web server in the world
- Linux and *BSD *NIXes, Android
- Github, Google Code, bitbucket etc.
  - April 2013: GitHub announced it had passed the 3.5 million users mark and now hosting more than 6 million repositories
- Mozilla/Firefox, eventually
- Google Chrome
- GCC, Emacs, GNOME, KDE, Python, Eclipse, etc.

Linux

- Implementation of a UNIX-like OS kernel
- Main competition: at least was Microsoft Windows
- Final code is controlled completely by Linus Torvalds who operates a “benevolent dictatorship”
- The simplest thing that works today:
  - *“Talk is cheap. Show me the code”*
- For years, used no VC tools; patches submitted and controlled by Linus; then used Bitkeeper and now git
- Developers synchronise by e-mail, mailing lists and now git
- 2014 Linux Kernel Development report: more than 80 percent of contributions are *“demonstrably done by developers who are being paid for their work.”*

Apache

- Selection of core developers is a form of meritocracy
- Development coordinated by team of core developers
- No central leader
- All major decisions voted upon by the core developers: 3+, 0- required
- Code revision is controlled by Subversion (with git access)
- Bug tracking via BugZilla/Jira
- Developers synchronise via e-mail, mailing lists, web, and newsgroups

Mozilla/Firefox/Thunderbird

- OSS version of Netscape Navigator/Messenger
- Main competition: MS Internet Explorer/Outlook
- Difficult transition to OSS; years before stable release
- Didn't seem to help Netscape the company survive
- Development is co-ordinated by the Mozilla Organization (MO), (see **www.mozilla.org/mission.html**)
- MO operates a "benevolent dictatorship" like Linux
- Each module has an owner, designated by MO, but owner can be usurped
- Code revision controlled by Mercurial, bug-tracking by Bugzilla

GCC

- GNU Compiler Collection (C/C++ and related compilers)
- Started in 1984, released yearly or so since 1987
- Part of Richard Stallman's GNU project
- Experimental branch forked in 1997 as EGCS, later usurping to become official maintainers
- Guided by GCC Steering Committee
- *Many ports are funded or supplied by hardware industry*
- Large test suite, nightly regression testing, builds, snapshots
- Code revision by Subversion
- Bug-tracking by Bugzilla

Open Source and the Internet

- Prior to the popularisation of the Internet open source was interesting and useful to users, but producers had little incentive
- As the Internet became more popular, so too was the barrier to open source participation lowered
- In short, the Internet is a fantastic development collaboration medium
- Because of this it has made more sense for companies to release the source code to some of their products
- If the product is not to be charged for anyway, it is almost a win-win situation

Open Source and the Internet

- The first major example of this was the release of the source code to Netscape Communicator, which became mozilla and eventually was cut down to become Firefox and Thunderbird
- At the time Netscape and Microsoft had competing webserver products
  - Netscape Communications Web Server vs Microsoft Internet Information Services

Open Source and the Internet

- The *"browser war"* was actually a web server war
  - Both companies wished to sell their web server software
  - Both companies made their browsers work optimally with their own web server and gave their browser software away free
  - The hope is to obtain a majority in the browser market thus making their web server software more attractive
- Releasing the source code to the browser was something of a last-ditch effort by Netscape to harness unpaid developers in an effort to prise users from Microsoft's clutches

Open Source and the Internet

- Arguably this didn't work for Netscape, it was acquired (for a vast sum) by AOL
- But this model in which:
  - Ultimately most development is done in-house by a single company
  - which employs developers to directly work on the product
  - and which publishes the source code and accepts contributions from outwith the company
  - and in doing so accepts the risk of a project fork
- has seen success for other companies, notably Google and its Chrome web browser

Open Source and the Cloud

- The rise of cloud computing has meant that the binary is often not distributed to the user, never mind the source code
- Even if the source code is published, part of the cloud offering is the service itself, including **other** users
- Although modifying the published code and deploying it yourself is possible, that might not be enough
- However, cloud platforms at least make deploying the modified application a realistic possibility

Open Source and the Cloud

- In addition, we are now seeing a proliferation of clients for cloud services, especially on phone and tablet devices
- An obvious win for open source is to take the source of a client for one platform (say iOS) and modify it to run on another (say Android)
- And of course, the original service may incorporate your changes anyway
- The fact that client code is often written in JavaScript has at least desensitised some companies to the idea of open source

Open Source and Large Software Projects

- It's clear there are some successful open source software projects
- It's clear there are many unsuccessful ones
- It's difficult to judge such projects along our three criteria
  - There is often no particular schedule and a project cannot fall behind a non-existent schedule (still no excuse Perl 6)
  - There is often no budget and in any case the currency of open source is generally **time** not money
  - If it does not fit the users needs, then *some* of those users are able to adapt it

Abandoned Projects

- There are many abandoned open source projects, typically as the main developer has too little time or simply loses interest
- A **flossmetrics** study reported that only 6.8% of projects were active
- However, the question is how much of a failure are these projects?
- Some, unsuccessful open source projects quickly garner little support
  - Such projects may be abandoned in their initial phases
  - This means very little time and other resources have been spent on them
  - Something which many large unsuccessful commercial projects can only look at with envy

Abandoned Projects

- Several companies have noticed this and allow developers more flexibility to choose the projects they work upon
  - Here the source may not be open to the public but only internal developers
  - Developer enthusiasm gives at least some evidence of a project which might succeed

Open Source and Large Software Projects

- It's at least clear that incremental software development **can** result in usable software and hence successful projects
- The difficult question is teasing out whether or not it *promotes* success

Summary 1/1

- Software is not like tangible goods
- Difficult to devise appropriate revenue models
  - But this is a problem we have seen before for example in broadcast television
  - It is also a problem for all kinds of software and non-tangible goods such as music, literature and films
- Can address the *tragedy of the commons*
- Open Source projects are examples of highly distributed, parallel, user-driven development
- Provide us with good evidence that incremental software development can work

Related Reading 1/2

1. **The Open Source Initiative site**
2. **Wikipedia article on Open Source**
3. **Slashdot** (OSS advocate forum)
4. **Producing Open Source**, Fogel 2005, ISBN 0-596-00759-0 (also free online).
5. **Open Source Development with CVS**, Fogel and Bar 2003, ISBN 1-932111-81-6 (also free online).

Related Reading 2/2

6. **The Cathedral and the Bazaar** E. S. Raymond, published and occasionally updated online, 1997-2002.
7. **Two case studies of open source software development: Apache and Mozilla**. Audris Mockus, Roy T. Fielding, and James D. Herbsleb. 2002. ACM Transactions on Software Engineering and Methodology. 11(3) (July 2002), 309-346.
8. **The Architecture of Open Source Applications** (Free book, available online, as an ebook, to buy in paper)

Any Questions?