Software Development on Linux Systems

4002-XXX-XX

By

Cody Van De Mark

Today

- Open source business approach
- Open source relationship with business and proprietary software
- Open source communities, societies and politics

Open Source Business Approach

- It is a common misconception that free software does not generate revenue
- Open source software is responsible for billions of dollars in profit each year
- It is responsible for hundreds of billions of dollars in revenue each year as most of the Fortune 500 have integrated open source software directly into their services

Open Source Business Approach

- Open source has number of business models to generate revenue

- Many, if not most, projects are non-profit and generate little to no revenue; These typically include experiments, hobbies, socially beneficial projects and small projects, but are certainly not limited to these

- Many projects take a low revenue approach, such as donations

- There are also many projects that go for higher revenue approaches, such as providing a service or consulting

Open Source Business Approach

- Lower Revenue Approaches
  - Donations
  - Merchandise
  - Freemium/Open Core
  - Partner/Referral Programs
  - Advertising
- Higher Revenue Approaches
  - Service Provider
  - Support Provider
  - License Exemption Sales
  - Business Catalyst

*Note:

Low revenue approaches

are not always low revenue and vice versa

Open Source Business Approach

- Open source business approaches are not inclusive;

  The owner of the software may choose any business approach they would like to use

- Any business approach may be used including, becoming too important to go under or charging for new feature development

- Open source business approaches may also be combined with other approaches

  Example: Offering a web service, but also including advertisements and merchandise

Donations

- Open source projects often ask for donations when running as a non-profit group
- This is a good approach when using software that you do not intend to make any profit on, or is made in social interest
- Donations provide inconsistent income, especially when the project is not well known or has a very small user base

Donations

- Donations are typically used just to offset website hosting costs to help spread the project

- Larger scale projects and well known projects can gain a lot of money off of donations

  Example: Mozilla Foundation makes ~100 million dollars a year from donations by combining donations with a business catalyst approach

Merchandise

- Merchandise tends to be one of the lowest income approaches, but can be combined with other approaches to increase revenue
- The merchandise approach is the selling of t-shirt, mugs, etc to consumers
- Merchandise is often sold through Cafe-Press or some other market, and has your logo/name on it

Merchandise

- Merchandise holds two purposes:
  - Selling a product to your users
  - Promoting/advertising your project on merchandise
- Merchandise can be sold as a non-profit status allowing you to still collect donations
- Merchandise is often combined with other approaches as a gift to those who bought a service or support, and to promote your project

Freemium/Open Core

- Freemium/Open Core is the most difficult approach and a dangerous approach to use
- Freemium/Open Core is the offering of a free, open source version of the software (the core) and also a paid closed-source version
- This is difficult due to the inherent problems that exist within the approach;

  For this reason, it is not a recommended approach

Freemium/Open Core

- The free version and the paid version must be designed very carefully

- If the paid version does not add much value or is quickly integrated into the open source code by the community, no one will buy it;

  Remember less than 5% of your users will probably even consider a paid version

- If the free version is watered down to make the paid version better, then no one will adopt the weaker open source version

Freemium/Open Core

- If the paid version is significantly better than the open source version, the community will betrayed and find/create a different project as a competitor

- Your open source version must be specifically licensed to allow a paid version to exist;

  If the license is chosen poorly, you may find the paid version will not be able to exist anymore if anyone integrates/links code under a license that prevents proprietary code, such as GPL

- Finally, open source communities tend to avoid open core versions for the same reasons and may avoid your project entirely

Partner/Referral Programs

- Partner/Referral programs allow a coexistence between multiple parties through a mutual agreement

- This approach has you make a deal with another party to recommend them as a suggested group;

  In exchange, they give you a portion of the revenue they get from your referral

  Example: Ubuntu listing IBM as a suggested server manufacturer or System 76 as a Laptop manufacturer

Partner/Referral Programs

- Partners may be support providers, services, hardware providers, other projects, etc
- It is not uncommon to link to other related open source projects or organizations to create a mutually beneficial relationship;

  As they are also open source, they may not be able to provide you with anything back besides a referral as well

Partner/Referral Programs

- Depending on the size of your project and your partners, this can be a high income approach

  Example: You are a popular web software project that refers people to consultants and reliable web hosts;

  In turn, they provide you with money from each sale they make

- You must be careful on choosing your referral partner as both of you need to look good;

  You do not want to choose an unreliable or unethical partner

Partner/Referral Programs

- You do not want to choose a closed-source competitor as they will usually try to make you look worse than their product to promote sales

- You also may not want to choose a competitor within your domain if you are offering a service (IE: Referring a service provider for the same service you offer at a similar price)

Partner/Referral Programs

- You may however, want to choose an open-source competitor if you both can work together and share code for mutual benefit;

  There are no laws against open-source monopolies as long as anyone can compete against you with your own code without any barriers

- The trick is to work together for the benefit of all, including the benefit of all projects and the benefit of society

  Example: CouchDB works with SQLite to improve both projects for developers and end-users

Advertising

- The advertising approach is offsetting cost through advertisements

- Usually ads are embedded on web pages to help offset hosting costs;

  This is a common practice on open source software websites and for free web services

- It is not uncommon for donations to be powered through advertisements

Advertising

- Advertising can be done under non-profit status allowing you to still collect donations;

  This is because advertising can be set up to “donate” to you for advertisements

- As, your site or service grows in popularity, advertising revenue increases as well;

  This can allow for fairly high revenue through advertisements

Advertising

- If you are offering a paid service or a partner is offering a paid service, you may embed ads for that service within your open source software;

  Usually this is only culturally acceptable on web pages unless the advertisement is for other open source projects

Service Provider

- Service providers offer a service that generates revenue, though the software itself may be open source
- The service provided may or may not be open source
- There are at least three approaches to this:
  - Offering a service with the open source software
  - Offering a service for the open source software
  - Offering a service unrelated to the software

Service Provider

- Offering a service with the software is when you power the service through your open source software,

  of course, anyone else may offer the same service through your software legally

  Example: Much of Google's software is open source and useful to many, but the costs to compete against them are astronomical, preventing direct competition

- It's also possible your software does not actually provide a service you offer, but allows people to build services;

  Example: Web frameworks like Django or Codeigniter

Service Provider

- Offering a service for your open source software is when people pay for a service you offer they can use with your open software

  Example: You may have open source backup software but offer an offsite back up service or storage for people who want to

- This may be tied into the open source software, but should be completely optional to the end user

- This approach should not only rely on the adoption of the open source software as it narrows the scope of the service too much;

  Instead it should work with any service

Service Provider

- Offering a service unrelated to the open source software is another common approach
- In this case you are providing a service as a business, but support an open source software package;

  Your service allows you to support to project and also run as a business
- Many service providers develop software for their needs and then release it as open source to benefit others;

  Much of Google's open source software begins this way

Support Provider / Consulting

- Support providers offer support contracts, such as consulting, training, technical support or development contracts
- This is one of the highest revenue approaches for open source software, but requires a business and the adoption of any software you are offering support for

  Examples: RedHat and Canonical take this approach; Both offer support for various software, even if it is not their own and use the revenue to continue development of open source software
- Being the ones who develop the software puts you in a better position for offering support as you will be trusted

License Exemption Sales

- Occasionally, the owner of software may sell license exemptions to companies who are interested in licensing open source projects;

  This can only be done under certain licenses and at the discretion of the software's owner

- This may be frowned upon by your community as you are selling open source code that has been contributed to you by other developers under the open source license who did not intend for their code to be used in a proprietary project

License Exemption Sales

- License exemptions are when you give permission to a certain company/group/individual to use the open source software outside of the terms of the license under limited conditions

- This may be selling written permission to use the source code under a different open source license or under a proprietary license

  Example: MySQL sold license exemptions to various companies to embed MySQL within proprietary code under limited conditions;

  The MySQL license is GPL and prevents embedding MySQL into proprietary code

Business Catalyst

- The business catalyst approach is when your software acts as a catalyst for your business approach

  Examples: Firefox using Google as the default search engine as they get donations for each Google search done

  Canonical's inclusion of their Ubuntu One service into the Ubuntu operating system

  Google including a marketplace in Android for people to purchase products while collecting 30% of the profit

Business Catalyst

- The business catalyst approach can be a very high revenue approach if you can control the catalyst;

  The consumer should not be locked into the business you are offering, but it may be built in for convenience as in the previous examples

- This approach also relies on audience adoption as it is unlikely a small audience can provide enough revenue to support the business;

  Remember only ~5% of your users will consider the offer

Relationship to Businesses

- Many open source projects are sponsored by businesses and used in a business's advantage
- Almost all, if not all, of the Fortune 500 companies heavily use or rely on open source software
- This is done because open source software allows thousands of developers to contribute to a project that may be benefiting the business;

  Recall, Debian had 73,000 man-years of work in 2007

Relationship to Businesses

- Many businesses support open source software as it is a catalyst for innovation;

  Companies can no longer keep up with the innovation of competitors without leveraging open source development

- Open source software quickly evolves and can take on thousands of developers;

  For this reason, open source cost is financially beneficial to businesses if the software is in the benefit of society or developers

Relationship to Businesses

- Companies were once fearful of open source software expecting it to undermine their business and steal trade secrets
- Some companies are still skeptical of the benefits of open source, but most have begun to openly adopt open source software
- Companies like IBM, Google, Cloudera, Canonical, Mozilla and RedHat are built almost entirely on open source software, and have integrated it into their business plan

Relationship to Proprietary Code

- Much of open source software is generated through companies who modify existing open source code
- Often companies will leverage open source code, but need to make modifications or plug-ins for their needs that get contributed back to the project
- Most development of proprietary code is directly involved with open source code as c/c++, python, php, ruby, perl and bash are open source languages;

  Those with compilers have optimized open source compilers also

Relationship to Proprietary Code

- Open source is implemented into proprietary code in a number of ways:
  - Platform - When the platform is open source (IE: Linux, PHP)
  - Linking - When a component is open source, but not the project itself (IE: Apache, MySQL)
  - Direct Integration - When open source code/libraries are directly compiled into proprietary code (OpenGL, SSL)

- All of these ways are dependent on the license of the open source code;

  Some licenses will not allow certain implementations

Relationship to Proprietary Code

- As companies commit new code back to open source projects, both see benefits
- Open source projects gain new code and integration from commits back
- Companies see their code integrated and improved, gaining them performance and minimizing their integration costs in the future by having their code directly integrated into the project

Open Source Communities

- Open source is made up of many communities designed for individual projects

- Any open source project that starts to have a user and developer base has formed a community;

  The larger the project, the larger the community

- Where open source communities differ from regular software communities, is that in an open source community everyone has a say and interest in the project

Open Source Communities

- Open source communities are typically very democratic as everyone gets a say

- Many projects have mailing lists or websites dedicated to the future of the project, in which everyone can discuss it

  Example: Ubuntu has brainstorm.ubuntu.com where users can submit ideas for the future and others uses vote on them and discuss them

- You will hear of certain communities being referred to as a “dictatorship”, but they are still democratic;

  This just means the decision is ultimately made by one person

Open Source Communities

- As anyone is contributing to the code, anyone has a direct say both verbally and programmatically

- If the maintainers are not listening to your voice, you are always able to fork the project and take it in your direction;

  This is actually a common practice within open source

Open Source Society

- Open source communities act together as a society, but rarely will you hear open source referred to as a society;

  Usually society is used when referring to open source as a whole, including software, hardware, economics, science, medicine, food, information, etc

- Normally, it is just referred to as a community or the community as a whole

- Open source communities can be very small and dedicated to projects, but communities act together as larger communities

  Example: MySQL Community → Ubuntu Community → Linux Community → Open Source Community

Open Source Society

- Each project has a community that works together to improve the project, but many projects are inclusive of other projects

- Communities work together for the improve of each project democratically

- As a developer, it is normal to be in several or many communities as you may have interest in many projects

  Example: As a MySQL community member you are interested in the improvement of MySQL, but you may also be a Fedora user and are trying to see better integration of MySQL into Fedora

Open Source Society

- Active communities are those with rapidly developing/evolving projects
- Larger communities will often hold annual summits, which are large conferences held to discuss the direction of the project, problems, innovations and new related-projects
- Much of the information gathered during a summit is diffused and pushed upward, if applicable

  Example: MySQL Summit → Ubuntu Summit → Linux Summit
  MySQL Summit → Fedora Summit → Linux Summit

Open Source Society

- Though summits do not always happen and are certainly not available for all projects, much of the information from a summit or community is still diffused into other communities
- As long as a direction is in the community's interest, they are usually open to new ideas
- Open communities can be referred to at a community level (IE: MySQL community), a greater community level (IE: Debian Community) or as a society (IE: Open Source Community)

Open Source Politics

- Open source, like anything, faces politics internally and externally

- People do not always agree on the direction of a project or decisions made;

  This is the nature of development and personal interest

- Unlike industry however, you can fork the project and take it in a different direction

Open Source Politics

- Forks projects occurs more often than you might think;

  In fact, it is somewhat common practice

- If the forked direction turns out to be the better direction, then that either becomes the primary direction or is merged back into the original

- This happens at a micro-level very often as this is how a tremendous amount of code is developed

  Fork → Improve → Merge

Open Source Politics

- At a higher level, this is less common, but still is considered a common practice, especially when testing which direction is better
- Many projects may argue different directions and create entirely new projects
- It is important to note that a complete fork of a project into a new project often divides the community, though it is possible they will merge back together in the future

Open Source Politics

- Open source politics may be internal (within the community) or external (outside of the community)

- Internal politics are the arguments of direction or design within a community;

  These often include things like which license should be used or which software should be used for a project

- These can become heated arguments and create new communities

  Example: Vi vs Emacs

Open Source Politics

- External politics are arguments about direction or design from another community or business
- Proprietary businesses who have interest within a project can put a lot of pressure on a project to go in the direction of their interest
- Similarly, other communities also can put pressure on a project to go in the direction of their interest
- Ultimately, it is up to the project community to decide, but external arguments sometimes offer valid and important points

Open Source Politics

- External politics can become heated and create tension between two projects that cause them to no longer want to work together
- This has been responsible for design decisions of projects in the past to no longer use a project even though it was in their best interest
- As a whole, the open source community tends to get along well, but there is much tension between various communities and businesses
- Occasionally, new projects arise out of external tensions where external communities create a new projects to meet their needs