Open Source – A New Software Development Paradigm

Bela Ban

Open Source (OS) is a new paradigm for software development, exemplified by (but not restricted to) the Linux development community. Simply put, it is based on the notion of giving software (including sources) away for free, and making money on services, customising and maintenance. The departure from a closed to an open development model requires explanation, especially with respect to the advantages of OS for users, and how suppliers of OS can benefit from OS in terms of revenue. This article gives an overview of OS and tries to destroy myths often heard in conjunction with free or open software. There are indications that OS is able to establish itself as a valid and working business model for software creation, in addition to the existing ones.

1 Definition

Free software has existed for a long time, rooted in the academic community. The *Free Software Foundation* [FSF 99], founded by Richard Stallman, is commonly seen as the midwife and leader of free software. Newer projects such as Linux or Apache, have joined the effort only recently.

As some proponents of free software felt more and more that their model would be able to achieve more than just providing free software to the (mainly) academic and research community, a few advocates of free software founded the *Open Source project* [Opensource]. Its major purpose is the dissemination of free software, mostly targeted at the penetration of the business world (or "world domination", as Linus Torvalds once jokingly called it). The term *free* had a too revolutionary and anti-commercial taste, therefore *open* was chosen instead. Additionally, the OS license is less restrictive than some of the free software community, and allows for example to make money with free software.

Eric Raymond, the main proponent and co-founder of the Open Source project, is generally credited with establishing the movement of OS through his seminal paper "*The Cathedral and the Bazaar*" [Raymond 98a].

Open Source software, as its name implies, includes source and binaries in a software distribution, that means, the source code must be included in a distribution, or there must be a way to obtain it easily (e.g. downloading from a web page). Its license requires vendors to provide the software for free. As they therefore cannot make money on the software itself, they need other incentives to provide OS software. Some of the business models underlying OS are discussed later in this article.

An example of OS vendors are the various Linux distributors, who package the kernel plus a suite of tools and applications together with printed book-form documentation (and an optional support contract) for a fee. However, conforming to the OS spirit, the same material is also provided in electronic form on their Web sites for users to download. The added value of their service is printed documentation, a pre-configured set of applications and the benefit of avoiding to download (potentially large) distributions from the web.

2 Characteristics

The characteristics of OS are described in [Raymond 98a]. I shall briefly summarize the most important ones.

> *"Provided the development coordinator has a medium at least as good as the Internet, and known how to lead without coercion, many heads are inevitably better than one".*

This contradicts Brooks's Law which states that "adding developers to a late project makes it later" [Brooks]. However, the Linux model has shown that by carefully dividing the development efforts of literally hundreds of developers located all over the world and connected via the Internet, it is indeed possible to develop such a system, without getting bogged down by complexity and communication overhead. A "benevolent dictator" (Torvalds) is *primus inter pares* among a small set of co-developers, who are responsible for a specific subsystem of Linux, who in turn accept changes from a larger number of developers. Such a hierarchy avoids that the dictator himself becomes a bottleneck. Another argument is that the culture of OS projects (bazaar) is distinctively different from closed projects (cathedral); developers in bazaars are less "territorial" about their code and more willing to reuse code written by others. It would go too far to describe the mind-set and culture of OS developers in detail here, but an excellent paper into the social forces governing OS projects is given in [Raymond 98b].

Bela Ban is a post-doc researcher in the CS department at Cornell University where he is currently working on a toolkit for reliable distributed group communication (http://www.cs.cornell. edu:/home/bba/javagroups.html). His interests include OO in distributed computing, meta object protocols, components and triathlon.

"Good programmers know what to write.
Great ones know what to rewrite (and reuse)"

This suggests that the amount of reuse in OS projects is higher than in closed projects due to the fact that there are no secrets, product fees, non-disclosure agreements and other types of legal bindings typically found in cathedral projects which prevent large scale reuse from occurring. Having source code available is a very good way of customizing a software, and avoids the need to re-invent the wheel in case the binary release of a software cannot be adapted to one's needs.

"Release early. Release often. And listen to your customers"

Contrary to cathedral projects, bazaar projects are often released at a very early stage, giving potential co-developers the chance to play with the code, possibly finding bugs and sending bug fixes back to the author. An OS project always looks at its users at least as testers and perhaps even as potential future co-developers. The invitation to report bugs and even send corrections with a bug report to an OS project, together with the promise to be honourably mentioned in the credits section, seems to be enough incentive for a large number of programmers to contribute to a project, no matter how small or large the contribution is. Raymond argues in [Raymond 98b] that prestige acquired through a contribution to a project is considered higher in the OS world than monetary benefits.

The direction a software project takes depends a great deal on the suggestions of its user base. Since users have real needs, this process ensures that a project never steers into the waters of unneeded functionality.

"Given enough eyeballs, all bugs are shallow"

This statement underlines the importance of having a software exposed to as many users as possible, on different platforms, using a variety of operating systems and having a variety of differing requirements. It states that even the hardest bugs will eventually be found and corrected (not necessarily by the same person). The more people use the software, the more bugs will be found. Having source code available is an advantage compared to commercial software, as users of the code can not only detect a bug, but also localize (and possibly fix) it in the code.

Even the biggest company in the world cannot match (let alone afford) the manpower available harnessed by an OS project, where developers are spread all over the world, using the Internet for communication and coordination.

3 Advantages

In this section, I will summarize some of the benefits for users of OS software, taking Linux as the up to date major example of an OS project. Linux started as a one-man (Linus Torvalds) programming project at the University of Helsinki in 1992-93. After releasing a first version of the kernel, more and more developers joined the effort, and Linux grew gradually into a full-blown operating system. It seems that nowadays Linux matches and even exceeds other operating systems in terms of stability and efficiency, making it one of the big players on the server side.

Fewer bugs: a large number of people use the software, therefore more bugs will be detected. As source code is available, developers may even correct a bug they found and send back a fix to the authors of the software.

Better reliability: the point above leads to more reliability. An OS software is typically used in many diverse hardware and operating system environments, and is therefore tested to its limits. Linux as an example of OS also seems to be more efficient (faster, using less memory) than other commercial operating systems.

No vendor dependence: OS software is largely developed in a decentralized way. When the owner of a project stops maintaining it, someone else will take over from him and continue the effort. This is in contrast to a vendor going out of business: in this case a customer may have the right to obtain the source code, but still needs time and money to maintain it. Another aspect is that a software can be posted to any platform, without depending on the vendor to do so.

Shorter development cycles: new features are developed in less time as compared to commercial projects. Bug fixes are sometimes available the day after they were reported. This is because a new feature or bug fix is always appealing to at least one member of the large decentralized development community, and hence will be done quickly and unbureaucratically. In commercial organizations, change requests and bug reports are ordered according to perceived importance, and consequently it might take a long time until they are provided.

Better support: the OS community is very helpful in supporting users (bug fixes, addition of new features). A question put to any of the newsgroups of an OS software is often answered within hours (for free, without having to purchase a support contract). However, if this is not enough, people can always buy commercial support for a certain OS software from a "support seller". Support selling is one way of making money with OS, and is discussed later.

Better configurability and personalization: as source code is available, an OS software can always be configured to one's personal needs by modifying the code. If the modifier feels that others might share this need, he will typically feed the modification back to the project.

Educational benefits: last but not least, having the source code available allows newcomers in the field to study the implementation of a software, gaining valuable experience. There are indications that more and more operating systems classes taught at universities use Linux. Researchers might modify Linux to test out new ideas, which would be impossible using commercial operating systems.

4 Business Models

The most important question for a potential provider of OS is how to earn money with it. I am looking here at the business aspect of OS, not at idealistic reasons (for an excellent treatment of the non-monetary aspects see [Raymond 98b]).

The major business models for OS are support sellers, loss leaders, widget frosting and accessorising [Raymond 98a].

Support Seller

As the name implies, this model is centred around supporting OS software. It includes for example selling Linux distributions (e.g. RedHat, Caldera, SuSe), customizing, training, and consulting. Even if all the software to setup a Linux box might be free, it would still be a tedious task to find all the packages that one wants to install, and make sure that they are compatible to each other. Services are in high demand; especially commercial organizations care about "solutions", that is, they prefer their hardware and software to be installed by the same vendor, who would also give support, and perform customizing. In a nutshell: the software itself becomes less important, therefore give it away for free, and make money on the services.

Loss Leader

In this model, giving a certain area of the product portfolio away for free increases the likelihood that the other products that are only commercially available, will be bought. An example is a company that penetrates a market with a free product (e.g. to become known in a new area), and offers enhanced versions of it, or entirely different products related to it, for a fee. Netscape is probably the most prominent loss leader, giving the Communicator product away for free, and making money from its NetCenter product. Other examples are Sendmail Inc. (for Sendmail) and Scriptics (for Tcl/Tk). In these cases the loss of potential revenue through giving a product away for free is compensated by sales of other products.

Widget Frosting

This model applies to hardware (widget) vendors. To increase sale of hardware, they provide the software for their hardware for free. This model was heavily used in the days of the mainframe, where companies like IBM only sold "boxes" and did not charge for accompanying software. Even nowadays, PCs contain a large number of pre-installed software packages (usually not increasing the price of the PC), as an attempt by different PC vendors to gain an advantage over their competitors to sell their own PCs. The idea behind this is that software has become a commodity, and can be duplicated many times (legally or illegally, e.g. pirate software), whereas hardware still has to be purchased by each user separately.

Vendors of peripheral devices, such a graphics cards, network interfaces etc., seem to increase their sales in the Linux community by providing free device drivers, specifically tailored to the Linux operating system, and making the specifications public. Especially in the area of peripherals (e.g. graphics cards), providing device drivers for every possible hardware configuration, and ensuring that all cards work in all possible configurations, is almost impossible. Making the card's specifications and drivers open allows to partially move that effort to the OS community. The advantage is exemplified by Linux, where a lot of peripheral devices are supported.

Accessorising

Selling accessories "around" the OS market seems to be a viable business model too. Accessories could be books, journals, conferences etc. The companies following this model are not typically involved in providing OS software themselves, but build on what is produced by others. However, as for example in the case of O'Reilly & Associates, OS developers might be sponsored by such companies, as the latter indirectly increase the revenues of the companies.

Others

There are various other business models of OS, including some hybrid approaches. Hecker [Hecker 98] gives a good overview for the interested.

5 Outlook

OS has come a long way: initially only used and maintained by the academic and research community, it is gaining acceptance in the commercial world. The rise of the Internet can be seen as an accelerator of OS: projects can be developed on a world-wide scale, with developers interconnected through the Internet and "shipment" of software done through the Internet. Additionally, as a working infrastructure, the Internet demonstrates the power of OS, as it is mostly based on free software (BIND, TCP/IP, Sendmail, Apache etc.).

In the author's opinion, the near-term future of Linux is on the server side: its reliability and performance are not matched by other commercial alternatives. An example is the Apache Web server which has by far the largest share of the Web server market. In the longer term, the target of Linux is clearly the desktop. However, to succeed in this area, Linux has to become more user-friendly, and easier to install. Efforts in this area include provisioning of more complete and easier-to-install distributions, and PCs pre-conceived with Linux.

References

[Brooks 75]
Brooks, Frederick. The Mythical Man-Month: Essays on Software Engineering. Addison-Wesley Publishing Co., 1975. ISBN 0201835959

[FSF 99]
http://www.fsf.org

[Hecker 98]
Hecker, Frank. Setting Up Shop: The Business of Open-Source Software. http://people.netscape.com/hecker/setting-up-shop.html

[Opensource]
http://www.opensource.org

[Raymond 98a]
Raymond, Eric. The Cathedral and the Bazaar. 1998. http://www.tuxedo.org/~esr/writings/cathedral-bazaar/cathedral-bazaar.html

[Raymond 98b]
Raymond, Eric. Homesteading the Noonsphere. 1998. http://www.tuxedo.org/~esr/writings/homesteading/homesteading.html