OPEN SOURCE VS. CLOSED SOURCE

Vidyasagar Potdar, Elizabeth Chang
School of Information Systems, Curtin University of Technology, Perth Australia 6845

Keywords: Open Source, Open Source Software Development Process

Abstract: Open source software development represents a fundamentally new concept in the field of software engineering. Open source development and delivery occurs on Internet time. Developers are not confined to a geographic area. They work voluntarily on a project of their choice. Developers work for peer-recognition and self-satisfaction. Open Source software is always in an evolutionary stage: it never reaches a final stage. As new requirements emerge the software is enhanced by the user/developers. In this paper, we give an introduction to the insights of open source software development. We then elucidate the perceived benefits and point out the differences between open source and closed source software development approaches.

1 INTRODUCTION

The concept of free software is not new. It had been here since 1960s, universities, such as MIT and corporate firms such as Bell Labs freely used source code for research [Webber 2000, Perens 1998]. Software was not a means of revenue generation. It was used to hook more and more customers to buy new computers [Malcolm 1998]. In early 1980s, Microsoft started writing software with the sole purpose of profit. It gave only compiled code; source code was hidden from the user. This move had a great impact on the programmer community, particularly at MIT. Richard Stallman, researcher at MIT, was frustrated because he could no longer modify his software to satisfy his requirements. As a result he founded the "Free Software Foundation" (FSF) to develop and distribute software under the General Public License history (GPL) whose main motive was to prevent proprietarization of cooperatively developed software. Around the same time Bruce Perens defined a set of guidelines that a software license must grant its user, and he called this Open Source Initiative (OSI). In this paper we describe the open source software development and compare it with closed source software development. The paper is organized as follows. In section 2, we describe how open source software is developed. In section 3, we compare open source and closed source software development approaches.

2 OPEN SOURCE SOFTWARE DEVELOPEMNT

Bruce Perens [Perens 1998] defines that Open Source is a specification of what is permissible in a software license for that software to be referred to as Open Source. It should be freely distributed, it should come with it's source code, it should allow modifications and derived works, it should maintain integrity of the author's source code, it should not discriminate against persons or groups nor should it discrimination against any fields of endeavor, the license must not be specific to a product and it should not contaminate other software. To put it in simple words, software that violates any of the statements mentioned in the formal definition of open source is not open source software e.g. trial software's, use restricted software, shareware, freeware, royalty free binaries, royalty free libraries etc.

2.1 Who is an open source developer?

Simply put, "any one who contributes to the open source project is an open source developer", such as a user of the software, a developer who develops the software, a debugger or hobbyist who likes spending time on open source, or a promoter who funds such a development.

2.2 What do they do in Open Source?

Open source developers are involved in a variety of

Potdar V. and Chang E. (2004).
OPEN SOURCE VS. CLOSED SOURCE.
In *Proceedings of the Sixth International Conference on Enterprise Information Systems*, pages 609-613
DOI: 10.5220/0002603206090613

activities such as designing, coding, debugging, using, etc. Each activity occurs simultaneously. Developers are free to choose whichever project they want to work. Open source developers are highly efficient; they don't spend long hours by starting projects from scratch, rather they find a similar software and makes the additional changes to suit their requirements [Raymond 2000]. Parallel debugging is the key to open source success [Valloppillil 1998]. Users also play a vital role in the debugging process by reporting bugs to developers or sometimes fixing it themselves. This is considered the best way to test the software because these users actually act as beta testers. Apart from this the main task of open source developers is releasing new versions of the software that is bug free. Developers are well aware that users are the best beta testers. So the more often the software or component is released the more often we can expect bug reports, and can try and fix it to make the upcoming version more robust.

3 OPEN SOURCE VERSUS CLOSED SOURCE SOFTWARE DEVELOPMENT

OSSD is a recent phenomenon, while traditional closed source software development has been here since the dawn of software development. One major difference between these two models is source code visibility. In this section we will point out most of the differences between these approaches.

Closed Source Software Development (CSSD)	Open Source Software Development (OSSD)
Process Models	
Have clear phases and milestones	No clear cut milestones
High level view, it is a Liner engineering process	Concurrent and parallel process
Interactive-liner development occurs but not on a high scale [Satzinger et.al]	Very high degree of iterative and oscillative development because developers regularly keep updating the software on daily or weekly basis.
Re-iteration may not be possible always	Re-iteration is highly possible
It normally follows a spiral or iterative model of development i.e. software development goes through all planning, design, implementation etc phases recursively [Satzinger et.al].	It follows an evolutionary model for development since the software doesn't have a final state and keeps on evolving according to customer needs [Hissam et.al 2001].
Schedule	
There is a deadline	No Deadlines
Requirements Definition and Specification	
Single Requirement	Multiple Requirements
CSSD starts with requirements definition and specification. Here requirements are vague. Project developers are not aware of the actual requirements. They need to interview stakeholders to elicit requirements and then start implementing [Webber 2000, Satzinger et.al].	OSSD starts with a motive of requirements satisfaction. Requirements are clear, as developer is fully aware of the requirements [Hissam et.al 2001].
All the user requirements may not be implemented because of time or budget constraints [Scacchi 2002, Satzinger et.al].	All user requirements may be implemented, as user is often the developer [Webber 2000, Hissam et.al 2001].
Users may suggest requirements but they may or may not get implemented [Satzinger et.al].	User suggests additional features that often get implemented.
System Architect and project manager decide which requirements will be incorporated [Satzinger et.al].	Core members of the project decide which requirements to be incorporated [Raymond 2000, Hissam et.al 2001].
Documentation	
Project Plan is documented, and followed	There may or may not be a project plan
Once the requirements are clear they are documented [Satzinger et.al].	Requirements may or may not be documented [Scacchi 2002].
Design and Testing must be documented	Design and Testing may not be documented

Analysis and Design	
System Architects and Project Mangers spend a lot of time in designing the project [Satzinger et.al].	Designing is often merged with implementation. No separate design phases [Tran 1999, Tran et.al 1999, Hissam et.al 2001].
This kind of development is more of a solution kind of development. Developers create solutions for a big company. It is more like customized development.	This kind of development is more components-based i.e. plug-n-play type software. Developers create small programs, which work on a variety of platforms.
User may find one or more proprietary solution for a particular problem domain.	User is free to choose from a variety of free solution for a particular problem domain [Hissam et.al 2001].
Software Architecture	
Maintaining consistent software architecture is enforced during the development.	As a software system evolves there is a possibility that its architecture may change. Maintaining consistent software architecture is difficult.
During the development, system changes are often made without considering the overall effect on the systems architecture. These changes often introduce structural anomalies between the software's conceptual and the concrete architecture [Tran et.al 1999].	Minimum concerns, so we need to maintain the software architecture.
There is rarely a drift between software's conceptual and concrete architecture.	OSSD is highly prone to drift because of its highly collaborative and distributive nature
Developers and managers maintain the software architecture and prevent it from drifting.	Most OSS developers are involved in OSSD just as a hobby so they don't care much about the software's architecture. Developers are free to contribute their developmental effort. This freedom often causes the architectural drift.
Implementation	
The rate of development is comparatively slower that the open source because the number of developers assigned to a CSS project can never match a full-scale open source project like Linux [Satzinger et.al].	Rate of software development is very high because it is a parallel collaborative development..
Developers productivity may decrease if he is forced to work on a project which he is not interested in.	Since developers are not forced to work on a particular project their productivity increases [Hissam et.al 2001].
Here developers work for economic incentives [Webber 2000].	Developers work for peer recognition. People know that recognition as a good developer is easily montizable [Webber 2000].
Testing	
Users don't act as beta testers.	Users act as bug reporters, beta testers, implementers etc. Whenever a user finds any bug in the software one may immediately try to solve it or bring it to the notice of the community [Webber 2000].
Release and Delivery	
Release is not too often. There may be only yearly releases.	Releases are quite often. Software is released on a daily or weekly basis [Webber 2000, Hissam et.al 2001].
Product is often released due to marketing pressure and tight schedule. Such a product may be buggy [Satzinger et.al].	Product is only released once the developer thinks that it has reached a functional stage [Webber 2000].
A project that starts is forced to complete. It may also get infinitely delayed due to improper planning or management or inability to complete [Satzinger et.al].	May or may not finish depending upon user interest. If user or the project owner losses interest in the project it may get delayed, unless that project is taken over by some one who finds that project interesting [Kuwabara 1999, Raymond 2000]
Product	
Product may soon reach a finalized state once the documented requirements are implemented [Satzinger et.al].	Product is never in a finalized state. Requirements emerge and get implemented. It is an evolutionary process [Hissam et.al 2001].
Success depends upon following a tight schedule [Godfrey et.al 1999].	Success of any open source project depends upon the needs and interest of the community [Godfrey et.al 1999].

Maintenance	
Service packs are needed quite often to repair bugs.	Service packs are not needed, as bug reporting and bug fixing is a common feature in OSSD [Hissam et.al 2001].
Maintenance is major phase in software development.	In OSS most of the time is devoted to active development and corrective maintenance rather than preventive maintenance [Tran 1999]. Preventive maintenance is considered a boring job as it hampers the flow of the development process.
Productivity, Quality and Cost	
Adding more manpower to a project to increase its speed often delays it as it increases the level of coordination and complexity.[Kim 2001]	OSSD has its own way of maintaining coordination and complexity.
Slow and expensive	Fast, Better and Cheaper [Scacchi 2002]
Doesn't work well with speed, quality and cost. At one time only one factor can be satisfied fully. E.g. if speed is maintained quality and cost may go up or if cost is to be maintained quality and speed may go down [Lerner et.al 2000, Satzinger et.al, Scacchi 2002].	All the three factors can be satisfied simultaneously. Cost is reduced because no one is paid for the job everyone is a volunteer. Speed is increased because development is parallel and collaborative in nature. And finally quality is maintained because the product is released only when the developer think the product is stable and workable [Scacchi 2002].
Source Code	
Source code is hidden from the user.	Source code is open. User can anytime view and modify the code [Perens 1998].
Hidden source code prevents user from modifying the software to add new features.	Source code availability helps user to modify the program to suit individual needs [Perens 1998, Hissam et.al 2001].
Environment	
Often we find centralised, single site development takes place.	Often Decentralised, distributed, multi-site development takes place.
Development happens in a geographically confined area.	Development occurs on the Internet, which facilitates rapid development [Webber 2000]
Group work and Communication	
Inconsistency is easily managed by face to face or weekly team meeting.	Open source is co-operative and need high level of co-ordination over the Internet and multi-site. Lack of coordination among developers results in code forking [Webber 2000]
Security	
Security thru 'obscurity'	Security thru 'open source'
Market believes commercial CSS is highly secure because it is developed by a group of professionals confined to one geographical area under a strict time schedule. But quite often this is not the case, hiding information doesn't make it secure, it only veils weaknesses [15].	OSS is not market driven it is quality driven. Community reaction to bug reports is much faster compared to CSS which makes it easier to fix bugs and make the component highly secure.
Third party security certification is not possible with CSS	You can ask for a third party security certificate or get your system scrutinized by a professional security expert for possible back door entries. [15, Obasanjo 2002 , Gutschmidh 2001]
Security cannot be enhanced by modifying the source code.	The ability to modify source code could be a great advantage if you want to deploy a highly secure system..

6 CONCLUSION

From the study that we have conducted it comes to our notice that OSSD is similar to its traditional counterpart in many aspects, but there are many areas in which it differs tremendously and these features make it different from the CSSD. As a concluding remark we can say open source software is a competent alternative to CSS.

REFERENCES

Godfrey, M. W., Qiang T, *"Evolution in Open Source: A Case Study"* http://plg.uwaterloo.ca/~migod/papers/icsm00.pdf [Nov 01, 2002]

Gutschmidh, T, 2001, *"Thoughts on Java and Open Source Security"*http://softwaredev.earthweb.com/sdopen/article/0,,12077_631291,00.html. [Aug 25, 2002]

Hissam S, Weinstock C., Plakosh D., Jayatirtha A., *"Perspectives on Open Source Software"*, Software Engineering Institute, Pittsburgh, Nov 2001 p49. http://www.sei.cmu.edu/pub/documents/01.reports/pdf/01tr019.pdf. [Nov 01, 2002]

Kim J., 2001, *"A descriptive process model for open source software development"*http://sern.ucalgary.ca/students/theses/KimJohnson/toc.htm [Nov 01, 2002]

Kuwabara K, 1999, *"The Bazaar at the Edge of Chaos"* Chap 2: A Brief History of Linux. http://www.cukezone.com/kk49/ linux/chapter2.html. [Oct 27, 2002]

Lerner J, Tirole J, 2000*"The Simple Economics of Open Source"*, p.11, Available at:http://www.people.hbs.edu/jlerner/simple.pdf. [Oct 28, 2002]

Malcolm M., 1998 *"Profit Motive Splits Open Source Movement"*, http://content.techweb.com/wire/story/TWB19980824S0012 [Oct 27, 2002]

Obasanjo D, 2002,*"The Myth of Open Source Security Revisited v2.0"*http://softwaredev.earthweb.com/sdopen/article/0,,12077_990711,00.html. [Nov 01, 2002]

Perens B, 1998 "*The Open Source Definition*" Available at: http://perens.com/Articles OSD.html [Oct 27, 2002]

Raymond, E. R., 2000, *"The Cathedral and the Bazaar"*. Avalable at :http://www.tuxedo.org/~esr/writings/cathedral-bazaar/cathedral- bazaar/ [Oct 27, 2002]

Satzinger, Jackson, Burd, 2000 "System Analysis and Design in a Changing World", Thomson Learning.

Scacchi W., 2002, *"Is Open Source Software Development Faster, Better and Cheaper than Software Engineering?"* Available online http://opensource.ucc.ie/icse2002/Scacchi.pdf Accessed on: Nov 01, 2002

Tran, J.B., Holt, R.C., 1999, "Forward and Reverse Architecture Repair" *Proc. Of CASCON '99*, pg. 15-24

Tran, Godfrey, Lee, 1999 "Architectural Repair of Open Source", http://plg.uwaterloo.ca/~migod/papers/iwpc00.pdf [Nov 01, 2002]

Tran J. B., 1999, *"Software Architecture Repair as a Form of Preventive Maintenance"*, Available online: http://plg.uwate rloo.ca/~j3tran/papers/thesis.html Accessed on: Nov 01, 2002

Valloppillil, V, 1998, *"Open Source Software, A (new?) Development Methodology"* Available at:http://www.openresources.com/documents/halloween-1/node4.html. [Oct 27, 2002]

Webber S., 2000 "*The Political Economy of Open Source Software*", California. http://e-conomy.berkeley.edu/publications/wp/wp140.pdf. [Jan 7, 2004]