Competing on Open Source: Strategies and Practise

Nilendu Pal

T.R. Madanmohan

“The notion of freeware or free software doesn't mean free in the financial sense, but instead applies to distribution, exchanging code, and innovation. To understand the concept, you should think of free speech, not free beer,"

- *Richard Stallman, one of the pioneers of the open source movement*

Introduction

For many years now, the dominant business approach employed by commercial software industry has been proprietary software. Industry giants such as Microsoft and others built their billion dollar empires providing proprietary software that builds on extending the capabilities of developers and users. Over the past five years, a new development has been occurring, that of surge in open source which involved developers at many different locations and organisations sharing code to develop and refine programs. The widespread diffusion of Internet opened new opportunities for open source projects. A number of open source products began to dominate their product categories. For example, according to Netcraft as of late 2000, Apache powers over 59 percent of server installations. Linux operates as the operating system over 17 percent of all new commercial servers (Redherring, 1999). Many of the major corporations including HP, IBM and Sun have launched major projects to develop and use open source software. For a commercial organization engaged in software product development, the ‘open source community’ is a supply base, very similar to the programmers the organization employs for developing software. Except that the open source community can produce superior quality software at a lower cost. Working with open source community can involve significant changes in way a typical software development process is managed.

To a management researcher open source development opens several strategic issues. How do firms compete with open source?. What resources become critical in managing their growth?. What strategies do they adopt to co-exist with dominant proprietary software firms?. How do they interface with communities of practice to exploit network externalities?. What strategies do they adopt to lock-in developers?. This paper seeks to address some of these issues, by making a preliminary exploration of commercial open source initiatives. Reflecting the early stage of the field, we do not seek to develop new theoretical frameworks or statistically analyse large samples. Rather, we focus on six mini-cases, three commercial companies and three non-commercial initiatives. We seek to draw initial conclusions about the key strategic aspects that underlie the open source initiatives.

Open Source Movement: A Brief historical sketch

Sharing of software, especially the source code is nothing new. As early as 1960 and 1970 researchers and developers across MIT, Berkeley, Bell Labs and Xerox’s Palo Alto Research Centre shared codes of `C’ language for developing UNIX based applications (DiBona, et.al., 1999). These projects undertaken on a highly informal co-operative basis had no restrictions on reuse or conferred any proprietary rights. This informality led to problems when firms started exploiting/enforcing intellectual property rights. For example, AT&T’s efforts in enforcing copyrights related to Unix. In response to such litigations, the first efforts to formalise co-operative software development emerged from Free Software Foundation (FSF), initiated by Richard Stallman of the MIT AI lab. The historical developments thereafter are presented below.

Open Source Time Line

Late 1983	Richard Stallman starts the GNU Project, an ambitious attempt to build an entire free operating system based on Unix. Stallman creates the GNU General Public License (GPL), which allows anyone to download, modify, distribute, and even charge a fee for the GNU source code, a process he calls "copylefting." The main stipulation is that any changes to the source code or any new software created with the code must be shared with the developer community.
Dec 1987	Larry Wall posts to Usenet the first version of Perl, a Unix-based programming language he created to scan, manipulate, and print text files. The first version is released under a GPL, but Wall feels the terms are too restrictive and writes his own distribution rules, which he calls the "Artistic License."
1989	University of Helsinki student Linus Torvalds decides to write an alternative to Minix, a stripped-down Unix variant used for teaching computer science. He wants his version to run on PCs. Linux is born. Cygnus Solutions forms to provide commercial customization and support for the GNU toolset.
1992	Torvalds decides to "copyleft" Linux in honor of the work the GNU Project has contributed to his operating system.
1993	As CD-ROMs gain popularity, for-fee distributors of free software become more prevalent.
Dec 1993	FreeBSD 1.0, a derivative of a long-standing flavor of Unix developed at various points at AT&T, Novell, and UC Berkeley, is released to the Net and on CD-ROM. The user license, like the Perl license, does not require developers to submit their changes to the source code back to the community.
Oct 1994	Bryan Sparks founds Caldera with start-up money from former Novell chief executive Ray Noorda. The privately held company, now with 50 employees, takes Linux and resells it with a variety of utilities and applications. Perl 5 is released with extensions, which give Perl programmers a much more flexible framework for adding new features.
Jan 1995	A team of programmers decides to take the source code of the National Center for Supercomputing Applications Web server, update it, and release it to the public. It is renamed the "Apache" Web server because of all the patches used to upgrade it. FreeBSD 2.0 is released.
Apr 1996	The source code for Apache is completely rewritten. Apache overtakes NCSA as the most popular Web server with 29 percent of the market. Like the GNU Project and Linux, the core source code is maintained and updated by a team of programmers. But the Apache license does not require users to submit changes in the source code back to the community.
Aug 1997	The first Perl users' conference draws an estimated 1,000 attendees to San Jose. Wall delivers the keynote.
Jan 1998	Acknowledging that it can no longer charge for its browser software due to Microsoft's competitive pressure, Netscape Communications announces it will release the source code of the upcoming Communicator 5.0 for anyone to download, modify, and redistribute. The market share for Apache and its derivatives tops 50 percent.

Source: http://news.com.com/2009-1023-207625.html

Free software Foundation introduced a novel licensing procedure that aimed to preclude the commercialisation of co-operatively developed software/products. It introduced a licensing format termed the General Public License (GPL, also known as copylefting as an antonym to copyright) under which the user had to agree to make the source code freely available (or at a nominal cost) and not to impose licensing restrictions on other users. During the past two decades, the freeware idea has resulted in numerous software projects, generating names such as Apache, GNU (which stands for "GNU's Not Unix"), and Linux. Many founders and early users of the Internet have turned to technology to develop their personal philosophies, and this spirit of communal innovation for its own sake has played a significant role in the tremendous growth of Silicon Valley.

"*Lots of people think it's strange that the commercial companies are making money from the work of the technical people, but everybody gets what he wants out the project*," said Linus Torvalds, the creator of Linux, an eight-year-old Unix-based operating system for PCs.

Most Open Source programmers would probably not write software and give it away if it cost them something to do so. However, they don't perceive the effort of writing it to be a cost. Most of them write software to solve a specific problem that they happen to be facing, or to "scratch their personal itch", as Eric Raymond (1998) points out. Lerner and Tirole (2000) based on labor economics, especially the literature on career concerns argue the economic rationale for developers openly sharing their ideas and codes. Johnson (2001) based on public goods framework argues that open source is the private provision of a public good. Open source benefits from the fact that individuals know their preferences better than a firm does and also from the fact that a greater skill set (that belongs to a community) can be exploited. Lerner and Tirole (2000) argue that while improvements in the open source software are not appropriable, commercial firms can benefit from public relations and indirectly in a complementary proprietary product segment. They argue that right governance structure and timing are very crucial for successful exploitation of open source software. This study attempts to analyse both for-profit and non-profit open source initiatives extending our understanding of the competition, use of complementary assets and revenue strategies.

Case Studies

All theory development in the empirical sciences is a mixture of deductive and inductive reasoning that involves shuttling back and forth between the two modes of thought . The study adopted a case research. The case approach is most deemed approach for this research problem, in that richly detailed information is likely to be obtained. As access to primary data was minimal, secondary data in various forms like articles, web postings, reports, literature, white papers and news that reflects the latest happenings in the firms analysed is what has been studied for the purpose of this research. In addition we also interviewed knowledgeable sources from Indian Software industry for their inputs on the development strategy adopted by the firms. In the following paragraphs we examine commercially motivated open source initiatives.

Ximian

www.ximian.com

Ximian is one of the finest examples of a successful, financially stable and profitable commercial company, whose business model is based on open source software. Ximian is one of those few case studies where we get an opportunity to understand the commercial aspect of open source initiatives, for a software company.

Ximian is the founder and primary contributor of two of the most popular and talked about open source software,

- GNOME (www.gnome.org)
- MONO (www.go-mono.com)

GNOME was started in August 1997 by a bunch of 24-year-old college students. During 1998, 'Red Hat Labs' was the first commercially funded group to invest development resources in GNOME. During October 1999, two commercial companies were formed, that made a commercial product based on the open source GNOME software – Eazel and HelixCode (later renamed at Ximian). In second quarter 2000, Henzai, and Gnumatic were incorporated. As of today, only Ximian continues to survive as a viable corporate entity developing commercial products based on open source GNOME software. Ximian has established its credibility in providing product-quality Software based on open source code base. Based on this, Ximian has able to convince some of the biggest IT companies such as Sun, IBM, Compaq and HP. to hire Ximian to port open source software on their proprietary platforms, and support the same. This has helped Ximian distribute its open source software to thousands of users, who are connected to the Internet. This has also helped Ximian build an informal alliance of open source companies, who in turn come together to offer their products jointly, to end-users. Such a distribution channel is essential for open source companies, who cannot use traditional sales channels (because of non-commercial nature of their software). Ximian is also using this platform to distribute commercial software to end-users, through a paid-subscription program. 'Red Carpet' is a channel for software distribution, which can potentially reach millions of desktops running Linux, across the world. This is of immense interest to companies developing commercial software applications on Linux, since it provides them a low-cost channel to sell their software to these millions of Linux users, who were till now out of reach for these software companies, because of high channel cost and low returns. Ximian also created 'Red Carpet', a product allowing seamless delivery platform for making open source software directly available on the end-user's desktop.

In Feb 2001, Ximian managed to rope in HP as a customer, with HP agreeing to port GNOME on HP's Unix operating system, and shipping the same as the default Window manager, replacing its own propriety software. Moreover, HP decided to have a business relationship with Ximian for the same, inspite of the fact that the GNOME software was open source, and HP was free to take it and directly from the community, without approaching Ximian. In July 2001, Ximian announced a program to create an open source equivalent of Microsoft's .NET initiative. This would potentially put Ximian in direct confrontation with Microsoft. Again, Ximian ensured that it had the backing of the industry leaders like HP and Red Hat for this initiative. In January 2002, Ximian changed the license agreement that governs the MONO project (from GPL to LGPL). This move was highly appreciated by the industry, and that immediately made Intel and HP fully commit to MONO project, and even contribute to source code developed by HP and Intel engineers, to the open source community. This is one of the best examples of large IT giants showing their commitment towards an open source projects, and even willing to contribute to it. It's a big win for Ximian and MONO, and the open source initiative, in general.

In August 2001, Ximian launched a Linux replacement for Microsoft's Outlook Mail client and PIM (Personal information manager). The complete software was open source and was available for free download. This is equivalent to taking Microsoft head-on, since it attacked Microsoft's most profitable revenue source (MS Office, which included the Outlook Mail client). But, Ximian not only pre-empted Microsoft's retaliation, but turned Microsoft into an ally, when it announced a bridging software, which will allow Microsoft's proprietary Messaging Server (MS Exchange) to work Ximian's Evolution mail client. With this bridging software (which was proprietary and is being sold at a very steep price of

US$69), Microsoft will now be able to target all the Linux desktops within the corporate and link them with its Email Messaging servers, thereby making its servers more acceptable. Also, looking at the price of the bridging software, it is almost certain that Microsoft is charging Ximian a hefty license fee to allow it to access its products with its proprietary MS Exchange software. This is a classic win-win situation for Ximian and Microsoft, and allows Ximian to exist peacefully with Microsoft, and compete with it in selected areas. Ximian has also created a community based open source initiative to develop an implementation of Microsoft's .NET framework on operating systems other than Windows. The initial Proposal for MONO project was posted in July 2001. "Miguel de Icaza", a Ximian co-founder, who currently serves as the CTO, is the founder of the MONO Project. This project is in a very preliminary stage, and it would take almost 1 year before any of this open source software can be used for building .NET solutions.

Redhat

www.redhat.com

Red Hat name is synonymous with Open Source and Linux. So much so, that when developers need to download the Linux kernel for their development needs, they mostly go to www.redhat.com rather than www.kernel.org, which is the official repository of Linux kernel sources. Red Hat takes compilers from Cygnus, web servers from Apache, an X Window System from the X Consortium (who built it with support from Digital, HP, IBM, Sun, and others), and assembles these into a certifiable, warranted, and award-winning Red Hat Linux OS. Red Hat started as a software distribution company in 1993. Red Hat is also the only pure open source software company that is publicly listed at NASDAQ, and has been profitable since the last quarter. The most distinguishing characteristic about Red Hat's open source strategy is the importance they had given to 'Branding'. Careful brand management of Red Hat Linux among Linux distributions, and careful market position of Linux among OS alternatives, enables Red Hat to enjoy a unique growth and success not seen among many other open source software vendors. Red Hat has a very focussed open source strategy, which it has practiced over the last few years:

- Take software from open source community
- Test it rigorously in its Labs for stability and compatibility
- Work with the open source community to bring the software to a product quality level
- Validate the software on different hardware platforms
- Release the software as part of a distribution (package) that is easy to install and well documented
- And finally, provide support and warranty for the software

These very steps that Red Hat adopts as part of its business strategy, bring in immense value to users of open source software. This is demonstrated by the fact that users actually pay money to Red Hat and buy 'Red Hat Linux' along with support, even though the entire software is open source and is available for free download from Red Hat's website. Unlike Ximian and other open source companies, Red Hat did not have the good fortune of being the founder of a well-known open source software project. Hence, they did the next best thing. Red Hat flagship product is the Linux OS itself. So, Red Hat hired some of the key programmers/hackers who regularly contributed to the Linux kernel, and were distinguished members of the Linux open source community. This provided 'Red Hat Linux' two key benefits: 1) provided Red Hat with an early access to the code-changes and bug-fixes that were going to be included in the next release of the Linux kernel, and 2) enabled Red Hat to get its own code-changes and bug fixes get included in the next release of Linux kernel by the open source community

This was a key differentiator of Red Hat compared to any other Linux vendor, and provided the following benefits to its customers:

1. Customers were ensured that 'Red Hat Linux' distribution is closest to the Linux kernel put up by the open source community
2. Bug fixes and workarounds provided by Red Hat for its Linux distribution stands a very high chance of getting included in the next release of the open source kernel, thereby protecting all applications software customers have developed using Red Hat Linux, and ensuring that it will work properly with the open source kernel and all other commercial distributions based on the open source kernel
3. For Developers, MIS Managers and the CIO, it meant that using 'Red Hat Linux' is a far safer option than using the open source Linux kernel or any other commercial Linux distribution.

On the marketing side, Red Hat forged partnerships with leading Hardware companies, and cut deals with them, that would allow these companies to pre-install Red Hat Linux on their Servers and Desktops. Red Hat has such deals with IBM and Dell. Red Hat also provides Technical Support services to users of Red Hat Linux. Red Hat has deeply entrenched itself with "Linux OS". Its branding is synonymous with Linux. If Linux adoption increases, Red Hat is sure to enjoy an increased market acceptance. Red Hat's other revenue sources include 'Training' and 'Certification' programs. This is a good way to leverage on the brand equity that Red Hat enjoys among Linux users.

SourceForge

www.sourceforge.net

SourceForge is web-based service provided to the open source community, for hosting projects, and provide basic project management features. SourceForge has been one of the most visible examples of open source revolution. Till date there are more than 347,073 developers who are registered at SourceForge, and participate in more than 33,359 open source projects, covering almost every conceivable domain. SourceForge provided the very basic support an open-source project needs. An Online identity for itself, and a web-based project management system where developers can come together, contribute their software code, build a community around the open source software, and continue working with the community in improving the quality of software and adding more features to it. SourceForge provides these services as part its web-portal at SourceForge.Net:

- Project Web Server: Each project has their own space for web content and CGI scripts. PHP scripts are also supported on the project web server, allowing you to build a more refined web presence for your project.
- Tracker: Tools for Managing Support: SourceForge provides a suite of integrated support management tools, called Tracker. Tracker provides bug-reporting facilities, the means for your users to submit support requests, the means for developers to easily submit patches for your review, and a suggestion box where people can post feature requests.
- Mailing lists and discussion forums: SourceForge provides projects with mailing list services and web-based discussion forums. The ability to administrate the mailing list is provided via a web interface.
- Releasing the software: Projects hosted on SourceForge may release their software through the use of a web-based file release system. Releasing the software through SourceForge provides greater visibility, and the means to track the number of times the software is downloaded
- Shell services and compile farm: SourceForge users are provided with access to a shell account via SSH. This shell may be used for basic shell functions, has its

own crontab, and may be used to directly manipulate the web content for the project. SourceForge also provides access to a diverse network of hosts running a variety of operating systems. The SourceForge compile farm may be used to test the software on operating systems that one may not have direct access to otherwise, and to generate pre-built binaries for these platforms if so desired.

- MySQL Database Services: Each project is, upon their request, provided with their own MySQL database. Projects may use this database for development and testing, or to drive a component of their project web site.
- Project CVS Services: Each project hosted on SourceForge is provided with their own CVS repository to aid in further development of your project. Developers in each project are automatically granted write access to their project CVS repository via SSH. Anonymous, read-only pserver-based access to the repository is provided to the general public.
- Category Listing: Projects hosted on SourceForge may classify themselves in the Trove, a massive database of Open Source projects. The Trove is searchable from the SourceForge site. By categorizing the project within the Trove, it becomes easier for other to locate various open source projects, quickly and easily.

The entire software running the SourceForge.Net operations was itself open-source, and was available to anybody to implement their own portal. SourceForge is owned by VA Systems (www.vasoftware.com). VA Systems started its operations as a hardware vendor selling PC preinstalled with Linux. Then, on 28th October 2001, the inevitable happened, and VA Software announced that they are discontinuing the open source version of SourceForge software, and the next version of SourceForge 3.0 will be proprietary software.

Key learning from Cases

Multiple alliances is the key: For open source initiatives to succeed it is important that the firm devise multi-usage alliances. Each of the partners should find the open source complementary and offering network externalities.

Adopt judo strategies: As seen from Xian's example open source firms should be able to move rapidly into new product areas, be flexible enough to give way (or better pre-empt) when conflicting with a superior firm (Microsoft in this case) and avaoid sumo competition (for more on judo strategies, see Cusumano and Yoffie, 1998).

Open source is not a solution for troubled business: Open source projects that are owned and supported by commercial organizations are equally susceptible to economic downturns are other commercial projects. The parent company's financial woes can easily kill an open source project. If the basic business model of a company is flawed, having an open source project will not help improve the cash flow. Having an open source software project, in parallel with commercial operations can act as a kind of insurance, during bad times.

Move from Open to Close: Theoretically, and legally, it is possible to change the license of software, and convert it from open-source to proprietary license. Then, it becomes possible to sell or commercially exploit this proprietary software by offering it as part of a commercial solution, or providing paid-services using it. One may also find this strategy used, albeit strategically by big firms to initiate development of a software/tool, and file patents once the tool is matured and has garnered enough user base. Recall Sun and J2EE and IBM and ebXML.

Developing new revenue options is not easy: Linux being open source software, it is very difficult for Red Hat to build a strong revenue model based only on Linux. Red Hat has tried to come up with open source software solutions, in the area of and E-

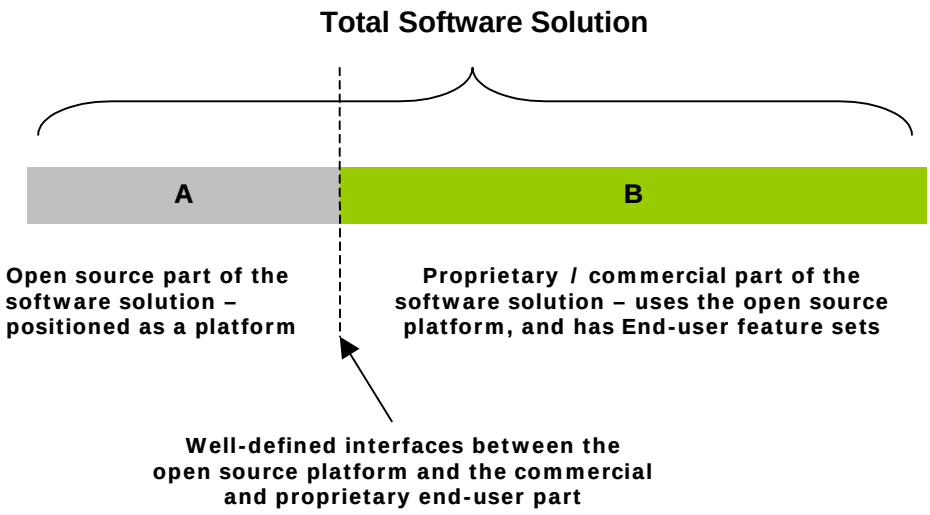
commerce and Embedded systems. But, Red Hat's positioning is so close to Linux that customers have not shown sufficient interest in these solutions from Red Hat. They come to Red Hat only for Linux, and prefer to use software solutions from other companies that have credibility in the solutions space. Ximian is another company, which faced a similar situation. Ximian was synonymous with GNOME - GUI Desktop software for Linux. There was nothing more to Ximian than GNOME. But, over a period of time, Ximian has added 'Red Carpet' – a software distribution system for Linux based systems, 'Evolution' – an Email client with Personal Information Manager (PIM), and has recently launched 'Mono' – an open source implementation of Microsoft's .NET initiative. And, Ximian diversified soon enough so as to not get identified only as a GNOME company. 'Red Hat' probably got little late here. And, it may have to pay a price for that, in terms of increased spend in creating an altered brand identity for open source Solutions business, that is different from its Linux OS. In the following section we describe how a commercial firm can plan for an open source practice, and how to organise it.

Developing an Open source practice

Historically, open source initiatives have typically been closer to the Infrastructure and Platform space, and less towards the End-use/GUI side. Most of the open source initiatives led to the development of software, which was difficult to be used directly as an end-user application, but could be used as a backend solution to implement an end-user software product. The reasons are: 1) it is hard to develop End-user applications, because of non-standard graphical systems, and also because programmers are not good graphics interface designers and 2) much of the open source software was written by engineers to solve tasks they had to do while developing commercial software or services. So the primary audience of the open source software were originally engineers Given the nature of the open source movement, it is prudent to consider open source initiatives for the platform portion of a software solution. The end-user part of the solution is better developed by a commercial entity, under closed-source environment. Hence, building a business model based on open source initiative involves the following steps:

1. Splitting the total product/solution into two parts - 'platform' and 'end-user'
2. Creating an open source community to develop the 'platform' part of the total solution
3. Developing the 'end-user' part of the solution in-house, while having close interaction with the open source community

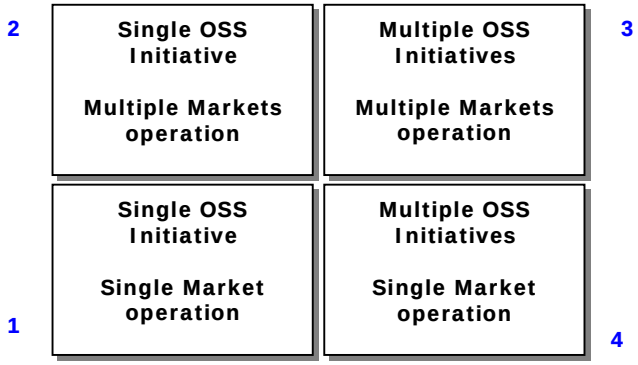
Having identified the 'platform' and the 'end-user' parts of the solution, the company needs to work with an open source community to develop the 'platform' part of the solution, and need to constitute a development team within the company to work on the 'end-user' portion. A software solution based on open source based can be seen as:



Here, **A** is an open source project, which implements a 'platform' for an overall solution. Commercial organizations/vendors can implement commercial/proprietary end-user components over the open source platform. **B** can also include 'software services' which a vendor may provide as part of customisation/porting of the open source platform software for a specific requirement. The dotted line represents the interface APIs that allow the non-open source software to interact with the open source software. The interface APIs should also be in public domain.

Types of Open source initiatives

Given the above framework, there are four ways a commercial organization can constitute its open source practice.



Single OSS Initiative - Single Market

The companies following this strategy are focussed in one market space only, and use only one open source technology as the platform for building products and solutions. Such companies are usually Single-product Company, or having a portfolio of related products for a single vertical market. Such initiatives are the simplest to operate. And typically, most of the open source initiatives start in this manner. Once the open source software matures, and becomes stable, the community starts looking at other markets/domains where the software can be used.

This particular model is well suited for smaller companies who do not have a large workforce, and there is no surplus engineering manpower available within the organization. But, considering the investment required in initiating and managing an open source community, the cost might far outweigh the benefits if the operations (and revenues) of the organization are not significant. At this point, we do not have a measure of what should be a reasonable qualifying size/revenue beyond which an open source initiative can add value to a company. It is left to individual organizations to weigh the costs and benefits, and take a decision about pursuing such an initiative.

Single OSS Initiative – Multiple Markets

The companies following this strategy are leveraging a single open source technology across mutiple vertical markets. Intitutively, this seems to be the ideal model to be followed, since this effectively leverages the invesement made in the open source initiative across multiple products for different vertical market segments or domains.

In this model, the open source initiative is chosen such that, the output of the initiative will provide a 'platform' which can be used to build various commercial products for different market needs. To adopt this model, the company should be operating in multiple markets with specific offerings in those markets. It should have significant engineering resources to support an open source initiative and collaborate with the public-domain community. Large organizations, that have a portfolio of products or services for multiple vertical markets, with any single technology focus, are likely to adopt this model.

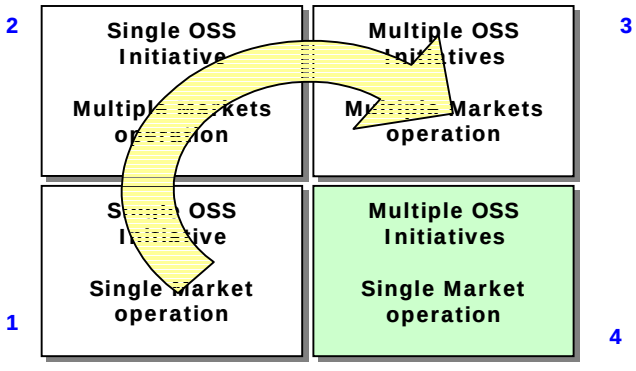
Multiple OSS Initiative – Single Market

This is quite a benovelant model, wherein the company spends considerable bandwidth and resources to initiate and manage multiple open source initiatives, and leverages it for its products/services for only one domain/vertical markets. Companies that are likely to adopt this model are those, who are world leaders in a particular product/technology, and dominate the markets. An example could be Nokia, which uses multiple open source initiatives to add value to its products and solutions for the Telecom market.

Multiple OSS Initiative – Multiple Markets

The companies following this strategy are typically large companies, who can sustain multiple open source initiatives, and leverage them across multiple markets. Or, these could be companies, who were involved with other open source initiatives, due to historical reasons, and now want to constitute additional open source initiatives, that can provide technology and platforms for building products and solutions, to service multiple markets.

Out of the 4 quadrants that are possible, choosing the appropriate quadrant is a strategic decision, since it determines the rest of the open source strategy for the firm.
It is also understood that the quadrant will change with the evolution of the company and changes in the environment. Hence, there will be shift from one quadrant to another over a period of time. The diagram shown here depicts the typical transition that will take place over a period of time, for most of the companies. The 'Single Market-Multiple Open source initiatives' will remain as a special case, and will be appropriate for particular type of companies as described in the previous sections.



Open Source and Business Focus: Options and Challenges

By nature, product companies are focussed and have a tendency to grow exponentially over a period of time. For them, to stay focussed and leverage any open source initiative

to the maximum, ‘Quadrant 2’ is the safest option to choose from. But there can be a situation, where a company may opt for ‘Quadrant 3’ type of initiative. This may happen, if a product has requirements for multiple modular components, and there is a threat of some of the competitors working on an open source initiative for some other components.

Services companies have a different problem to face, since open source initiatives are predominantly used for developing software components that go into products. Here are some of the ways in which service companies can benefit using open source initiatives.

- If they are offering system integration services or implementation services using a set of standard proprietary product, they may like to promote an open source initiative to replace the product that costs the most, or which has a monopoly supplier. Of course, the feasibility or implementability of such software using open source methodology needs to be taken into account.
- Service companies can split their offerings into two parts - open source (free) offerings, and proprietary commercial (paid) offerings. Hence, depending upon their markets and offerings, they may go in for either Quadrant 1, 2 or 3.

Open source Ratio and Competition

The variation in the ratio of open source (platform) and end-user (proprietary) component of the total solution affects the revenue model, and other business parameters for an organization. This would be true not only to a software product, but to a service offering, based on open source platform as well.

For products where X is very small, the net offering will have a very small open source component, and a large proprietary content. Such products are likely to be similar to existing proprietary products and will follow similar business rules that the proprietary products use to compete in the marketplace.

- Because of the low content of open source component, there is no significant cost-reduction that the company can take advantage of. Hence the cost structure of this product will also be similar (maybe slightly less) than other proprietary products.
- Customers are likely to perceive these products as proprietary products, and will base their purchase decision based on factors used for purchasing proprietary products. And, these products will face direct competition with existing proprietary products.

As X increases and approaches a high value, the final product will have a very high open source content. Such products will be radically different than the existing proprietary products.

- The open source products will have a very different cost structure, allowing greater flexibility in pricing, compared to the proprietary products.
- On the other hand, competition with companies building products using the same open source platform will be intense, since there will be a much smaller differentiation.

Hence, the key point here is to decide on an appropriate ratio, that will be beneficial to the company, and will allow it to compete favourably against the proprietary products, and at the same time keep the competing open source products at bay. Varying the ratio gives rise to new product offerings that are commercially and financially different. So, a company can also have multiple product offerings that basically differ in the percentage of open source component contained in the product, to service the needs of various segments of the marketplace.

There are no readymade solutions for choosing the appropriate ratio, since it depends on individual company's capabilities and business focus. But, we present here some basic guidelines that will help companies arrive at an appropriate ratio for themselves.

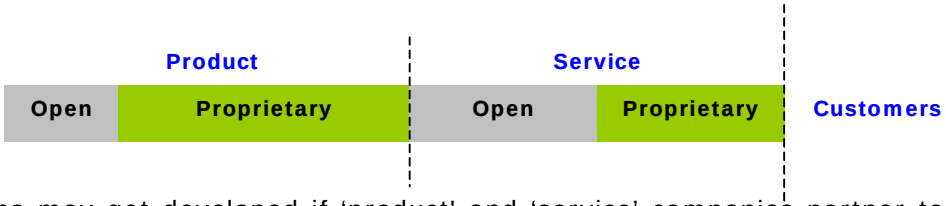
- As far as possible, companies should try to have the part proprietary, where they have an advantage against their competitors. This advantage can be because of
  - Intellectual Property ownership (patents, etc)
  - Exclusive tie-ups or business relationships, that deny the competition access to certain intellectual property, that is necessary to develop the product
  - Software components that is already available with the company, and not with the competition, and which requires a significant amount of time/resources for development.

- In case of an integrated solution (product + service), the proprietary component can be a service component, if the company has a better service infrastructure than its competitors.

- If the company is small in terms of resources and engineering capabilities, and the competition is made up of large firms, the company can rely on its speed, agility and faster decision making abilities, which the larger firms are likely to lack. Hence, the proprietary components can consist of components that rely on evolving/rapidly changing standards, or new functionality that is yet to become a standard but provides a lot of value to customers.

This model can also be adopted for services. In that case, the 'open' part refers to a free of public service that is offered, and the 'proprietary' part refers to a premium service that comes with performance guarantees and service level agreements. Service companies are likely to use the 'open' services as a low cost promotion strategy, or to increase the penetration level of their services to a larger population, to whom they can later sell their proprietary services, that come for a price. Service companies usually provide services based around a commercial product. In near future, service companies are also likely to use the open source model to encourage alternatives to some of the existing commercial products, so as to negate the advantage enjoyed by the product company or to bargain for a better deal. Hence, companies who are promoting open source product initiatives may approach service companies for their support, both as a contributor to the open source development, and as a partner who can use the open source product to offer solutions to end-customers.

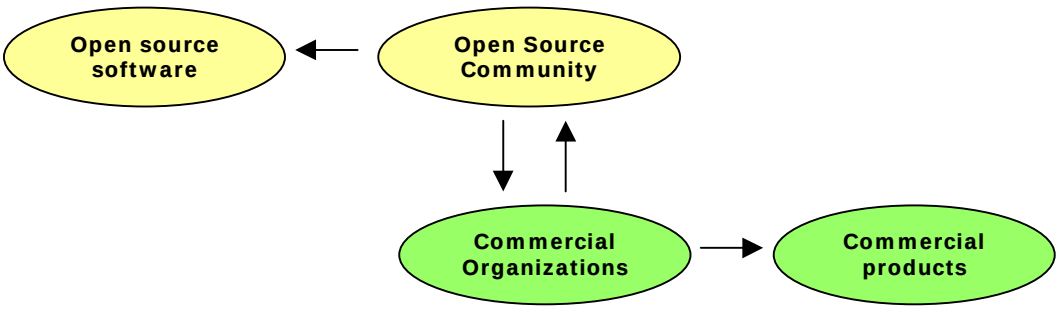
There can even be 'solutions' (products + services) that can have the following design:



Such solutions may get developed if 'product' and 'service' companies partner together to provide a combined solution offering to customers. The emphasis here is on the fact that 'Service' companies can design a service offering around a product, that is part 'open' and part 'proprietary'.

Open Source Practice - Implementation Plan

Traditionally, open source initiatives have focussed only on the community, the companies sponsoring the open source initiative, and the open source product being developed.



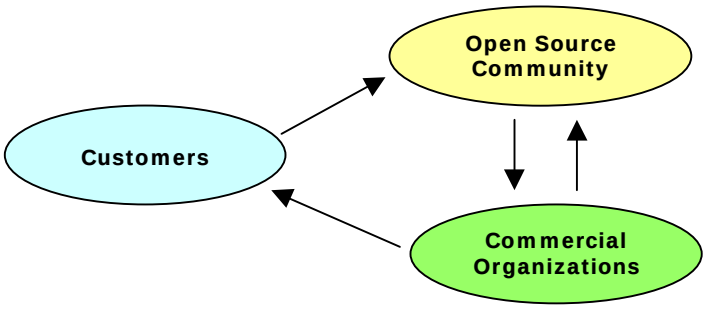
In this model, customers of the commercial products were not involved. Hence, there was no opportunity to win their trust. The only way customers will be trusting the software solution, will be if they were involved in the development of the open source version of the software, or had used the same earlier.

So, if customers already are using the open source version of the software, it creates a new problem for the company in terms of convincing the customers to buy the commercial version, especially when the difference between the open source version and the commercial version is not significant.

Hence, as a sponsored open source initiative by a commercial company, they have two ensure atleast two things:

- Make sure that the open source version of the software being developed by the community is not in direct competition to their own commercial software
- Get the prospective customers involved in the development of the open source portion of the software, so as to gain their trust, and make them feel part of the initiative.

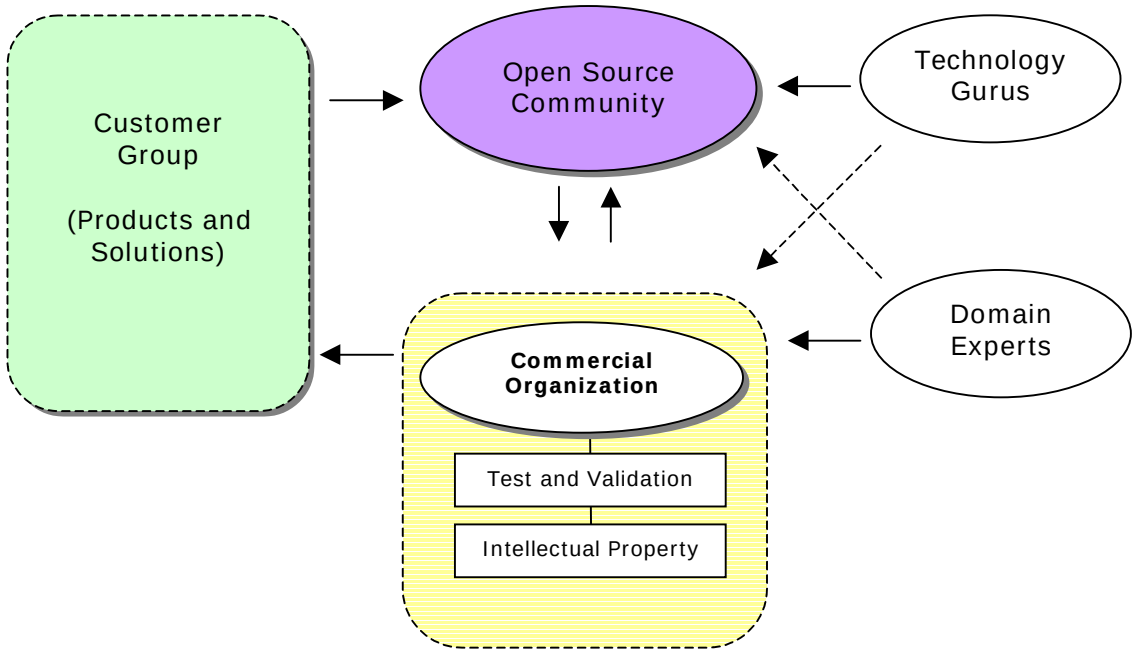
The first problems is taken care by our earlier models, wherein we only look at developing a ‘platform’ as part of the open source initiative, and keep the ‘end-user’ portion of the software as a proprietary component, being developed by the company. That way, the open source component will never be in direct competition with the final commercial end product. To take care of the second issue, we introduce the participation of customers as a necessary stakeholder in the overall open source initiative. Having ‘customers’ as a group, the objective now is to make them feel part of the initiative and get them to contribute some software components/modules to the community.



There are other groups that can add significant value to any open source community.

- **Domain Experts**
  They essentially are industry experts with significant experience, and help bring in the perspective of the end-user/customer. They add a significant value in terms of developing a high level design of the software.
- **Technology Gurus**
  They are the technology stalwarts, who have spent considerable amount of time working with technology, and have a significant expertise. They are invaluable when it comes to fixing a critical bug, or designing new features over existing code base.

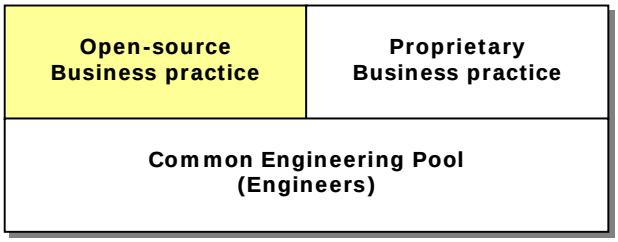
If we combine all this together, we end up with an integrated framework for driving an open source initiative. Described here is the framework that is being proposed by the author.



The framework described here is meant as a basic guiding model for implementing a business practice within a commercial organization, based on open source principles. This framework describes the various interactions that take place between the commercial organization, customers, open source community, domain experts and technology gurus, and ensure that those are taken into account while formulating an open source strategy.

Implementation Strategy

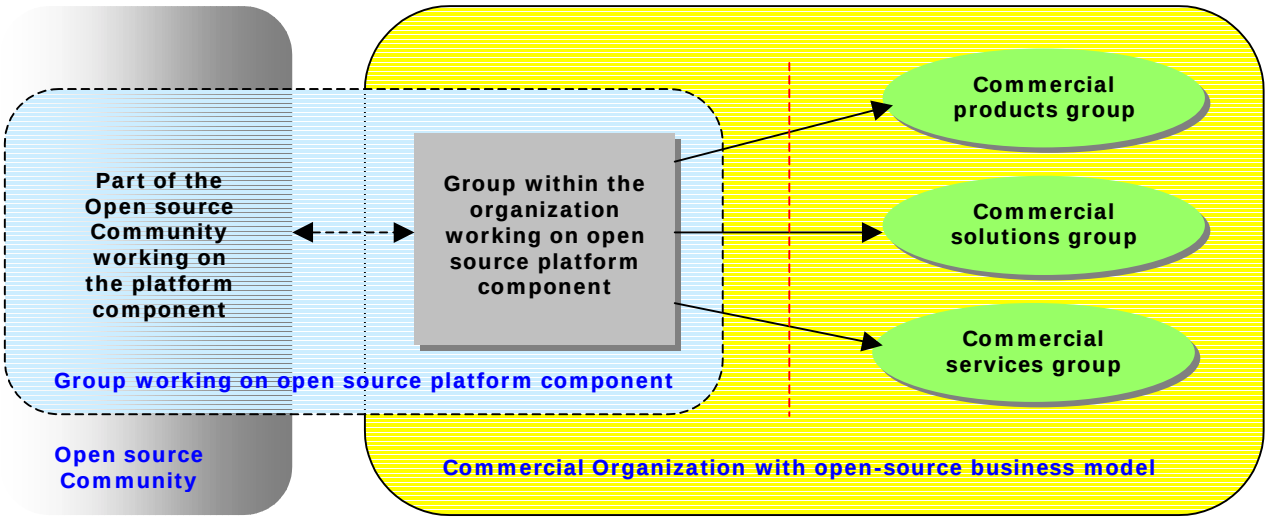
In this section, we present an implementation plan for an organization that wants to create an open source practice along with its current proprietary business practices.



It is recommended that at a strategy and planning level, the open source practice be kept separate and distinct from the proprietary practice. Open source is a new and evolving model for commercial organizations. Moreover, it requires an understanding that is radically different from the traditional proprietary software development model. Hence, to prevent misunderstanding and bring in clarity, the person heading the open source initiative should not be involved with proprietary methodologies. At an engineering level, there is not much different between open source and proprietary software. Moreover, majority of the engineers in the organization will be working on the proprietary portion, with only a few working on the open source components and supporting the public community. Hence, we do not foresee any problem in having a common pool of engineers working on both open source and proprietary software. In the next section, we detail the open source business practice group.

Structure of the Group

For each of the domain for which an open source practice is being planned, there needs to be a group working on that particular domain, and continuously interacting with the public domain community, and building the platform software. Based on this platform software, the organization may plan to launch a range of products/services after additional value-add on top of the open source platform.



This sort of a structure can be followed for each technology/market domain, for which an open source initiative is being planned. For the open source platform development group, there needs to be a person from the company, who is in-charge of the group within the company, and represents the company in the community. It is also beneficial to included members from outside the organization as part of the open source business practice. These members could be recognized technology gurus, domain experts, and evangelists, who can provide credibility to a ‘public community’ and can also guide it using their experience and expertise.

Infrastructure and Resource needs

Some of the infrastructure requirements for executing such a open source project are:

Web based Project Management system

- There should be a common web-based project management system, that would provide project information, discussion boards, mailing lists, file download system, bug tracking system, CVS, and a web based announcement (news posting) system.
- Such hosting facilities are provided by public-domain websites like SourceForge.Net. But, the company can provide such a hosting facility at its own corporate website, usually under a sub-domain http://opensource.company.com

Tools and Development Environment

- The group within the organization working on the open source platform portion should be using the same development tools and environment that is being used by members/developers of the public domain open source community.
- To speed up the learning cycle, the company can provide a package of open source tools and utilities that will be required during the course of development
- The community should define a coding-standard that will be followed. Hence, all members who want to contribute source code should conform to the coding standards for being considered for check-in into the central source repository

Rules, Regulations and Processes

Based on the structure proposed in the section, there should be two distinct groups of people, one of which is working on the open source platform component, and the other on the commercial end-user component.

Measurement criteria

Open source platform group

As a group, the measurement criteria should be a set of deliverables that can be tracked and measured. The idea here is to not have financial objectives. Some of the measurement criteria could be:

- % Completion of the platforms software component
- Number of active contributors in the public domain group
- Activity percentile of various sub-group within the community
- Total numbers of users/members in the community

Proprietary end-user group

This group can be managed as any other business group within a commercial organization. Some additional measurement criteria can be introduced, that will ensure a cohesive working environment with the open source group within the organization. They can be:

- % of open source platform components, that is being used to build the commercial products (more the percentage, the better it is)
- % of total bugs for the open source platform component, that have been reported by the commercial group (this gives an indication of the extent to which they are using the features of the platform component)

Open Source Community Management

Studies have shown that transparent management of trust is imperative for open source initiative to foster and designing appropriate intermediaries would facilitate trust building (Dessein, 1999). Employing visible open source developers or evangelists and involvement of media with strong commitment to open source need to be considered while designing mechanisms of community management.

Broadly the community should have a two-tier structure:

Tier-1: Board of Directors

- The Tier-1 of the community is the board of directors, that will play a role in providing leadership to the community, and provide directions to it.
- The community should elect a Board of Directors for a fixed term, which should be given the responsibility to manage the community and arbitrate during dispute.
- The board of directors should be elected from the members of the community, in a democratic manner.
- In exceptional circumstances, distinguished members outside the community can be requested to be on the Board, to add credibility to the project, or to bring in knowledge and competency that is not available within the community.

Tier-2: Developers, Contributors, and Users

- This group will essentially constitute of all the developers, contributors, users and general members of the community.
- The emphasis here should be to get those people to contribute who already own "software implementations" useful for your project, and are willing to make it open source.
- We should also look at participation from developers, who are employees of prospects.

- The Categories of Contributors are:
    - Promoting company (includes strategists, zealots)
    - Users of the project (customers/prospects)
    - Technology Gurus
    - Domain Gurus
    - Freelancer programmers
    - Journalists/Cheer leaders

OSS Project Management

Code Check-In

- The community should nominate a member from the Board of Directors, who along will have write privileges to the common source code repository of the community. All code changes and updated should be sent to this member, and he/she alone will have the authority to decide what goes in, and what does not, based on the objective of the project, as decided by the Board.

Project Management

There should be dedicated responsibilities for managing the

- Code Check-in process (Code Captain)
- Bug database maintenance
- Infrastructure Support
- Bug Database Maintenance
- Documentation/Website maintenance

- Only the web-based project management portal should be used for all official development activities of the community. This will prevent fracture of the development effort, and also minimize code-checkin problems.

Offline Meets

- If resources permit, offline (physical) meets can be organized that can provide an opportunity for community members to come face-to-face and build an offline relationship with each other
- Such meets/events can also provide an opportunity to reward key contributors, and help promote the community by inviting the 'Press'

Open source licensing

Choosing the proper licensing scheme for the open source platform is very important, both from legal and commercial perspective. There are several 'Licenses' under which various software has been released in the public domain. Some of the most common open source licenses are

- GPL (GNU Public license)
- LGPL (Lesser GPL)
- MPL (Mozilla Public License)
- BSD License
- MIT License

All the license schemes that are approved by the Open Source Group are listed at: http://www.opensource.org/licenses/index.html

These licenses vary significantly in terms of the freedom available to users of the open source software for enhancements and using it in proprietary applications. GPL is considered to be one of the most stringent of the open source licenses, and it mandates that all enhancements made to the open source software be contributed back to the public domain, and any use of open source software within a larger system results in the entire system becoming public domain. Due to these reasons, GPL has become very unpopular with the community that seeks to build commercial businesses based on open source software. The current trend is to use MIT License, or LGPL, which allows significant benefits to users of open source software, in terms of ability to retain enhancements to public domain sources, and including open source software within proprietary software. For companies who intend to release sources under open source licenses, it is important to consult Intellectual Property lawyers, and understanding the implications of a particular license, before adopting the same.

Marketing and Promotion

Open source projects typically take anything between 6-months to 3-years for producing a tangible and usable software. Such long period of time may see significant changes in motivation level of the community members, either due to external factors, or just because of the long time period. Marketing and promotion within the community and outside the community is very important to keep the momentum on and increase the

motivation level of the community members. For such needs, having media representatives as members of the community helps in a big way. They are in a better position to judge the mood of the community, and provide the necessary motivation by projecting the achievements of the community to the general public, and to other communities.

Also, participation in Open Source Conferences and Events can provide significant visibility to the community and its members. ‘News Updates’ on the communities activities, circulated within the community, on a regular basis can help bring the community together, and keep them informed of each other’s achievements, specially for large communities with many sub-projects, where there may not be much interaction across various sub-groups.

References

Cusmano, M.A and Yoffie, D.B. Competing on Internet Time, Free Press, 1998.

Dessin, W. 1999. Authorities and Communication in Organizations, Unpublished Working Paper, Universite' Libre de Bruxelles.

DiBona, C., Ockman, S and M. Stone (Eds), 1999. Open Source Voices from the open source Revolution, Sebastopol, California: O'Reiley.

Johnson, J.P. 2001. Economics of Open Source Software, Unpublished Working Paper series, Massachusetts Institute of Technology

Lerner, J and J. Tirole, 2000, The Simple Economics of Open Source Software, NBER Working Paper 7600.

Raymond, E.S. 1998. The Cathedral and the Bazaar, http://www.tuxedo.org/~esr/writings/cathedral-bazaar/cathedral-

http://primates.helixcode.com/~miguel/gnome-history.html
http://primates.helixcode.com/~miguel/helix-history.html
www.ximian.com
www.redhat.com
http://www.oreilly.com/catalog/opensources/book/young.html
www.sourceforge.net
http://news.com.com/2100-1001-275469.html
http://www.grantbow.com/letter.html
http://sf-genericinst.sourceforge.net/
www.vasoftware.com/sf