



Open Source Success Factors

By CK Wong 2006.07.11

<http://www.ck-wong.ca/Freeware/open%20source%20success%20factors%2020060725.pdf>

Introduction

Open source software has been existed over quarter of century. Its perforation becomes wildfire in the last decade. What are the driving forces?

Major Factors

The major road blocks for freeware to be successful are limited by the following factors:

1. Large adaptation communities. To make a piece of software successful, it must have a large user base. Without that it could not proclaim its success and no motivation of promotion.
2. Source management. With all the contribution to the world, the source code must be submitted to a repository and able to retrieve and control's updates. The concept of multi-stream development is being tested to the most extend in the case of open source. There are a number of technology advancements this possible.
3. Development environment. Code development is very much like sweatshop two to three decades ago. The basic tools are limited to text editor. The luxury of a language sensitive editor and debugging tool and its library supporting environment is a must to deal with monster size code body.
4. Requirements formalization. Software development depends on the proper formulation of the requirements. Requirement specification has been supported by multiple standards. The most important is the ability to define clear and reusable requirement. With the new generation of object oriented designed developer emerging from college and home, the requirement specification could become a clear medium of exchange.

The following sections explore the changes that contribute the success.

Adaptation by Communities

When a piece of software used by a small community the penetration momentum is small. The compound effect of snow ball applies to the adaptation of software. In the case of the freeware, the entrance to end user is hinged with the supporting service around it. Without the supporting service, it is just a standalone application. For example, an MP3 player without MP3 music files would not be very popular. An application like Netscape gains the acceptance because Netscape creates contents so that the end user could use the Netscape Communicator. When it comes to corporate, the matter becomes a little bit more complicated because there is liability. The tradition model of service is accomplished by the warranty and liability to some degree to protect the consumer. This becomes the key point and the stigma when a corporate develops an application on top of the freeware. When the freeware component failure, who will be liable. The tradition

liability model is built on a pyramid model. The last layer liability is depending, to some degree, on the lower layer of software it builds on.

This liability model becomes a sticky point. After the Java and Linux hit the user community, there are two liability models show up. One is specifically tribute to the fact that now warranty or liability is provided. Every one will have to live with it. The acceptance of no liability usage model by corporate increases the perforation of freeware. The acceptance is further fortified by the quality of the software which in many cases high and bug turnaround time is much faster than the corporate supported and paid software.

The second worry by the corporate which usually has a time limited period to fix the problem in software industry could not encompass the uncertainty of the bug fixing turnaround process. This is addressed by entrepreneur who provides the custom changes for corporate for such situation. Red Hat is the prime example that supports the Linux with a fee.

What the corporate sees is the cost reduction in the software manufacturing. The benefit is too good to pass.

Another major factor for the adaptation is the relaxation of the GNU licensing model. This model is a nightmare for the legal people. The terms and conditions are so alien to the commercial language that the corporate lawyers have a difficult time to comprehend. The worse nightmare was the term and condition that the integration to the freeware made the product a freeware **with source code available to whoever demands**. The trade secret will not be protected. The later version allows the meager integration to by pass this retrain. The development of loadable/shared library provides the technical support for the meager integration.

What we conclude here is that the adaptation of freeware is not just company politics. It is a strive effort from both side of the world to fill the gorge of separation. I give the kudos to Free Software Foundation, GNU, Sun and especially IBM.

Source Management

When one person works on a piece of software, it could be confused enough to lose a piece of two. When the body of software is doing parallel development, the source management becomes a very difficult job. The chaos amplifies when you have multiple streams of developments. For example, Linux supports have to work on the new version and fixing the bugs of the current version of various flavors. The change from one bug fixing may have to propagate to all other stream of development. The logistic of integration and the testing concerting are all become a complicated jobs.

Existing software such as CVS is doing a good job but when the developers are located around the world, it is not just the source management task we have to provide the

network that allows the exchange of files. The concurrent access will take on a true and accurate meaning.

Fortunately, the cost of DSL, cable modem and many other high speed Internet devices are available in a very low cost. Just like AOL, the access the service would cost the user to pay for the edge access. The AOL service is free. The result allows the faster turnaround of the changes and testing. We have observed the phenomenon that free promotes usage and can benefit the world rather than a single corporate or person.

Internet has become a pay it forward service. It is not free. The payback comes in some other way we may not expect.

Development Environment

The development environment composes of the hardware, the software and the network. In previous session we have discussed the cost of the Internet usage has made the ISP penetrate to over 50% of the home. We don't need that high percentage for the developer but they need high speed. In Canada, we enjoy the luxury of cheap 5M-6M access through cable or DSL. In the State and other part of the world, the speed is generally limited to 100K or 2M at home. Consider the 56K was the norm in 5 years ago, this is a quantum leap.

Hardware has been quantum leap too. An old Pentium III class machine is as expensive as the refurbished P4 machine. This is especially true for the laptop. The brand new laptop is usually cheaper than the used one with less memory and smaller hard drive. The faster machine allows the hobby developer tackle more complex software.

Integrated development is also getting much more sophisticated. This can provide you flexibility to search the library and documentation easily. With the availability of Eclipse IDE (donated by IBM), there is a standard platform people could share in any operating systems. The development experience becomes portable.

Requirement Formalization

Requirement formalization is the essence of software development. For freeware to be adopted by other companies, it must provide a unique function that other people either found it too costly (like command line editing) to develop or has no intellectual property to develop (like the compression algorithm). Freeware must have the requirements that fit many bills and many applications. The continuous recycling of requirement creates conical form of requirement which essentially the core of the problem with no extra fat. This lean and mean specification is the basis of reusability.

For example, there is complain on Linux that no real-time support is offered. To handle this problem, there are several attempts to solve it through micro-kernel, kernel modification, special system services and other. Until there is a popular solution that every one, the requirement refinement continue. This no time constrain process is unique to the freeware community but cannot exist in any business.

Are we going to become a freeware world?

This will not happen. We have access to many services, e.g. book from the library. We borrow the book, in Canada the service is *FREE*, but no fee is required. However, the municipal tax pays the fee. This means the service is paid indirectly while usage of content could be free.

You can consult the medical dictionary on the Internet for free because you pay the ISP already.