

FOSS Business Models for Developing Countries in Asia

Rahul De'

Indian Institute of Management Bangalore, Bangalore 560076, India

© IOSN South Asia Node (CDAC Chennai)- 2008

This work is released under the Creative Commons Attribution 2.5 License. For full details of the license, please refer <http://creativecommons.org/licenses/by/2.5/legalcode>

1 Introduction to FOSS

Free and Open Source software (FOSS) refers to software that is made and distributed in a particular manner that differentiates it from what is known as proprietary software. The first difference is in how FOSS is made: making software entails writing lines of code in some high level language such as C++ or Java that is then compiled to create binaries that can be run on any computer. Binary software is unreadable to programmers, as it is in ones and zeros, and as such cannot be directly modified; the source software is needed for that. Typically, the lines of code are written by programmers who work in teams and collaborate to create huge repositories of code, running into millions of lines for some applications. As would be evident, there are many complexities in writing such huge software programs and they arise mainly due to the requirements of coordination and design that are necessary to obtain a properly functioning binary at the end of the exercise. If all the programmers writing the code belong to the same company and their coordination is controlled by the managers within the company, then this style of making software is proprietary. If the programmers do not necessarily belong to any particular organization but are independent programmers coordinating through the Internet and writing code in a collaborative manner, usually of their own volition, and making all their code available publicly, then this is referred to as FOSS.

The second difference between FOSS and proprietary software has to do with the manner in which the software is licensed for use and distribution. Broadly, FOSS licenses encourage open and un-controlled distribution, whereas proprietary software licenses specify direct controls and restrictions on who can use the software and how they can share it. There are a very large number of licensing styles followed by FOSS practitioners, however the broad principle adopted by all is that of allowing open and free access. It is interesting that one of the few restrictions that is imposed on FOSS licenses is that of preventing changes to the license that blocks the free/open source nature of the software.

In popular writing and speech, FOSS is often referred to as 'Open Source,' which is the term that was commercially accepted by some companies that had based their business around FOSS and in particular around Linux. However, the terms 'Free Software' and FOSS are also used quite commonly.

1.1 Brief History of FOSS

With the advent of the computer industry in the 1950s in the United States and Europe, software was treated as the instructions that were packaged with the hardware that was the main product sold. Companies like IBM gave the software bundled with their machines, and this included compiled binaries as well as the source code. Source code was distributed on physical media and modified by the users to suit their needs.

Along with the companies that primarily made hardware, there evolved companies that primarily made software and which comprised of firms that wrote software for users as well as manufacturers of computers. With such firms, software as a product became important. Two basic kinds of software evolved: systems software that was primarily used to drive the hardware

and control all peripheral devices, and software that was mainly intended to solve business or scientific problems. This division created two types of products that companies made - system software and application software. These companies evolved strategies to protect their investment and effort in writing the software, what is now called intellectual property (IP), and so introduced licenses that prevented distribution and sharing of software. This took a sharp turn in the late 70s and early 80s with the growing popularity of personal computers and mini-computers. Many companies were founded, such as Microsoft, whose main line of business was in manufacturing software whose rights they were keen to control rigidly for their own continued profits and survival.

It was as a reaction to this turn of events that several groups of people around the world started writing software that they desired should remain 'open' and freely available for anybody to use and modify. The most prominent groups were at MIT and at Berkeley. A researcher in the MIT Artificial Intelligence Laboratory, Richard Stallman, initiated the Free Software Foundation whose explicit goal was to create software, system as well as application software, that could be freely distributed and used and modified by all. For this purpose, Stallman invented the Gnu Public License (GPL) that became a standard for free software licenses. The group at University of California at Berkeley was involved in developing software and utilities for an operating system that later came to be known as Berkeley Software Distribution (BSD). Here too the licensing was such as to allow free use and sharing of the software.

In 1991 a Finnish computer science student, Linus Torvalds, posted a message on an online message board asking support to build a kernel for a Unix-like (Unix is a kind of operating system) operating system that he had started constructing. He received a large response from enthusiasts all over the world and riding on the newly emergent commercial Internet, this

project became very successful. The Linux operating system was a result of this initial effort, an operating system kernel that has literally become synonymous with FOSS and has spawned an industry around itself. Linux was built using the tools already developed by Stallman and his group and adopted the GNU licensing scheme for its own development.

1.2 The FOSS Culture

FOSS has promoted a culture of openness and free use regarding software and information technology in general that has attracted much investigation and comment. What intrigues many is that there are now thousands of people around the world who are willing to spend a huge amount of their time and money to develop extremely complex and very useful software that they simply give away. Why do they do this? What motivates highly competent programmers to give hours and days of their precious spare time to build highly sophisticated software for which they know they will not receive any compensation?

The original reason is that this was an attempt to subvert industry control. In interviews Stallman stated that he was upset with the fact that he could not share with his colleagues the modifications he had made to some commercial software. In many cases he couldn't even modify the software even though he had identified bugs and was willing to give the modified code back to the creators. With the advent of personal computers, this trend of preventing users access to source code became quite common and Stallman initiated the Free Software movement. However, this reason, though rich with political overtones, is not sufficient to explain why thousands of young and relatively inexperienced users, such as those in colleges in India, joined the movement to write and give away software.

Many argue that programmers join the FOSS movement to satisfy a personal itch. It is an

attempt to build something that is not available, in the manner that you would want for yourself, and then making this available to the community for others to use and also help build. Linus Torvald's example to build an operating system that would suit his dreams resulted in his initiating the Linux community. There are over 200,000 projects on SourceForge.org, an online site for FOSS developers to start and manage projects, and many of these have about one or two developers associated with them. This is an indication of the nature of the 'itch' that FOSS developers have - they want to start something new and initiate a project with some goals in mind, regardless of how many will join them and hoping that many will.

FOSS developers who are associated with important and very successful projects such as Linux or Apache (Apache is a FOSS product that is used to host or 'serve' web pages; it is the most popular server on the web) command a very high regard in the community of developers. New and inexperienced programmers see them as role models. This is a culture of peer recognition that permeates the FOSS community and is a means of self-actualization. Programmers want their work to be recognized by their peers and so are willing to work very hard and diligently. Making a contribution of code to the Linux kernel (the kernel is the central part of any operating system), for instance, is very difficult. Anybody wanting to do so has to have a very good grasp of this vast and complex software and then make a contribution that will be scrutinized quite acutely by a community of very experienced and talented peers, and then may be included in the kernel. This is a very difficult task and is comparable to publishing an article in a top academic journal. If a contribution is made then the recognition is immediate and very satisfying for the developer.

FOSS contributors are sometimes understood to be participating in 'gift economies.' Gift economies are different from exchange or market economies in the sense that the acts of giving,

sharing and openness are privileged, as opposed to measurement of value, exchange for profit, and closeness. Gift economies enhance the self-esteem of the giver, as the community to which the giver belongs values the gift by further sharing, and enables the giver to access gifts from others. The gift economy nurtures values prevalent in ancient tribal cultures that celebrated belonging, community, sharing and giving, while at the same time acknowledging mastery of the contributor and his/her independence.

In contrast to the gift economy, some observers conclude that FOSS developers actually participate in giving to enhance their own worth and rating in the community for better job opportunities. Many programmers, particularly those starting out, seek to make contributions to projects that have corporate links and where they are able to seek employment based on the evidence of their contribution. For many, FOSS participation is a direct source of income, as in the case of MySQL (MySQL is a FOSS database application that is very popular; a case study is included below) developers who are paid by the company for working on code used by the company, although they are not directly employed by MySQL.

1.3 Open Standards

One of the outcomes of the culture of FOSS is that of a permeation of the demand for Open Standards. In the context of Information Technology (IT), there are many standards that exist that are maintained by international standards bodies as well as by individual governments. A standard, such as one maintained by the Institute of Electronics and Electrical Engineers (IEEE), specifies a design for a particular object such as software or hardware, that will be adopted by all parties involved in making or using the objects. Standards enable inter-interopability amongst IT components and are strongly guarded as well as supported by interested parties. Standards are typically proprietary, that is, they are owned by a particular entity, individual or corporation,

and therefore their usage by others requires payment of royalties. Open standards, on the other hand, just like Open Source software are unrestrained standards that can be adopted by anybody without the obligation of paying royalties.

An example of an Open Standard is the Open Document Format (ODF). The ODF specifies a format for word processors, spreadsheets, presentations, graphics and formulate. This format can be implemented by any software in an unrestricted manner. The ODF is being promoted by many in the FOSS community, however there is strong opposition to this from the proprietary software vendors. Many governments and standards bodies have taken a stand on the ODF, and many that have not are under pressure to do so.

1.4 FOSS Licenses

FOSS licenses enable a wide range of freedoms for use, distribution and modification as well as restrictions and limitations. Licenses are held by individuals, or organizations, and are created to suit the needs and purposes for which the individuals or organizations created the software in the first place and the manner in which they wish to share.

Broadly, FOSS licenses may be classified under three broad categories: the Berkeley license, the GNU General Public License (GPL), and the Mozilla license. Most licenses, and there are many, fall under these three categories, in the sense that the licenses are derived from those defining the category.

The Berkeley license was created by researchers at the University of California at Berkeley and is the most 'open' of all the licenses. It essentially allows free use and distribution of all the software, both source and binary, covered by the license. It also allows unrestricted modification, then distribution of modified and derived works in any manner and under any licensing

arrangement (a software package that is built up from an existing package, by adding more code, is called a derived work). It only insists that the copyright information of the original software be carried with the derived and modified works. Also, it disclaims any warranties of performance, and prevents use of the originating organization's name to endorse or promote derived software products.

The MIT license is similar to the Berkeley license and allows the recipient of the software to use, distribute, modify, copy, merge, sell, or license the software in any manner possible, with the restriction of retaining the copyright notice and the permissions thus granted. The Apache license (Version 2.0) is also similar to the Berkeley license, however it is a much more detailed license statement, specifying clearly the meaning of terms such as 'derivative works,' 'contribution,' 'object,' 'source,' etc. The Apache license then clearly states the rights granted to the licensee for use, distribution, modification, licensing, etc, with the explicit provisions of granting both copyright and patent licenses to the recipient. This explicit grant of copyright and patent provisions is special in the Apache license and makes it attractive to many FOSS product-based businesses.

The GPL was authored by Richard Stallman in the early eighties and remains one of the most prominent and widely used free software licenses. In its intent the license enables the recipient to freely use, modify and distribute any software, and also carry forward these rights to further users of the software. Under GPL, a modified program may be distributed, however, the new program has to have the same, GPL, license and the source of the program has to be distributed also. The restriction of retaining the original license ensures that future recipients will not in any way be blocked from further modification and redistribution. The license demands that the copyright information, information on modifications, and information on absence of any

warranties be included with the software.

One of the restrictions that GPL imposes is that of not allowing combination of GPL software with software having other licenses. To overcome this, Stallman also proposed the GNU Library General Public License (LGPL) that permits such combinations, and is meant for sub-routines or partial programs that work only with other calling programs. Also called the Lesser GPL, this license explicitly allows the code to be used with non-free or non-GPL license calling programs. Care must be taken to read and interpret the LGPL carefully, as it is quite precise about what is a derivative work and what is not. Essentially, a program that dynamically calls a LGPL library is not a derivative work, however one that has to be compiled with a LGPL library to create a single executable becomes a derivative work and hence is bound by the license.

The Mozilla Public License (MPL) was created as part of the effort to create a FOSS browser based on Netscape's (Netscape was one of the first Internet browsers used to browse the web) code. This license was created by a corporation and hence has greater legal language than the other licenses. It essentially allows derivative works to be licensed as MPL, but allows bundles of software that contain MPL to be licensed in other ways. It discusses patent clauses and also disclaims any warranties. The MPL license was used to create several other licenses such as the IBM license, the Common Public License and the Eclipse license. In each case the license was modified to address patent concerns that corporations would have regarding the software they were building and the kind of community they wanted to enable around it.

The important point to note about all these licenses is that they permit free use and modification as long as the code remains within the organization and is not re-distributed. This gives the freedom to corporations to deploy a particular software as widely as they want (without having to count the number of users who will use the software simultaneously or

serially or have to maintain audit trails of usage), and to make whatever modifications they want to suit their own conditions, without having to compromise their competitive position. Also, these licenses enable the 'service' model of business to flourish, where vendors can configure a particular free software package for a client, for a fee, and not be bound to publish their changes, as they are not distributing the software.

Some care is required while licensing products that have multiple contributors, in particular when the contributors are from different corporations. The project management would require maintaining documentation on the contributions and contributors, obtaining warranties that the contributor has rights to assign the license (perhaps including the contributor's company) and warranties that the contribution does not infringe patents.

2 FOSS Applications in Business

FOSS has diverse applications in business. From basic word processing and spreadsheet type computations to providing highly scalable and mission-critical database services, the applications that FOSS products are put to are as diverse as computing needs in business. One can safely say, at this point, that there is a FOSS application for every type of business computing environment. A few examples of the variety of uses for which FOSS applications exist are as follows.

One of the most important databases in the world is now a FOSS product called MySQL (a case study of MySQL is provided below). This database is used very widely by e-commerce firms that run their web-based business off this database. The database is used to track customer activity, sales records, hold product information, track transactions, inter-connect different business activities and also act as a background data storage for other business software or in embedded systems. MySQL is used by companies such as Yahoo, PriceGrabber.com, Adobe, BBC

News, Alcatel-Lucent, Dell, Intel and Sony.

Many companies have adopted the Apache web server as the basis for delivering web applications. The Apache server is one of the most successful FOSS products and, in fact, is considered to be responsible for the widespread growth of the web. It is responsible for serving, or providing, the largest number of pages on the Internet and currently (Dec 2007) is said to serve about 50% of all the pages. Apache is included as a web server with many well known products such as Oracle's database, IBM's Websphere and Borland's Kylix.

Enterprise Resource Planning (ERP) and Customer Relationship Management (CRM) tools are some of the most sophisticated softwares that are available for businesses nowadays. ERPs are widely adopted by companies in many industries and SAP, a proprietary software, remains the market leader in this category. ERP software are highly complex in nature, bringing together many business applications such as manufacturing, finance, logistics, marketing etc under one common platform. Two FOSS products, Compiere and OpenBravo, compete strongly in this market. Compiere is written in Java (under the Mozilla license) and has a strong presence in the small and medium business market. Compiere also has a CRM component that is now giving it a strong advantage. Other FOSS ERP software include ERP5, GNU Enterprise and SQL Ledger.

There is a larger offering of CRM software in FOSS and these include SugarCRM, Splendid CRM, Centric CRM, and Daffodil CRM. These packages allow companies to manage their sales force, maintain customer profiles, and design marketing campaigns based on recorded histories. For example, the LibrePOS package allows point-of-sale transactions to be recorded on a variety of devices. This FOSS tool is very popular and over 100,000 users have downloaded the software since its initiation in 2005.

The Open Office suite of desktop applications are quite popular business applications. This

suite is a FOSS product that is very similar to the Microsoft Office suite. Although Microsoft's desktop applications still retain over 90% of the business market share, Open Office has been able to make inroads in small and medium businesses where cost of the applications has to be limited. Open Office is bundled with most Linux desktop distributions and includes applications such as a word processor, a spreadsheet, a presentation manager, a database, and a drawing editor. Open Office enjoys extensive support from the developer community and has steadily increased its features and installed base.

3 FOSS Business Models

3.1 Overview of FOSS in Business

FOSS is very widely used in businesses now. It is very well known that large corporations such as IBM and HP and Oracle generate multi-billion dollar revenues from their businesses that are based on FOSS. It is also well known that there are many companies such as Amazon and Google that deploy FOSS extensively throughout their organization and leverage it for significant business advantage.

Broadly, there are three strategies that businesses follow within the realm of FOSS and these are reflective of the larger strategies employed by companies in the software business. The strategies may be classified as service, platform and product strategies. (See Figure 1.) Each of these business strategies arise from the peculiar innovations that FOSS permits and is bound by.

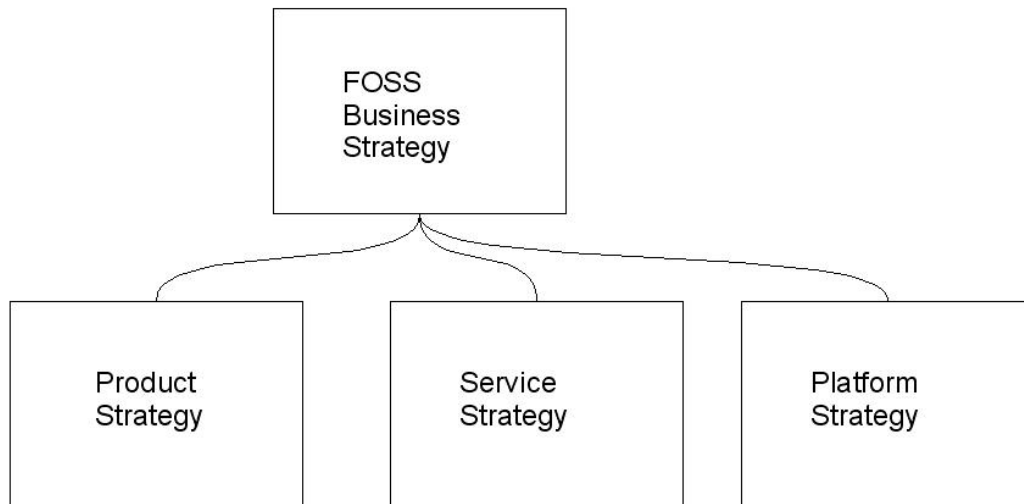


Figure 1: FOSS Business Strategies

The service strategy is deployed by companies that base their revenue model on services provided to support a FOS software. The service may be provided in the form of a periodic subscription that the company offers to its clients or it may be in the form of contracts offered for specific projects. The service is usually for support and maintenance of a FOSS product the client has adopted. Such businesses typically rely on employees who have strong skills in FOSS technology. Extensions, in terms of new features, may be made to the FOSS product and the source code for these is then given back to the community (based on the requirements of the different licenses).

Some companies adopt a FOSS product that they support the development of, while at the same time maintaining its presence in the open community for adoption and extension. Typically

the product is a 'platform' for developing other products that have direct use. Its value to the 'parent' company is in enabling other applications and products to be developed using the FOSS platform product that prove to be revenue generators. This is referred to as the platform strategy.

The product strategy involves building products that are entirely in the FOSS domain or are built up on FOSS. These products may have dual licenses, such as one for free distribution and the other for proprietary distribution, or may use only a single FOSS license. The company bases its revenue model on continuing enhancements to the product and drawing revenues from licenses and support. The core strength of the company remains in building up the product and its features as it may have a strong competitive advantage in this domain.

Within the framework of the three broad strategies outlined, many companies have evolved several innovations that give them a unique business model based on which they are able to carve out a competitive position.

3.2 Competitive Strategy

Companies that adopt or use FOSS often have to adopt a competitive position that determines the effectiveness of their strategy. This competitive position has to do with how they respond to other players in the market who are operating in their space of products or services. A stronger competitive position will enable them a better market share and growth possibilities, whereas a weak competitive position will reduce their market share and their revenues.

Businesses assume one of two classic competitive positions in any market, either as a **low cost** producer where they provide a product or service which is cheaper than their competitors' and thus more attractive, or as a **differentiator**, where they provide a different product to their

competition and compete for market share. What determines their success in the competition is their ability to deal with partners and customers and their ability to erect barriers to new competition. Most companies have to have effective **strategies to bargain** with their suppliers and their customers. This includes the ability to set prices with vendors, to set delivery times, service levels etc. With customers it is the ability to set service levels, delivery schedules, pricing modes etc. If any company is weak in its bargaining position then their ability to remain competitive is threatened. Companies have to remove **barriers to entry**, such as new technology or high investment levels used by competitors, to enter new markets. This condition is essential for new players to enter a market and take market share. If a company is already a player in a competitive market, then it has to set up barriers to entry for other players.

In information technology businesses, such as those in FOSS, the ability to create **lock-in** and to effect **network externalities** also has a strategic impact. A lock-in is created when customers or users of your product are somehow motivated to stay with your product and not move away. Moving away, or switching, incurs a cost that gets them locked-in to your product. Lock-in can be effected by a design that enables it. For instance, document formats create lock-ins, specially when they are designed in such a manner as to allow only a particular software to use them and not others. Users are then compelled to use that particular software package, as shifting to another would incur extra work or purchase of additional software. Externalities refer to those effects that occur when many people share a particular technology and contribute to the advantage of everyone by being in the network. For instance, those using email benefit more if more people start using email rather than if they are the only ones doing so. Thus, email benefits from network effects. Network effects also contribute to lock-in, because the larger the number of people using the software, the easier it is for others to stay in the network (for sharing files

and data easily) than it is to move out.

In the following section we consider a number of case studies of FOSS businesses and see examples of specific competitive strategies that the companies have adopted.

4 Case Studies

4.1 Brief Case Studies

MySQL

MySQL is a company that makes and distributes an open source database product called *MySQL*. It was started by a team of four developers in Sweden following the “pure” open source strategy of first building a community of developers who were interested in the product and gave their effort in building it up to be a highly efficient and reliable database product. MySQL was first released in 1996, and was available only for Linux platforms. In the initial phases MySQL, as a company, charged a fee for providing support and specialized extensions to the product that could be freely downloaded by anyone. MySQL was rapidly adopted by many developers who were interested in developing applications that required a fast, reliable, and relatively fuss-free database that was available for free. MySQL, by the early 2000s, became a part of the well-know LAMP (Linux, Apache, MySQL, and PHP or sometimes Perl) software bundle or solution stack, which was widely adopted by businesses wanting dynamic web pages that ran on free software, that could be configured easily and that could be deployed without any licensing problems. LAMP was bundled with practically all the Linux distributions and became readily accessible to those who wanted to experiment with setting up a web-based business or service.

MySQL currently ranks with the biggest database players in the software industry. If

measured on installed base, which is a count of the number of installations of the database there are worldwide, MySQL ranks third with 33% of the market share, following Microsoft and Oracle, and is ahead of the long time leader in databases - IBM. MySQL though ranks very poorly in terms of revenue where it had a gross revenue of \$40 million in 2005, whereas the total industry was sized at \$15 billion.

The competitive strategy that MySQL follows is that of being a low-cost provider that is also able to supply niche and customized solutions. In this sense it competes with large players like Microsoft who have a low cost product also. MySQL's advantage lies in its product being practically free to acquire and install, for testing, something many startups and small businesses are prone to do; and then clients tend to stay with it owing to the lock-in that is created by their familiarity with the product. MySQL as a product has much fewer features than products like Oracle, however its strength is its performance and scalability, and its ability to be configured for particular needs, something that is ideal for small and medium businesses.

MySQL follows a dual licensing strategy that creates a strong competitive position for it. It allows anybody to download and use the community developed and GPL-licensed version, whereas it restricts the distribution of another version of MySQL that is under a proprietary license and that is provided on a subscription basis. MySQL obtains most of its revenues from the latter version, by providing direct customizations for clients. This dual licensing strategy acts as a strong barrier to entry for competitors also working in FOSS to set up a business model imitating that of MySQL, while using the same product. Since MySQL holds the copyright to the original code it can license it in multiple ways, whereas others cannot. The proprietary license also creates a lock-in for its customers who now rely on MySQL to provide all support and customization.

MySQL's rapid adoption and growth in the industry has led users to have more faith in its offering and also in its commercial credibility. Already a profitable and growing company, it has recently been acquired by Sun Microsystems.

Mozilla

One of the first browsers that popularized the Internet in the early 90s was Netscape. Based on code created at the University of Illinois, Netscape became a very successful product and the company became one of the early successes of the "dot com" era. However, Microsoft soon came out with its own browser, Internet Explorer (IE), which became a strong competitor to Netscape and eventually became the dominant browser as it was bundled with all the Windows operating systems (illegally, as was later shown in a court ruling) shipped with desktop and laptop computers. Windows' dominant presence in the operating system market enabled IE's success too.

Netscape was bought out by AOL corporation, and then its code was made open source in 1998 and came to be known as Mozilla. This move was observed carefully by competitors as it was unusual: a closed-source and proprietary product was opened up to the free/open source community for the first time. The Mozilla project continued with professional developers also working on building and enhancing the browser. Mozilla was later spun off by AOL as a foundation (a not-for-profit institution) that was backed by several companies. Mozilla's goal was to develop a high quality browser, as by now Microsoft had the monopoly share of the market and had stopped development on IE. In 2002, after several intervening products, Mozilla released its first lightweight browser called Firefox.

Firefox's release was preceded by a volunteer donation drive by Mozilla, where users and supporters contributed to releasing an expensive two-page advertisement in the *New York Times*.

The response was immense and millions of people downloaded the FOSS browser. It had an added attraction to users, besides its excellent user experience, which was that it allowed users to configure the interface according to their needs, which many did and also submitted these as enhancements to the source. Firefox's popularity grew immensely (also because it was platform independent, allowing Windows users to use it too), and it now commands a significant share of the browser market.

Mozilla maintains a differentiation strategy. Since both Mozilla and IE have zero price, Mozilla is used and preferred by users owing to its different user interface and features. The Mozilla community has created a large number of add-ons that enhance the feature set and performance of the browser. These plugins create a lock-in for Mozilla and also allow a revenue model for its partners (mainly through advertisements).

Red Flag Software Co.

Following the mandate of the Chinese government to avoid proprietary software for which they would have to pay out massive royalties, the Red Flag Software company set out to ship Linux and Linux-based computers in China. Beginning in 2000, the company soon became one of the world's largest suppliers of desktop personal computers installed with Linux (their website is at <http://www.redflag-linux.com/eindex.html>). Within China, Red Flag held about 56% of the market share (in 2006) for Linux shipments. The company ships its own distribution of the Linux OS called Red Flag Linux Desktop and in partnership with firms in Japan (Miracle Linux) and Korea (Haansoft) it has developed a uniquely Asian flavour of Linux called AsiaNux.

Red Flag's product originated in the work done by scientists at the Chinese Academy of Sciences' (CAS) Software Research Institute. Since the inception of Linux, they had worked with the worldwide development of the kernel and simultaneously developed their own version,

which was mature by the late 90s. When Red Flag was founded, with financial backing from a government supported venture capital fund, several scientists joined the new company from the CAS. Managers who had been working with multi-national companies such as HP and Sun were hired for Red Flag. As the company grew, it hired both fresh graduates from colleges and trained them in-house on Linux, as also experienced managers who could sell the product within the Chinese government and industry. Red Flag currently has 120 employees, is headquartered in Beijing and has offices in several cities.

Red Flag's main product, an operating system, is sold in several market segments. Its main revenue earner (60%) is the server operating system that is used by enterprises, including government organizations, for applications such as databases, email etc. Its desktop operating system competes with Windows and ships directly on computers of manufacturers such as Lenovo. In this sense, Red Flag directly competes with Red Hat Linux and Novell (who sell SUSE Linux). Red Flag also produces an operating system for embedded applications, another for clustered computers, and several other advanced servers.

Red Flag currently has a very strong presence in government departments and public sector organizations (sometimes as high as 95% of the install base). It includes in its government client base agencies such as the China Post, the National Foreign Exchange Management Bureau, Custom General Office, Digital Library Project of Universities and the National Development and Reform Commission, amongst others.

Red Flag's competitive strategy is different in different markets. In the desktop market it was the cheaper alternative to Windows. It could sell to enterprise and government customers who were looking for a cheaper but equally good alternative to Windows, as also an operating system that could handle all their application needs. In the home computing market Red Flag had a

minor share, primarily due to the widespread availability of pirated Windows. Customers preferred Windows as it had better support for multi-media and gaming software, and so they would either buy computers with Windows installed (and pay for it) or buy a computer with Red Flag on it and then install a pirated version of Windows.

In the server market Red Hat has adopted a niche strategy (of differentiation). It provides customized solutions to enterprise clients and competes with players such as Red Hat, Novell, Sun and Microsoft. Its advantage lies in local support and somewhat lower costs. It does have an advantage as a 'national' product, however that advantage is not strong, as enterprise customers demand performance over the origins of the product. So it has to compete with the best offerings from around the world. One tactic that Red Hat has adopted to fight this competition is to make feature adjustments, such as the ability to install and upgrade packages, that are suited to local conditions but based on the ideas of leaders such as Red Hat.

Eclipse

The Eclipse case is unusual in that it was developed as a proprietary source product by one company, IBM, and then released as an Open Source product to the community. It had no direct commercial implications for IBM but served to build an 'ecosystem' that it could leverage for its other products. This is an example of a platform strategy.

IBM created the Eclipse platform in response to an internal need to consolidate a welter of their own software products that had been developed independently, had their own developers and markets, and had their own development styles and look and feel. In a competitive space in which platform products from Microsoft with its .NET and Sun with its Java product were looking strong, IBM wanted to have a competing product that in addition to taking market share from competitors also provided a means to consolidate its own diversity. IBM had made a

strategic choice in adopting the Java tool for many of its products, but was unhappy with Java's graphical user interface features as available in the Swing toolkit (the Swing toolkit is included with the Java programming package and is meant for developing graphical user interfaces). IBM's developers thought that Swing was slow and did not conform, in terms of look and feel, to the native operating system it ran on. The developers thus created a graphics toolkit of their own called Software Widgets Toolkit (SWT), also built to work with Java but with an additional feature that it was built separately, and ran much faster than Swing. In order to deploy the SWT, IBM's engineers created the Eclipse environment, which was initially used to build Java and SWT based applications.

Eclipse performed very well, but IBM now had the problem of motivating its own diverse divisions to use this platform, as well as enable a community of others to do so. IBM then decided to open the source code of Eclipse and create a consortium of companies that would use and maintain the platform. IBM saw this as an advantage as it could now build a platform that conformed to users' needs, as well as get its own employees to contribute to and adopt it. When IBM released the source in 2000, the response was immense and there were a million downloads of the product within a short span of time. Consortium members welcomed the product and suggested several improvements that were rapidly implemented. IBM was then able to induct a larger number of members in the Eclipse consortium.

The Eclipse product represents a platform strategy for FOSS business. IBM directly does not realize any revenues from Eclipse, although it does incur costs for maintaining it. IBM's strategy is to build other products using Eclipse that generate revenues for it. Thus IBM is keen to continue supporting and advancing Eclipse along with help from its consortium members, who also use Eclipse for developing their own products, while IBM's gain from this is a standard

platform that helps it consolidate its own diverse products, as well have access to a constantly updated and industry standard platform.

Optaros

Optaros Inc has a unique business model in which it takes FOSS products and integrates them for customers. Its specialty is in collaborative and social networking type of products, widely referred to as 'Web 2.0.' As such it is an international consulting and services company that uses FOSS products such as Ajax, JBoss etc and configures them for its customers, and in the process also builds new code that it may contribute back to the community based on the license requirements.

Optaros was founded in 2004 and has 130 employees in 9 offices in North America and Europe. By 2007 it had completed over 190 projects for 75 unique clients and had completed 92% of them on-time and on-budget. In 2007 Optaros had a market capitalization of \$20 million. Optaros has clients such as ABB, the city of Zurich, the New York Times, Swiss Hospitality Services and the State of Geneva, amongst others. Optaros also partners with a set of select FOSS product companies such as MySQL, Novell, JBoss, and others that provide it with a deeper knowledge and understanding of their products, the product road maps, and skills.

The Optaros business proposition is to shift the mindset of customers from 'buying' FOSS solutions to 'assembling' FOSS solutions. The difference is that the former mindset emphasis lower cost as a primary advantage of FOSS, whereas the latter emphasizes speed of implementation, flexibility and control of integration, and standardization. Optaros is able to achieve higher speeds by relying on off-the-site downloads of software that can be used for integration instantly, as opposed to waiting for software to be acquired (via a request-for-proposal process). Flexibility is achieved by adopting products based on open standards, whose

future growth and roadmaps are publicly available and which can also be influenced by Optaros. Standards-based products also prevent vendor lock-in and allow clients to define the time and manner in which they would want to upgrade their software products.

Optaros' strategy is an example of a 'pure play' service strategy, where it takes FOSS components and products and re-assembles them for its clients, maintaining a distinct product management advantage over its rivals. Optaros has deep strengths in its skill-base, where employees are very knowledgeable about the FOSS products and are able to deploy them in new and unique situations rapidly. This strength of Optaros becomes an entry barrier to other players. Optaros' business strategy seems easy to imitate: download FOSS products that are freely available and configure them for clients. However, this simplicity hides the depth of knowledge required to deal with complex products such as JBoss, Ajax and others that Optaros has mastered.

Google

Google is one of the most popular search engines on the web. It is backed by the company of the same name and that is now one of the largest ecommerce companies in the world. Google is well known for using FOSS products, particularly Linux on its massive collections of servers that provide the search facility to millions of users world-wide.

Google's open source product OpenSocial provides developers with a set of tools that can quickly build a social networking site or can be used inside a site like MySpace, to add new features. Google has made the entire source code available for developers to download and also provides space on its own servers for the networking site to be hosted if it is required. Google won't make any revenues directly from this product but will, most likely, rely on directing traffic from these sites to its current revenue generating sites.

Google has also launched the Android toolkit that enables developers to build an entire application set for any mobile phone. This 'Open Handset Alliance' is an alliance of 30 companies that will participate in developing the toolkit. The toolkit has been launched under the Apache license that allows developers to make proprietary extensions (and not release them as FOSS). This has the advantage that some developers can adopt the toolkit and build their own market-specific applications for mobile devices. Google again does not expect to derive direct revenues from this product, however, it is possible that future extensions will allow it to build in advertisement capabilities that is its main revenue source.

Google is deploying a platform strategy in creating these FOSS products and releasing them to the community. As with the case of Eclipse, Google is using its vast resources and talent pool to build complex products that its community can use. This gives Google the flexibility and community support to launch other revenue generating services.

Network Theory Limited

Network Theory Ltd (NTL) is a publisher of books on free software. They publish manuals on free software, in book form, and also provide the manuals online in digital form. NTL accepts already prepared software manuals and provides additional services such as proofreading, error correcting, example checking and correcting, indexing etc. These corrections are sent back to the authors for improvement.

NTL realizes revenues by selling paper versions of the books and manuals. The books are sold through regular outlets such as Amazon.com. NTL uses the revenues to hire and train staff to perform editing and proof-reading tasks.

NTL's business philosophy reflects the open ideas of FOSS. They want to allow manual users to be able to change and modify manuals, according to their own needs, as opposed to having

one that is frozen and unmodifiable. With the 'source' of the text available in digital form, users of the software can make modifications to the software and then correspondingly to the manual also.

NTL differs from traditional businesses in that it does not retain its profits. It donates the surpluses it generates to the FOSS community, in particular to projects it has published manuals of. They also publish their contributions to the FOSS community on their website.

4.2 Extended Case Studies

Oracle Unbreakable Linux

Oracle is one of the largest software companies in the world with annual revenues of about \$20 billion. Its main product is the Oracle database that has a dominant market share and is widely used by large and small enterprises around the globe. Oracle also has a large portfolio of products that it sells and services in addition to the database and these include middleware, server software, analytics software, and database management software. All these products are proprietary in nature and licensing.

Oracle's FOSS offering is the Unbreakable Linux (UBL) service. It is a service that Oracle offers to its enterprise clients that includes support for many FOSS products such as Linux, including Linux from other vendors such as Red Hat and Novell, the Apache web server, other FOSS databases and applications. The service includes maintenance of FOSS products, adding features, deployment and testing. The clients are mainly large enterprise customers who have adopted other Oracle products such as the database and are running them on a Linux platform, or are using a FOSS server or application along with the Oracle products. Such customers seek support that is comprehensive and covers not only Oracle products but also the complementary

FOSS products. After its initiation, the Oracle Unbreakable Linux offering has already registered over 1500 customers that include companies and organizations such as Yahoo, Dell, Ambercrombie and Fitch, the city of Las Vegas, Stanford University, and the Beaumont Hospitals.

Oracle has recently launched its Unbreakable Linux program in the Asia Pacific region and has hired Shane Owenby as its Senior Director for Linux and Open Source. In an interview Owenby said that the main reason Oracle introduced the program is because customers demanded higher quality enterprise-focused support.

“Oracle Unbreakable Linux was created to give enterprises the comfort to adopt and increase their usage of Linux. Some customers felt the move to Linux was risky as they didn't believe the financial stability of Red Hat or Novell was sufficient to trust the strategic long term move of their systems to Linux. When Oracle launched UBL then that issue went away. Even though these companies are doing well in the market, some customers were taking a very long term view of support. Suppose they deploy a 10g RAC cluster with Linux and are running Oracle on it, and they are sure of Oracle's presence, but what about the Linux component? A number of customers approached Oracle with that concern that led Oracle into that market.” [10g is an Oracle database product that can be deployed on a cluster of servers. These RAC (Real Applications Clusters) clusters can deliver very high performance, up to 10 gigabits per second of processing capabilities.]

Oracle's focus is on large, enterprise customers, who have complex requirements and for whom the solution set offered by Oracle is a natural choice. Oracle also has a fairly large presence in the small and medium enterprise market (SME), where they support Linux and other

FOSS applications also. The important aspect, as Owenby points out, is that Oracle is providing support through its UBL program, where customers could demand support for Linux, or the Apache web server or some other application.

Oracle also has its own Linux distribution, called Oracle Enterprise Linux, that it provides as a free download through its web site. This distribution is based on Red Hat Enterprise Linux. Many customers have now opted for this distribution as it works well with other Oracle products and is in turn fully supported by Oracle.

Owenby is of the view that the main competitors for UBL are not the Linux vendors such as Red Hat or Novell, who also provide support for Linux and related applications, but the companies that provide and support an entire stack (a stack, for example, could consist of an operating system running on hardware, an enterprise application like a database, some middleware to support diverse clients, and some applications running on top of the database). Microsoft and the (high-priced) Unix hardware providers (such as HP and Sun) would be such companies as they cater to the large enterprise market and also support and maintain such complex solutions. The competitive advantage that Oracle has over its rivals is that of their customers preferring Linux as an operating system and then running other applications on this. With UBL, Oracle can attend specifically to this large and growing market segment.

UBL also contributes code, in terms of features and bug fixes back to the FOSS community.

To quote Owenby again:

“We contribute in two different ways. One, anything that we fix we automatically give to upstream. Let’s say we put in a kernel fix or a glibc fix or a gcc fix or Apache fix. We send that straight to the upstream provider for that package which could be the Apache foundation or Linus’ team of kernel programmers. Second, we also open a bug

report to Red Hat and give them that contribution. So we make sure, since our distribution is based on RH, that we give any fixes back to them. So we are abiding completely by the Open Source community's rules.

“What type of contributions do we make? It is a very long list. Asynchronous I/O is something that Oracle added to the Linux kernel a number of years ago, and it has been maintaining that component. Asynchronous I/O makes databases run faster.

Another thing we have contributed is a clustered file system, OCFS2 (Oracle clustered file system v 2). That was accepted upstream in the 2.6 kernel stream. We have been making contributions all along.

“We recently made a contribution to PHP. We added code to PHP that now allows PHP to run pools of Oracle databases, something that has never been done before. We contributed that code to the PHP project.”

Oracle has a long history with Linux and FOSS products. In the 90s, Oracle had conceptualized a network computer that was based on Linux. It had set up a Linux engineering team for that purpose. Over the years Oracle has developed considerable internal expertise in Linux, and much of its infrastructure and development is done on Linux. Oracle realized that it could offer this internal expertise in the market to its customers, however, it had to be careful not to jeopardize its working partnerships with Red Hat and Novell. Later, this restriction was not relevant and it went ahead with its UBL offering.

Oracle has acquired InnoDB, a database storage engine that operates with MySQL and is distributed with the same. InnoDB helps with processing database transactions such as compression, storage of data and indexes, recovery from crashes, and disk management. InnoDB too follows the dual-licensing policy, where one product is released under GPL, whereas other

versions can be released as proprietary and customized licenses for customers.

Oracle has also acquired BerkeleyDB, another FOSS database. BerkeleyDB was developed by Sleepycat Software, which was bought by Oracle. BerkeleyDB's specialty is that it can handle a very large number of simultaneous queries. It was originally developed as an embedded database for the Netscape browser.

Oracle's strategy in acquiring these powerful database products that were not competing in its own markets was to create a presence in the FOSS database space. Industry analysts say that with Oracle's backing these products have grown further, have expanded the total database market and in the event increased Oracle's market share. FOSS databases have grown the market by capturing those segments where traditional databases were not used in the past.

Oracle corporation has over 69,000 employees worldwide. Within this, there are a very large number of employees who are savvy with FOSS technologies. It is this strength that gives Oracle a strong competitive advantage and also acts as a barrier to entry for newcomers. Oracle has invested heavily in engineers who have the required skill sets in FOSS, and these engineers are scattered across the globe. Many of its current employees are those who have made strong contributions to FOSS and are well known in the community. As Oracle employees, these engineers continue to make contributions to the FOSS community.

Owenby states that FOSS products in themselves don't create any specific value proposition for his customers. It all depends on the customer's requirements. Some customers don't want their engineers to see the source code, because they are afraid that it could be modified and put in production and remain unsupported. "And some customers feel great about it, because if by some slim chance Oracle and RH go out of business, they can then hire their own engineers to support the product."

Customers see their requirements in their own environment and then make a choice for FOSS or proprietary products (and Oracle has both types). For example, Owenby states, some government customers have seen the potential in FOSS products and are writing policy to adopt the same. Their decision is based more on the future prospects of FOSS rather than on the product itself. Some insist on enterprise-quality support for any product.

Oracle does not have a particular FOSS licensing strategy as “there is no one size fits all approach.” Oracle evaluates the license of each project and if it is to make contributions to that project then those contributions must be under the same license the project uses. Oracle evaluates each FOSS project to see if it should adopt, integrate and contribute, develop and distribute, or offer enterprise support for that project

UBL also has partnerships with various vendors of software and hardware that enable Oracle to maintain a continuous presence in diverse markets. Most of their customers buy hardware from the biggest manufacturers such as IBM, HP, Dell, EMC, Hitachi etc. Oracle partners with them to ensure that drivers for these devices are built-in to the various Linux distributions and also that they are usable with Oracle’s products. With a complementary set of products, that are based on open standards, the market is improved for all concerned.

Intel

Intel Corporation is the world’s largest producer of semiconductor devices, also known as the chips that go into computers. Intel has a wide range of products that are used by computer and other manufacturers. Intel’s current annual revenue is about \$37 billion. Although Intel’s focus is on hardware and the core components of computers, it nevertheless takes a strategic interest in the development of software and applications, and partnering with such companies, in order to grow and sustain the entire ‘ecosystem’ on which its own products thrive.

Intel also has an initiative in the FOSS space. It is well known that Linux was initially developed to run on Intel processors. Intel has actively supported this effort, despite being a partner with Microsoft for their Windows products which run on Intel microprocessors.

According to Valsa Williams, Intel's Asia Strategy Manager - Linux and Open Source, "Intel's focus is to ensure that all software runs best on Intel processor-based platforms. The growth of open source models has resulted in growing significant customer usage. Intel has been a proponent of open standards and involved in open source for several years. Intel addressed the demand for Linux and OSS by working with the ecosystem and the community to foster innovative products and solutions."

Intel's competitive strategy is to strengthen its position within its ecosystem. As Williams stated, "contributing to key open source software, we can build value for Intel technologies. This benefits Intel and benefits the industry at large in helping advance new technologies. For example, we are amongst the top corporate contributors to Linux and Xen. With Xen, there are several vendor products in the market today from Red Hat, Novell, Citrix, Oracle, Sun... the entire ecosystem is benefiting." [Note: Xen is a FOSS virtualization software. It enables a single hardware chip to host multiple operating systems (say Linux and Microsoft Windows) simultaneously. Virtualization has found wide appeal as it enables organizations to consolidate their many servers onto a single chip or a cluster of chips, which helps with managing them.]

Intel has also initiated two major projects this year, LessWatts.org, to drive Power Savings Software Technologies, and Moblin.org, to drive software innovations around mobile Internet and consumer devices. Intel expects the industry to participate in these new opportunities and it is already seeing a great response. Lesswatts.org is a consortium of companies, users, developers and system administrators that is creating Linux-based software that will drive down power

consumption in computers. The web site provides software as well as tips on how to reduce power consumption of all types of computers, from servers to laptops.

DeepRoot Linux

DeepRoot Linux was setup in August 2000, in Bangalore, with the primary goal of working on Free Software full-time, of providing commercial services for Free Software and for building FOSS products.

DeepRoot's goal was to build affordable products powered by FOSS, sell them as hardware and software appliances and build a service business based on that. Says Abhas Abhinav, DeepRoot's founder and *i-take-charge*, "It was actually quite difficult to think about successful, large-scale Free Software products at that point in time. But we didn't know that! "

They spent four years developing their own hardware and software server appliances. They wanted to simplify server deployment and use (right from the hardware level), assuming that was where the market lay. They even manufactured their own server chassis and LCD panels and that received a good response from the market.

Over eight years, they'd built a variety of products - a network-attached data storage product, a bandwidth management solution for cable Internet service providers and enterprises, a general-purpose office server, etc, all based on FOSS. They were able to evolve some of these as polished products and release them - while others just could not be productized.

DeepRoot experienced the classic problem of technology startups: they worked very hard to build products but not as hard to build a sustainable business model based on those products. Admits Abhas: "We were all relatively inexperienced with how to manage a business. We just wanted to write code and develop products and really learnt the business (and project management) aspect the hard way."

“Today we realize that the Free Software part and the business are completely disjoint and distinct. The challenges of building a Free Software business remain similar to those that you might encounter while building a business of any other sort. If you can manage the business and management aspects well enough, the Free Software part is easier to take care of. Free Software is the solution and it must be employed to solve problems for people - not always an end in itself.”

What gave DeepRoot the push in the early years was the faith a lot of clients put in their business model. Some even gave them their extra hardware that they had lying around.

The most consistent challenge they faced with developing the business was that of getting passionate and motivated people to work with them. Abhas states that he had assumed that committed developers would want nothing more than to build free software full time, however that assumption didn't pan out and he had to struggle to build a team. Ironically, even today, when they have enough business, the challenge is to find enough people to meet the demand.

DeepRoot has a 15-member team and currently includes amongst its clients companies such as Air Deccan (an airline), TTK Healthcare Services, Arvind Exports, Manipal Group (a group of hospitals), and Fabmall (a chain of grocery/general stores). Their current products and services offerings include: 1) Messaging solutions which integrate and serve email, fax, instant messaging; 2) firewalls and proxy servers; 3) thin client terminal servers; 4) CRM solutions for helpdesks and customer care; 5) custom software development; 5) training for Free Software (system admin, desktop, developer).

Since they operate in many horizontal and vertical markets, their overall competition is complex. Abhas includes amongst his competitors various categories of players, corresponding to the models of FOSS businesses:

- Proprietary software product companies (with no FOSS involvement): these would include companies such as Microsoft, MDAemon, or Communigate Pro (a company that creates a suite of products for consolidating messaging across many media like text, mobile, groupware etc).
- Proprietary software product companies (that build products using or over FOSS): these products, like Communigate or Postmaster (a corporate email product), may be built using Free Software tools. While some of these products are completely proprietary, there are others like Zimbra (an email server product) and SugarCRM (a CRM product) that are built partly as Free Software and utilize a whole lot of Free Software infrastructure to operate. These companies may utilize the best FOSS products while continuing to retain the licensing and business model of proprietary software products.
- True FOSS product companies without a service business: these companies build and support products that exist as true FOSS offerings on the Internet but don't have supporting services offerings from their developers. People who use them are on their own for support. Says Abhas: "Such products are one part of what we count as best true competition for us. They follow Free Software principles in both spirit and letter and provide a very convincing alternative to proprietary software products."
- True FOSS product companies with a service business: just like the previous category, these products and companies are direct and true product competition. Many such companies (such as Red Hat, Novell etc) focus mainly on the very high end of the market. Since DeepRoot's focus is on the SME end of the market they don't always directly compete with these companies.
- Hosted software application providers: companies like Google and Zoho appeal very

favorably to small and mid-sized organizations. Most of this software is built using FOSS but is never distributed and hence is not subject to FOSS licenses. Their business models are built out of charging organizations in a per-user manner for access and value-addition to their software.

- Services companies with FOSS as an option: there are many companies that specialize in integrating disparate products for customers and getting those products to work for them. They don't focus on FOSS but have it as an available option that customers can access.
- Services companies with exclusive FOSS focus: these companies really believe in FOSS and understand that one of the best ways of promoting FOSS is to build a services business on it.

Deeproot's competitors' use various means by which to increase their market share, not all of which DeepRoot can replicate. These include: 1) well-funded promotions and strong brands; 2) distributing free (limited-functionality) versions of proprietary software; 3) strong marketing and sales channels; and 4) providing heavy margins to resellers.

DeepRoot is addressing the competition by some moves of its own: they actively participate in educational and awareness activities to make users more aware about the value of FOSS and software freedom. They try to get users "addicted" to FOSS products and solutions so that users have an expressed preference for FOSS (and eliminate proprietary products from their radar). They extend the state-of-the-art with respect to software usability and features. This is appreciated by users, and helps DeepRoot compete against other free softwares. They have created and continue to sustain a strong service base for their products and for other FOSS products. This is crucial as more and more organizations are seeking reliable support. They make strong attempts to establish their technical skills and superiority in servicing, extending and

supporting FOSS. They have realized that eliminating reasonable doubt about their abilities amongst clients enables more of them to adopt FOSS and DeepRoot's services.

DeepRoot feels that the market for FOSS in India, and in particular in Bangalore, is quite large. Often they get clients who are not able to judge their competence because they haven't seen other companies of a similar nature. There are very few metrics available to compare prices, service offerings and relevance. Customers are often forced to compare them to proprietary software companies.

By impressing upon customers their ability to both service and build FOSS products they are able to compete against pure service vendors who typically undercut them on prices. If customers are able to differentiate between the skills DeepRoot has to offer as opposed to a pure service company, DeepRoot gains a competitive advantage. On the other hand, since DeepRoot also builds its own products, it can lower prices as required.

Abhas is clear that FOSS gives them a clear competitive advantage: it gives them the freedom to develop and distribute software without any limitations and in ways that they feel are the best to adopt. FOSS affords them the ability to build up on others' work and not have to re-invent the wheel for everything they want to do. FOSS enables them to be a part of the large community of Free Software developers and be recognized for the work that they have done.

Of course, use of FOSS entails certain responsibilities, says Abhas. These responsibilities have to do with playing fair and spreading the word about FOSS. Use of FOSS does not present any liabilities as such.

DeepRoot market their products and services through various channels that include the Internet, direct sales, the FOSS community, resellers, referrals and repeat sales. The FOSS community plays a special role in that word-of-mouth is still the most authentic form of referral

that can be generated. When members of the FOSS community (in India or abroad) talk about DeepRoot on their blogs or mailing lists, they spread the word about the company and about its products and services. This is also the best form of feedback they receive.

DeepRoot has a clear licensing strategy: they work only with GPL. They feel it is the only license that will continue to enable freedom of usage, distribution and modification for them and their clients. Recently, they have also started considering the GNU Affero GPL Version 3 as it has specific clauses that make it mandatory to disclose modifications made to hosted software (eg. web-based software).

DeepRoot's most crucial resource is skilled personnel who have the motivation and knowledge to work with FOSS. Abhas observed that they not only look for people with good technical and FOSS skills, they also seek people who can work with the FOSS community, have good leadership and communication skills, and are able to work alone and learn quickly.

Their work is appreciated most by clients and users who understand the underlying philosophy of FOSS. This varies quite starkly amongst people, as they have different ways of seeing business and software needs.

Some look at FOSS a way of spending money under "operational expenditure" instead of "capital expenditure" while others look at it as a way of saving costs. Yet another group of customers look at FOSS as a way they can get customized solutions tailor-made for them.

Most of DeepRoot's vendors don't care about FOSS - they just care about whether it can make some business sense. They don't really have any specific interest in promoting or using FOSS.

Mahiti

Mahiti is a Bangalore based organization that aims to reduce the cost and complexity of IT

through the strategic use of FOSS. It is an IT service company producing applications and related training that facilitates knowledge sharing. Mahiti actively supports companies, civil society organizations, government institutions and civic agencies. Over the last nine years Mahiti has served many organizations by building multi-platform, multi-lingual and multimedia enabled web, Intranet and kiosk applications. Mahiti conducts trainings and workshops in partnership with donor agencies and multi-lateral organizations. There are 37 engineers working at Mahiti.

Mahiti provides FOSS solutions in four parallel areas defined by the kind of applications the products are put to: Joomla, Drupal, Python, Zope, (all are FOSS products used for online content management); PHP, Plone (FOSS programming environments); Mono.net, C# (general purpose programming environments); and Java, J2EE (platform independent, general purpose programming environments).

Mahiti competes both in the FOSS market and in the proprietary market. Many of their competitors, such as Zomega, I-vista, and NettiApps, operate in the FOSS space and also outsource work to Mahiti.

Says Sreekanth S Rameshaiah, Executive Director of Mahiti, that the competition is not very steep, “They have their own set of clients and we have ours.”

About dealing with competition, he said: “We keep adapting to new technology and developments and have tried to have experts and teams in each of the technologies that are available so that we do not refuse work. We have been substantially successful in maintaining this advantage. Integration of multiple languages where PHP applications integrate with Python applications, Flash with PHP and Flash with Plone have been some of our unique implementations.”

Mahiti have developed strong competencies in Python, Zope, Plone, Mifos and Joomla,

technologies they have worked with for over five years. They have experts in these technologies who form the core team. The work they do involves developing code that they contribute back to the projects. Mahiti sees FOSS as more than a business model. Contributing back to the community is an important commitment for them.

They do not push the cost factor as the primary benefit of FOSS. Said Rameshaiah, “The flexibility that comes built in to the FOSS applications, and the ease with which the application can be customized or extended to benefit the clients are the primary USP. Also the fact that FOSS applications can be made to inter-operate with legacy applications in a more seamless manner compared to proprietary software makes a big difference. Cost benefits are also considered by some clients as a key reason.”

Mahiti’s primary marketing tool is client references. Owing to their long presence in the market they get a lot of calls. They do FOSS advocacy in different organizations and speak about their services. Another area from which they get references is the FOSS community.

Adds Rameshaiah: “We first educate people about the use of FOSS and help them in training on FOSS. Once they see the entry barrier being destroyed it is easy to market as it sells itself.”

Clients do value their business model. Sometimes it makes it a lot easier for their clients to work with Mahiti compared to proprietary vendors. However, their vendors do not care about their FOSS strategy.

Mahiti release all their code under GPL 2.0. In some cases they use LGPL for some commercial clients.

While looking for employees, says Rameshaiah: “We look for young talented programmers who are looking a little beyond greenbacks. We do not care about the academics. Our primary screening check is to see if the programmers have logical thinking abilities.”

5 FOSS User Case Studies

Suguna

The Suguna Poultry Farms Limited is one of the largest poultry companies in India, with an annual turnover of Rs 14 billion. Based in Coimbatore, in the southern state of Tamil Nadu, it has branch operations in many other states. Suguna Poultry Farms is unique in the poultry business because of its strong vertical integration, where it manages its entire value chain from contract farming of maize, which is a key feed for its birds, to managing breeder and broiler farms, feed mills, and hatcheries.

Suguna is also among the first in the industry to implement an ERP solution, the Oracle E-Business suite, on a FOSS/Linux platform. This was in response to the tremendous demand they were facing and the growth they expected in the coming years (they expected to cross Rs 40 billion in a few years). Their choice of an ERP solution was to integrate their operations spread across nine regional and 100 branch offices, 6 poultry farms, 120 breeder farms, 30 hatcheries, 10,000 broiler farms, and 25 feed mills.

Suguna's choice of the FOSS/Linux platform was based on the key consideration of security, compounded with the problems it was facing with its incumbent Windows platform, those of poor performance and virus attacks. The company was also keen to try out the 'new technology' of FOSS and given that it was rolling out an ERP that would support this product, the choice was eagerly accepted. The IT department was not very aware of FOSS and had to be trained by consultants, for a fairly steep price. However, they learnt quickly and were able to roll out the system in reasonable time.

One of the prime advantages of moving to FOSS has been the high uptimes achieved by the

various systems in the organization. Since the systems were deployed across multiple locations with data being updated and used by thousands of farmers, the guarantees of performance were important. For instance, monitoring of feeds and medication for birds is very crucial and time-critical. Suguna was able to effectively monitor and control this process by the ERP implementation.

The other important advantage of having a FOSS platform for Suguna is that of scalability. They wanted the systems to scale as fast as their growth and for this Linux was well suited. After seeing the performance of Linux, the IT managers at Suguna were also considering moving all desktops to Linux also.

Yatra

The travel portal Yatra started in India in 2007 and in less than a year's operation it has gained considerable attention and market share in the fast growing travel market in India. Yatra Online Pvt. Ltd. has 220 employees and offers package deals and customized travel offerings to individual as well as corporate clients. Its services include rail and air bookings, hotel, car and bus reservations, and leisure activity reservations, all of which can be accessed either via their website or by calling in to the customer centers (where regional language support is available). It provides information on travel and hotel bookings for 5000 locations, including cities and rural areas, across India. It also has travel-related news and advisory services available online.

Yatra was co-founded by travel industry veterans who had a very good idea of the business and the technology that they wanted. They wanted a stable platform, and did not want to experiment much, on which to base their e-commerce business with the main requirements of being robust enough to handle the growth in the demand and also secure enough to handle the large number of monetary transactions. They settled for Red Hat Enterprise Linux (RHEL) as the

basic platform (along with Microsoft SQL as their database server and OpenOffice as their desktop application) on HP blade servers. The choice was based on RHEL having a lower cost along with a high throughput performance, and being secure and reliable. Yatra also wanted to build customized applications, with their in-house IT team, for which having access to the source code was important. Another important reason for choosing RHEL was time pressure – they wanted their systems to be up and running as soon as they had made public announcements of their upcoming portal.

Yatra has a 24/7 business where about 40,000 unique visitors log onto the site daily when about 2500 tickets are traded. Yatra is able to offer a unique business proposition, to differentiate itself from the strong competition, wherein it has tied up with all the airlines operating in India as well as with 1300 hotels spread across 130 cities and towns. For transactions, Yatra not only enables credit card purchases, but also direct bank credits, ITZ cash card and Cash-on-Delivery options. These partnerships required that Yatra have an architecture that could sustain business-to-business electronic-data-interchange links with all its partners that is reliable and secure. Also, it required a platform that could be flexibly configured and scaled to the needs of different partners.

Triburg

The Triburg Apparels Buying Agents company is located in Dubai, has regional offices in Sri Lanka and India and 22 branch offices spread over Asia and the United States. Their main business is identifying apparels manufacturers located in Asia, and sourcing apparels from them to supply major brands in the USA and Europe, such as Polo, J.Crew, Banana Republic and others. Triburg deals with 150 vendor factories located in five countries and has a capacity of 1.5 million units a month. They have annual revenues of \$ 225 million and about 300 employees.

One of Triburg's most important applications is its mail service where it processes close to 30,000 messages daily. Realizing a need to update their existing Windows Exchange solution, the outsourced IT vendor to Triburg, TAAB Software, and Triburg actively considered shifting to a FOSS-based option. The advantages they saw was of robustness and reduced threat from viruses. Money or cost savings was not an issue. Upon a systematic feature-by-feature comparison of Microsoft Exchange with a Linux-based server, the core team decided on the latter as it had more of the features they wanted (many of them built-in as opposed to being add-ons).

The implementation required the outsourcing agency as well as the in-house team to learn about FOSS products and how to manage them, which they did in a training center in New Delhi. A vendor was contracted to implement the Red Hat email server solution that included adding customizations such as specialized user groups, control panels, back up processes, etc. The vendor completed the migration processing in consultation with the Triburg IT team, and the actual migration took place during one afternoon, at the peak time of processing, without any glitches.

The users and the IT team are so pleased with the FOSS deployment that they plan to migrate other functions as well as desktops to the FOSS environment.

Bajaj Allianz Life Insurance

With the deregulation of the insurance industry in India, several private players moved quickly into the business and grew rapidly. One such firm is Bajaj Allianz, a joint venture of a large two-wheeler manufacturer in India and a large insurance provider from Germany. The Bajaj Allianz Life Insurance company was formed in 2001 and since then has grown, in 2007, to have over 200,000 agents in 876 offices across India, and gross written premiums exceeding Rs 53 billion. In the same year it added about 2 million new customers and had profits of Rs 662 million, the

largest for any private insurance company in India.

In 2005 Bajaj Allianz addressed the challenge of a rapidly growing sales force, of controlling costs for upgradation and maintenance of desktops and laptops used by their agents, and of having a stable computing environment free of viruses and security threats. The IT team in the company had been experimenting with and using FOSS products and so invited Intel and Red Hat to set up an experimental lab with Linux and other FOSS desktop applications running on Intel platforms. The lab was called the Center of Excellence (COE) lab and allowed the Bajaj Allianz internal IT team to assess the performance and stability of running FOSS on desktops.

The insurance industry relies heavily on information technology for all its core operations of selling premiums, recording periodic payments, maintaining records over decades, and maintaining active customer contact. Thus, they require a stable IT infrastructure that can sustain the information management over long periods of time. With the massive growth in the industry there is also a need for evolving new products and packages, all of which have to be supported on reliable IT platforms.

Bajaj Allianz already had a significant amount of Intel-based hardware in their infrastructure when they decided to experiment with Linux. They set up the COE to be sure that their operations could be adequately supported and scaled by FOSS. When the results of experiments showed that Linux use could be of value, they rapidly adopted Linux for their most transaction intensive applications, which were based on laptops and desktops. Within a year they migrated a few thousand personal computers with plans to shift thousands more rapidly.

With this success in place they started using other FOSS products such as communication tools and databases for back-office and customer support applications.

Eveready Industries Ltd

Eveready is possibly one of the best known brands in India for having been in operation since 1905, and for the dry cell batteries and torches that are very popular. Currently, they have revenues of Rs 8 billion and have expanded their products to include packet tea and mosquito repellents. In batteries they have a market share of 56% and in torches their share is 85%, making them leaders in each category.

Eveready has a network of 10 manufacturing units, 15 sales branches, 45 warehouses, and 3000 stockists who cater to 600,000 retail outlets. This entire chain was earlier managed on a network of distributed servers and databases, which had grown organically as the company's business had grown. In 2005 the management decided to coordinate the information infrastructure and looked to FOSS as a possible platform.

The legacy infrastructure at Eveready consisted of three different operating systems, including Unix and Windows flavors, several different database products and applications developed on different platforms (Java and others). The IT management at Eveready decided to consolidate their disparate platforms and applications to not only improve management but also to improve response times at geographically dispersed locations. They chose an ERP product from Oracle and decided to host it on Linux (Red Hat) running on Intel chips. Their choice of Linux was based on prior experience with FOSS as well as the support they were assured of for both the ERP application and the basic software it ran on. Another major factor for choosing Linux was the ability in future to scale up, as needed, based on commodity hardware.

6 FOSS in Government

Governments are some of the heaviest spenders on and consumers of IT products and services in

Asia. For FOSS businesses to succeed it is imperative that they maintain a clear focus on this segment of the market.

Governments in Asia, though, have varying stands on FOSS. Some, for instance China, have a very clear pro-FOSS policy and actively encourage FOSS development. Whereas others, such as India, have unclear policies, with some internal efforts by departments to introduce FOSS.

Singapore government has been and continues to be very slow in adopting FOSS products despite having a huge IT budget and one of the most advanced eGovernment infrastructures in all of Asia. Analysts say the reason is the lack of awareness about FOSS coupled with a relative lack of FOSS businesses in the region that can bid for and supply the knowledge and the services.

The Malaysian government had the most active and carefully thought-through FOSS policy which included precise targets for training, implementation, procurement, and skilling within the population for FOSS. This policy was announced in 2005, and included a precise roadmap for the implementation of the policy. However, under pressure from various proprietary vendors, Malaysia amended its policy in late 2006 to one of 'neutrality.' It argued that it wouldn't prefer any particular kind of software, whether free or otherwise, and would allow governments and departments to choose based on 'merit.'

In South Korea, FOSS products have been deliberately introduced in public sector organizations for a few key applications. This is not widespread, however, and some of the applications have run into compatibility problems. The government has no policy for adopting FOSS widely, although there is mounting pressure to have such a policy.

Indonesia has a policy for adopting FOSS products. The policy was shaped as much to tackle piracy as to support FOSS. In 2006 the government also released its own Linux distribution,

called IGOS (Indonesia, Go Open Source), in order to boost FOSS usage in the country. At the launch the government stated that FOSS was being supported to stimulate innovation in the information technology industry. (This policy has been strongly contested by proprietary companies such as Microsoft.)

Both Cambodia and the Philippines have FOSS adoption explicitly stated in their ICT policy documents. The focus on FOSS is essentially to foster growth in IT technology in industry. The adoption of FOSS in the private sector and government departments has been quite strong in the Philippines. Sri Lanka has a strong FOSS movement, with many companies and developers supporting FOSS. But the governments policies on FOSS are unclear. The National ICT Policy of Bangladesh (2002) does not explicitly mandate use of FOSS.

The government of the state of Kerala is one of the pioneering states in India to have an explicit policy on FOSS. The policy states that FOSS will be directly encouraged for the development of e-government systems and mandates the use of open standards and formats for data storage and use. The policy is inspired by the Right to Information Act passed by the central government of India that encourages the use of information and communication technologies in government and to make available information to the public.

The Kerala government also wants to make Kerala a major FOSS destination in India. It wants to sponsor innovation and entrepreneurship development around FOSS.

Other states in India, such as Maharashtra, Tamil Nadu (see case below), West Bengal and Madhya Pradesh, have introduced FOSS applications in various government departments and in government sponsored educational institutions. However, few have created an explicit policy environment for FOSS. This reflects the central government's views, which has remained 'neutral' to FOSS, treating it as yet another technology that may be selected by those wanting to

do so.

In India, the National Knowledge Commission, an advisory body to the Prime Minister, has clearly advocated the use of FOSS for building e-government systems. Their arguments, to use FOSS, include the ability to scale operations, to build customized software, and to have durable documents based on open standards.

6.1 Adoption of FOSS in Government

For developing countries in the Asian region, policy regarding FOSS adoption is driven by many factors. Primary amongst these is the fact that most proprietary software in the world is produced in the developed nations of the west. Acquisition of this software requires remission of large amounts of capital that governments feel they can avoid with FOSS. The issue of capital outflows for developing countries is also linked with dependency on other nations. If a FOSS adoption policy prompts businesses and government departments to adopt FOSS, in the long run this would reduce the nation's dependency on other nations, hence improve their relative international standing.

FOSS adoption by government agencies and departments is based on the awareness of employees about FOSS, their perception of FOSS features and its implications, their comfort with FOSS licensing, and the benefits they can derive from FOSS adoption. Awareness includes the department's and employees' understanding of what FOSS products are and how they can deploy them for their own requirements. It entails having an understanding of how the solution will fit the problem they have.

Perceptions about FOSS include the business value of FOSS, the political implications of some products that have zero price and hence cannot be budgeted for, and the fact of buying a

service instead of a product. Many employees of governments feel that FOSS is inherently insecure, is probably not of high quality and goes unsupported. These perceptions are often based on the fact that 'free' is equated with zero price and hence low quality.

Some government employees also hint that because FOSS products are perceived to be of zero cost, purchasing departments are reluctant to deal with them, as they may cut into 'kickback' payments.

ELCOT

ELCOT is the Electronics Corporation of Tamil Nadu and is the nodal agency that provides IT procurement services for all departments of the state of Tamil Nadu in southern India. In a bold move recently, led by its Managing Director C. Umashankar, ELCOT removed Microsoft Windows from all its legacy servers and desktops and replaced them with SUSE (a Linux distribution) servers and desktops. This move affected 1880 servers and about 30,000 desktops.

ELCOT is a pioneering agency of its kind in India, having been the first to introduce a comprehensive IT Policy amongst all states in India and also being one of the first to actively consider FOSS as an alternative to proprietary software. ELCOT is responsible for supporting the IT industry in the state and partnering with other government departments with the e-government initiatives. ELCOT actively promotes FOSS and open standards.

Tamil Nadu's government-supported schools needed to upgrade their desktops and servers, which were running Windows, and the cost of upgrading to Windows Vista was felt to be too high, mainly because of the additional hardware that would have to be acquired (Vista requires much higher RAM and also secondary storage than the earlier Windows operating systems). Also, the choice of software had to account for future upgradations required for hardware, software and telecommunications infrastructure. ELCOT thus settled for SUSE Linux Enterprise

server and SUSE Linux Enterprise desktop for the schools. As a result of this move, ELCOT saved millions of rupees in licensing fees and 25-80% on hardware costs.

ELCOT effected the rollout of SUSE with its vendor, Novell. It tried out the rollout first in its own offices, migrating all its systems to the SUSE operating systems, and after this success began the rollout in the schools. Testing out on their own systems first allowed them to directly observe the challenges that they would face later. One of the advantages they saw was the clear security implications, as Linux was far less prone to viruses and malicious attacks. Within their office, the acceptance of the new desktop was infectious. People got excited about the new technology and the ease with which it could be installed. ELCOT also arranged training sessions for its employees.

For the rollout in schools, ELCOT prepared an application package suite that would be parceled with the desktops and then, with Novell's help, prepared training software for teachers that could run in Tamil also. This helped teachers learn the new software quickly and also be able to configure it to their needs. The teachers liked the ability to experiment with the vast range of softwares available within the FOSS community, without having to pay exorbitant prices for licenses.

The success of the rollout in schools prompted ELCOT to promote this amongst government organizations also.

7 Total Cost of Ownership Debate

One of the biggest challenges to FOSS adoption in the business community, and particularly in the desktop market, has been the debate surrounding the 'total cost of ownership' (or TCO). Seeing that many users were simply downloading FOSS at zero cost and beginning to use it

profitably, proprietary software vendors raised this issue to argue that the initial or acquisition cost for any software isn't the only cost involved, but also includes costs arising from using the software, such as training personnel, maintenance, bug fixing, upgrading etc. Given these costs, the proprietary community argued, the advantage of a low cost acquisition of FOSS is wiped out, and, in fact, the total cost, they showed, was higher for FOSS. The FOSS community promptly challenged and countered this argument, showing on their part, that the TCO was lower for FOSS. Thus resulting in a debate.

TCO computations are based on including recurring and continuous costs of adopting a particular software. Some examples of these costs are: cost of annual or per-call maintenance and support, cost of additional features, cost of upgrades for software, cost of training personnel, cost of purchasing new hardware if so required by the new software, cost of additional copies of software for scaling up owing to demand, and costs of modifying other software to be compatible with the new one. TCO calculations are typically done for several years into the future, usually the expected effective life-time of the software.

TCO computations are confounded by several factors making the calculations and then the comparisons hard. For instance, if just two desktop operating systems such as Ubuntu Linux and Windows XP are compared, it would be hard to match them up feature-by-feature and then price them. It is because the feature bundles in the different variants of the packages would not be comparable. So the TCO comparisons are then based on the basic minimum capabilities that are required in the given circumstances which are based on the company in question, its business requirements, its desktop application requirements etc. Within this circumscribed context the other costs are added (those of upgrades, training, patches etc mentioned above). This approach then necessarily becomes specific to a context and its peculiar circumstances. To counter this

problem, the analysis is often conducted for specific sizes of companies (say small, medium and large) and the results generalized (not accounting for the kind of industry the company is in).

It is widely known that companies of both proprietary and FOSS products have amassed data showing that their particular products have lower TCO as compared to the others. This underscores the nature of the problem with the entire TCO debate. Using particular contexts and domains, it is possible to show off the merits of some products versus others. How generalizable these results are is always in question. Businesses using FOSS products would be best advised to perform their own TCO computations, perhaps based on some of the already published criteria and then decide on their own which type of software to adopt.

8 FOSS for Business: Recommendations

A study conducted for the European Union in 2006, showed that FOSS-related services could reach 32% of all services by 2010. Further, FOSS-related businesses could contribute up to 4% of the gross domestic product of EU by the year 2010. These figures are not surprising as FOSS already contributes in a large way in the economy of EU in terms of products, services and employment.

Given that Asia, and in particular India, have a large number of FOSS developers, it would not be impossible to assume that FOSS will play an important role in the economy in Asia too. As there have been no comparable studies of Asia, it is difficult to say just how much the contribution of FOSS is likely to be.

For businesses in Asia to set up and operate in the FOSS space it is essential that they evolve a strategy that uses the unique elements of FOSS. This section identifies a few issues that would be essential to evolve such a strategy.

8.1 Essential FOSS Skills

The cases discussed in the sections above point to three important aspects of FOSS that businesses have to build on.

1. Understanding FOSS products - A business or startup that wants to operate in the FOSS markets has to have a deep understanding of FOSS products and services. The product and platform models require that the business has to have built a FOSS product or participated in its creation. The service strategy requires deep knowledge of a particular FOSS product in order to be able to enhance, maintain or manage it, depending on the requirements of the client.
2. Deep FOSS skills - To have deep knowledge of FOSS the businesses have to recruit and train employees to have deep skills in FOSS. These skills entail not only understanding the technology and being able to manipulate it, but also understanding the community, the rules by which it operates, and the manner in which to participate in it. The FOSS community has its own explicit and tacit rules of engagement that have to be adhered to and any business wanting to operate there has to abide by them.
3. Have a clear licensing strategy - Businesses have to be clear about the kind of licenses they will work with and the manner in which they will build their business around the license. FOSS licenses enable freedoms and also maintain restrictions. Without a clear understanding of the rights and responsibilities that licenses entail, business cannot function in the FOSS space.

The points mentioned above are necessary for any startup that is operating in the FOSS markets, and these are required over and above the general skills of doing business, tackling competition, dealing with customers and competition, etc, that any business or startup in the IT

industry would require.

8.2 Strategies for FOSS Market Segments

Figure 2 below depicts certain segments in the FOSS market as identified in the case studies discussed in previous sections. FOSS products may be introduced in two types of markets (as shown on the vertical axis): replacement markets and unserved markets. Replacement markets are those in which client businesses already have some software product, most likely a proprietary product, and the vendor is trying to introduce a FOSS product as replacement. The unserved market is one in which no particular product is being used and a FOSS offering is being made. It should be pointed out that in the context of developing countries in Asia, the latter markets are quite strong and remain a vast potential.

	Large Enterprise Segment	Small-Medium Enterprise Segment	Government Segment
Replacement Market	Service and Support Scalability (Not cost)	Low cost advantage Features: security, stability, scalability	Emphasize FOSS culture Value of Open standards
Unserved Market	Few in this market. Quality and scalability.	Emphasize cost advantage	Emphasize cost less dependence Value of FOSS culture

Figure 2: Market Segments and Strategies for FOSS Businesses

Along the horizontal axis of the figure are the types of organizations that FOSS products can serve, which consist of large enterprises, small and medium enterprises (SMEs), and government departments. The definition of what constitutes a large enterprise and what are small and medium enterprises varies from country to country. Some countries define enterprises based on their net assets, others on their revenues and still others on the number of employees. One definition, from India, places medium enterprises as those with less than 300 employees and not more than 11 million British pounds in revenues. The World Bank defines SMEs as those employing less than 500 employees.

In light of the complex and multiple definitions, it would be best for businesses to identify their market segment based on their own experiences, as large enterprises would have different needs and characteristics compared to SMEs. Each segment would require a different strategy for targeting FOSS products and services, and this is discussed below.

To target the replacement market for large enterprises, FOSS businesses will have to focus on service and support. As is evident from the Oracle case, the main requirement expressed by large enterprises was that of continued and reliable support for FOSS. Another aspect that has to be emphasized is that of scalability of the product, as such enterprises would typically seek high growth and software to sustain this. Cost is not an issue for them.

There would not be many unserved large enterprises, as such enterprises are typically the first to adopt IT for their growth and strategic advantage. If it all there are unserved firms, such as those without advanced products like a CRM, this market will respond to quality and service features of FOSS products. Cost would not be an issue for them either.

The replacement market for SMEs, would be quite responsive to lower cost possibilities from

FOSS products as well as specific features that FOSS could offer, as compared to proprietary products, such as higher security, stability and scalability. For the unserved (and under served) market in SMEs, the cost advantage of FOSS has to be emphasized, where such an argument is possible. SMEs would typically look for cheaper, unbranded products that could serve their needs.

The FOSS culture of openness, sharing and freedom would work best in the government markets, as these values would align well with most government policies. Also, the value and need of open standards, their durability, and their cost implications would impress government departments. For unserved markets, the argument that there would be less dependence on foreign goods would work strongly, along with a promise of quality and the FOSS culture's inherent advantages.