International Conference on Multidisciplinary approaches in Social Sciences, Humanities and Sciences
Sri S.Ramasamy Naidu Memorial College, Sattur, Tamil Nadu, India (MASHS-18)
14th December 2018 www.conferenceworld.in ISBN:978-93-87793-61-3

OPEN SOURCE – THE EVOLUTIONARY IMPROVEMENT OVER EXISTING SOURCE CODE

[1] **Amit Sanotra,** Research Scholar, Computer Application, Sri Vankateshwar University, UP

[2] **Dr. Vipin**, Associate Professor, Sri Vankateshwar University, UP

ABSTRACT:

Open source depicts the interference with source code by anybody with the perspective to alter it. It allows the users to access the source code of a software any number of times to add or remove a piece of code to the accessed code and will impact the software with addition of new features or removal of non-used features as desired by the user with a view to redistribute it for further use of its own or user community. The positivity of open source is that it leaves behind the concept of piracy due to the availability of source code free of cost to everyone on the internet. Open source provides the opportunity to contribute to the pool of free software availability

Keywords – Source Code, Piracy, Software, accessed code, free pool

1 Introduction

The term *Open Source*simply means that the source i.e. *code of the software is open.* It means that the software can be downloaded or accessed without paying anything to the developer, in addition the user can download the source code too with the pretext to do any kind of modification in the already written code, for the sake of addition of some features in the software downloaded. Open Source has created the platformto contributeto the free pool of software.The open source software is available around the world without any restriction from the developer towards users.

The purpose of this paper is to provide a description what is meant by "open source". For the purpose investigated several well-known open source projects such as Linux [2], Apache [3] and Mozilla [4]. We also did literature studies on published materials about open source, notably *The Cathedral and the Bazaar* [5], *Rebel Code* [6], *Open Sources* [7] and various several on-line resources and interviewed individuals working on open source projects at

their free time as part of their job in large corporations. From there, we tried to dissect open source further by determining the characteristics that open source projects should or usually have. We determined a set of characteristics that are almost always present and others that vary among open source projects, and this serves as the core of this work. The rest of this paper is structured as follows: Section 2 presents a brief history of open source, which is important for understanding its motives and directions; Section 3 describes some open source characteristics that can be used in determiningwhether a project is or not open source; Section 4 provides some initial conclusions of our work; and Section 5 outlines areas that can be researched further.

2 A Brief History of Open Source

2.1 How it started

The idea to share within a cooperating community, where the source code was made available so that everyone could amend and share it began with the GNU project at MIT in the early 1980s. The idea was to provide *freedom* pertaining to software systems. In 1985 the *Free Software Foundation (FSF)* was pioneered by Richard Stallman to produce some income for the free software movement, not restricting itself to GNU.

Free software, as defined by the FSF, is a program that grants various freedoms to its users. A free software program provides its users with [15]:

- Freedom to run the program for any purpose
- Freedom to study and adapt the code for personal use
- Freedom to reallocate copies of the program, either gratis or for a fee
- Freedom to distribute improved or modified versions of the program to the public

The discourse used by the FSF tends to be confrontational and against proprietary (closed) software, since they view anyone producing this kind of software as big obstacles to the four basic freedoms mentioned above. This is mirrored in the restrictive viral nature of some of their licenses (see section 3.3).

2.2 Free Software and Open Source Movements

Open Source termwas coined in early 1998 in response to the announcement made by Netscape for releasing the source code of its web browser. The new term coined because of factors:

"…it was time to dump the confrontational attitude that has been associated with 'free software' in the past and sell the idea strictly on the same pragmatic, business-case grounds that motivated Netscape." [16]

Immediately afterwards, the *Open Source Initiative (OSI)* was set up to manage and promote the *Open Source Definition (OSD)*. The OSD was composed as a guideline to determine whether a particular software distribution can be called open source or not. OSD asserts nine criteria that open source software must follow; the main three are:

1. The ability to distribute and redistribute the software freely
2. The availability of the source code, and
3. The right to create derived and works through modification.The rest of the criteria deals with the licensing issues and spell out the "nodiscrimination" stance that must be followed [1].
4. The integrity of the author's source code must be preserved, making the sourceof changes clear to the community
5. No discrimination against persons or groups both for providing contributionsand for using the software
6. No restriction on the purpose of usage of the software, providing nodiscrimination against fields of endeavour
7. The rights attached to the software apply to all recipients of its (re)distribution
8. The license must not be specific to a product, but apply to all sub-parts withinthe licensed product
9. The license must not contaminate other software, permitting the distribution ofother non-open source software along with open source oneThe Open Source and Free Software movements can be compared to twopolitical parties within a community. While two political parties agree on the basicprinciples but disagree on practical issues, the Open Source and Free Software doexactly the opposite. They disagree on the basic principles (commercialism, licensing,etc.), but agree on (most of) practical recommendations (availability of source code,ability to modify the code, etc.). They even work together on many specific projects toachieve the same goal: to provide software that is free (in terms of liberty) for all [17].

2.3 Commercialization of Open Source

Open source tried to make Free Software more attractiveto business users since it allows free usage.This means that the companies don't stop in making profit from the software, as long as the source code remains available and canbe modified freely.The way of commercializing open source is by providingservice and distribution packages for software developed in an open source fashion.This is due to the fact that open source software is usually more difficult to installsince it was originally aimed for the hacker community. Free operating systems can be developed who inturn used to install only payable softwares. More and more computing corporations turn their attention to open source as abusiness opportunity. *Innovation*and sharing source code is good for facilitatingcreativity. Commercial organizations are forced to contribute to undermine and dominatingcompetitors. On the down side, they are afraid that maintaining control of an activeopen source project can be difficult. They are particularly concerned with the risk ofcode forking – the evolution of two (or more) separate strands of work from theoriginal code base, which threatens compatibility. This fear prevents some individualsand many companies from active participation in open source developments [7].

Although this code forking risk is always present but can be overcome by ethics of software community. Instead of basing their reputationon "what they have", they measure it against "what they give". This "gift-culture"encourages people to contribute more and binds people together in the same strand ofwork.

2.4 The Open Source Approach compared with Others

To know how the relation of open source toother software approach, some comparisons among several categories of software are there. For simplicity, We differentiate into two categories Open Source Software (including free software) and Proprietary Softwares.There are two kinds of software within the "free" category: *non-copylefted freesoftware* and *copylefted software*. Non-copylefted free software allowsthe permission to modify and redistribute, and in a legal term it is "notcopyrighted". On top of that, it is allows to add more restrictions to the modifiedversion, which means that some copies (modified versions) may not be free at all.Anyone can compile the program and redistribute the binary as proprietary software.*Public domain software* is a special case of non-copylefted free software. Whereas thecopylefted software, doesnot allow to have additional restrictions to beadded when

someone redistributes or modifies the software. As a consequence, everycopy of copylefted software, even after modification, must be a free software. Themost prominent distribution terms for copylefted software are covered in the *GNUGPL (General Public License).*

Proprietary software or paid software is *closed* software in that the source code is not available in any casefor users or community. It always binds terms and conditions for use, and its redistributionor modification is prohibited and if done by anyone will attract the piracy laws. There are two special cases within this group ofsoftware: *shareware* and *freeware*. Both allow people to download, use andredistribute the software for free, but modification is (almost) impossible because theyare usually released in executable (binary) format only. The difference is on the limitof usage, if someone wants to keep using a shareware, he/she must pay a license fee.

One important note is that freeware must not be confused with free software,especially because modification of a freeware is not possible (since the source code isnot available).

There are subtle differences between open source and freesoftware, in particular around licensing issues. For example, open source softwaremay use proprietary library (e.g. the KDE project [20] was using a proprietary librarycalled Qt until September 2000), which is unacceptable in free software.

Comparisons of different kinds of software

Table 1: Comparisons of different kinds of software

	Open Source (Free) Software		Proprietary Software		
	Non-copylefted	*Copylefted*	*Closed*	*Shareware*	*Freeware*
Availability of source code	Y	Y	N	N	N
Permission to					
• redistribute	Y	Y	N	Y	Y
• modify	Y	Y	-	-	-
• add restriction	Y	N	N	N	N
Modified version always free	N	Y	-	-	-
Free Download	Y	Y	N	Y	Y
Time Limit in usage	N	N	N	Y	N
Possibility of making money	Y	Y	Y	Y	N

Table above summarises the main comparisons between the characteristics of thosesoftware categories.There are subtle differences between open source and free software, in particulararound licensing issues. For example, open source software may use proprietarylibrary (e.g. the KDE project [20] was using a proprietary library called Qt

untilSeptember 2000), which is unacceptable in free software. Further investigationsurrounding these differences could provide better understanding, as highlighted insection 5.

3 Features of Open Source

After explaining the features and traitsthe open source projects usually have, we hope to beable to develop a clearer picture on what it really means for a particular project orsoftware development to be an open source projector not. The idea is to have a "ticklist"of open source characteristics, against which the characteristics of the project inquestion can be compared. Additionally, these characteristics highlight the fact thatjust stating that a project is open source does not necessarily provide a precisedefinition.

3.1 Disciplines to consider

It is important to highlight thatsoftware development is a very complex process that draws upon knowledge/expertisefrom many scientific disciplines. Therefore, to understand it better, it is necessary toemphasize its interdisciplinary nature. It appears that open source softwaredevelopment is no exception, and in order to determine the relevant open sourcecharacteristics, there are several disciplines that we would like to consider:

- Computing Science

Covering the technical aspects that need to be considered to engage in an opensource project.

- Management Issues

Dealing with managerial issues and how they relate to open source projects.

- Social Sciences

Addressing areas related to the communities involved in open source projectsand their behaviour.

- Psychology

Accounting for the characteristics of the individuals involved in open sourceprojects. The ethical behavior of individuals is most important.

- Organisational Aspects

Dealing with aspects such as organisational structures.

- Economics

Looking into economic models that underlie open source projects and/orcorporations with respect to their involvement in open source projects.

 Law

Focusing on legal issues.

TheOSI definition for the term open source does address legal issuesextensively, and encompasses some economic aspects. Whereas, it hardlytouches on computing science areas; it ignores the areas ofmanagement, psychology, social sciences and organizational aspects. Furthermore,there is no guarantee that a given project, by simply adhering to the OSI definition ofthe term open source, benefits from the positive effects that are usually related to theterm open source (e.g. being reviewed by many people). The open source softwarecharacteristics proposed by Wang and Wang [11] address some technical aspects, andin less depth, legal and managerial aspects.

In our attempt to understand open source, we determined a set ofcharacteristics that occur under that umbrella term, while considering the variousdisciplines mentioned above. Some characteristics are common to all efforts we wereable to investigate, whereas others vary between projects.

3.2 Common characteristics

We have identified six characteristics that are present insuccessful open source projects, these are addressed below.

Community

All active open source projects have a well-defined community with common intereststhat are either involved in continuously evolving its related products and/or in usingits results.

Motivation

Thereare*different types of contributors*, individuals and corporations. Individuals usuallycontribute for personal satisfaction; some have really strong philosophical beliefsothers do not care as much about such issues. Corporations usually get involved withthe aim to gain market share, undermine their competitors, or simply rely on productsgenerated by open source without having to build a fully equivalent product fromscratch.

Peer recognition also plays a role on motivating contributions by recognizing as appropriate and of good quality by the communityinvolved, both individuals and corporations have their status raised within the givenproject. Consequently, their opinions are considered more carefully with respect toproject related decisions and their reputation may even improve outside the projectboundaries.

Developers are always users

The set of people that contribute code to specific open source projects is alwayscomposed of those that are also users of the code produced. This means that opensource developers are a subset of the open source user community, i.e. all open sourcedevelopers are users, but not all users are developers.

This characteristic explains the fact that there are normally no precisespecifications or requirements documents clarifying what is to be achieved in theproject. Whereas expection must not be that the open sourcecommunity will start developing arbitrary kinds of software. Software developers areusually not expert users of medical systems, nuclear plant control systems, or airtraffic control systems.

The process of accepting submissions

An open source project evolves by receiving submissions from various sources toaddress various aspects of the project. The most common submissions are those ofbug reports and source code, others include documentation and test cases.

Development improvement cycles

Product improvement in the open source software development process can manifestin both breakthrough and continuous improvement modes. Breakthroughimprovement involves dramatic and relatively impromptu changes [21]. Evidence ofthis form of product improvement in open source development was provided byRaymond [5] in the development of Fetchmail. He notes that:

"The real turning point in the project was when Harry Hochheiser sent me hisscratch code for forwarding mail to the client machine's SMTP port…thisSMTP-forwarding concept was the biggest single payoff I got…The Cruftiestparts of the driver vanished. Configuration got radically simpler…the onlyway to lose mail vanished…and performance improved." (p. 47-50)

Continuous improvement involves an increased frequency of change but in smallerand more incrementally consolidating stages [21]. This philosophy of productdevelopment recognises that small improvements build up to larger improvementovertime, but with the added advantage of being far easier to implement. Incrementalproduct improvement through bug finding and fixing is a development hallmark of theopen source paradigm and is embodied in Eric Raymond's original characterization "release early, release often" [5] The idea is to get quick feedback, which can then beincorporated back into the product.More recently such anecdotal claims have been further reinforced by theresearch findings of Aoki et al. with the open source *Jun* project [22]. They trackedthe evolution of the software over 360 versions and identified both incrementalimprovements within single version updates followed by significant functionality

increases requiring major modification to the existing architecture.

9*Modularity*

The benefits of modular design are well established in all engineering disciplines, as itsupports increased understanding during design and concurrent allocation of workduring implementation [23]. However, due to the globally distributed nature of opensource development, well-defined interfaces and modularised source-code are aprerequisite for effective remote collaboration [24].

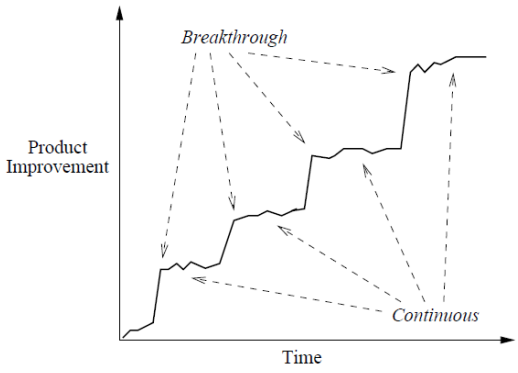

Figure 2: Open source product improvement over time.

3.3 Variable characteristics

There are various distinct areas where the open source are not common. Below is a discussion of some of those.

Choice of work area

Opensource projects often request contributions the areas in which they are interested. Some open source projects willprocess both solicited and spontaneous contributions, whereas other open sourceprojects may be prone to ignoring spontaneous contributions.

Balance of centralisation and decentralisation

The communities within various open source projects are organised differently. Somehave a very strict hierarchy differentiating among various levels of developers, whereas others have a much looser structure. In someopen source projects (e.g. Apache), it is even possible to have more than two levels ofdevelopers. But not all open source projects have multi-level developer groups.

Business model

Depending on the domain that an open source project addresses, different businessmodels may motivate the involvement of commercial corporations, researchers,individual developers and end-users. The business models we have identified so far10 are: own use, packaging and selling, and platform/foundation for commercial orresearch software development.

Decision making process

The decision making process relies on four dimensions that vary from open sourceproject to project. These are the *quality goals*, the *acceptance criteria* enacted, the*cognitive abilities* of the decision group, and the *social structure* within the project.

11

Submission information dissemination process

The information on submissions and their acceptance may be passively disseminatedby the means of newsgroups or comments in the code itself, it may be activelydisseminated by using emails and mailing lists, or there may be some dedicated webspace for statistical information.

Project starting points

Open source software projects may start from scratch or from existing closed sourcesoftware systems, either commercial or research. From the various projects that westudied we could

only find examples of projects that transitioned the full packagefrom closed to open source at once.

Visibility of software architecture

The software architecture of a computing system depicts its structure(s) and comprisesits software components, the externally visible properties of those components, andthe relationships among them [25]. The architecture of an open source softwaresystem may be itself open or closed. The "closedness" may occur intentionally oraccidentally. Having an intentionally closed software architecture means that the coregroup will consciously not reveal the structure to the general public. Anunintentionally closed software architecture suggests that the structure exists in somepeople's minds only.

Documentation and testing

Documentation and testing are important aspects of the software development process.

Good documentation allows people to use – and more specifically in open sourceprojects, to understand and modify – the software. Thorough testing enables the users(and the developers) to have confidence that the software they are using (ordeveloping) is going to function as expected.

These two areas are often overlooked or vary widely in the open sourcedevelopment process. Open source contributors tend to be more interested in codingthan documenting or testing. This is probably due to the nature of open source thattries to replace the formal testing process with "many eyeballs" effect in eliminatingthe bugs. Also, adding comments in the source code is often perceived as sufficient fordocumentation. There has been some effort in addressing the problem of lack ofdocumentation (e.g. the *Linux Documentation Project* [26] and *Mozilla Developer*

Documentation web page [27]), but this is still a rarity for smaller open sourceprojects. We have yet to find some sort of testing strategies for open source projects.They might exist, but implicitly and not open to the outside the project.

Licensing

The basic freedoms of open source software and how they differ from other softwaredistributions were discussed in section 2.1 and 2.4 earlier.

Operational support

In order to facilitate concurrent software development and fast controlled evolution,all open source projects implement some form of configuration management. This isenacted by using CVS, other tools, or even an ad-hoc solution using some web-basedsupport.The communication within communities related to specific open sourceprojects is done almost exclusively by electronic means, which are also used toorganise their work. The electronic means most commonly used are dedicated mailinglists, newsgroups, and web site. The exact structure and usage of web sites, mailinglists and newsgroups vary among open source projects.

Size

Size is not a distinctive measure in open source projects. Both involved-communityand code base sizes vary widely from project to project.

4 Conclusion

The term open source is being used at large in a vague manner not only in computing science community only but by other communities too, consequently creating confusion and misunderstandings. In ourefforts to understand open source we have done an extensive literature review,explored several web sites related to the topic, and interviewed some individuals andcorporations involved with open source. Our work was performed bearing multipledisciplines in mind.

We have determined many project characteristics that are relevant for opensource. Some of these characteristics are common to all efforts.

We understand that there is no waythat an absolute tick-list can ever be generated due to the variations that exist from one open source project to another, so additional variable characteristics may exist.

The various proprietary softwar and their equivalent open source software are categorized for better reference

Software Category	Proprietary Software	Equivalent Open Source Software
Operating System	Microsoft Windows	Linux Ubuntu
Browser	Internet Explorer	Mozila Firefox
Office automation	Microsoft Office	Open Office
Mathworks	MATLAB	Sci Lab
Graphics Tool	Adobe Photoshop	GIMP(GNU Image Manipulation Program
Drafting tool	Auto CAD	Archimedes
Web Editors	Adobe Dreamweaver	NVU
Desktop Publishing	Adobe Acrobat	PDF Creator
Blogs	Blogger	WOEDPRESS
Mobile	IOS	Android
Media Player	Windows Media Player	VLC Player
Database	Oracle ,Microsoft SQL Server	My SQL ,Mongo DB,HADOOP
Server	Microsoft Window Server	Red Hat Server,Ubuntu Server
Web Server	IIS	Apache

5 Future Work

Further there is need to clarify the exact difference between open source and free software, as well as create a table having the characteristics to describing the difference between open source software and free software.Somequestions required to be taken care off include:

- How to determine that the free or open source software is trust worthy ?
- Number of reviews can be studied before using the open source or free software, but still doesn't guarantees about the quality of open source or free software available.
- Mutual influencesneeded to be find between software architecture and groupstructure in open source or free software development?
- Is there any architectural decay in open source and free software?
- Is there responsibility attached with the developer or user to software developed as open source or free?

References

[1] "The Open Source Initiative: Open Source Definition", http://www.opensource.org/docs/definition.html.

[2] "The Linux Home Page at Linux Online", http://www.linux.org/.

[3] "The Apache Software Foundation", http://www.apache.org.

[4] "mozilla.org", http://www.mozilla.org/.

[5] E. S. Raymond, *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, O'Reilly & Associates, 1999.

[6] G. Moody, *Rebel Code: Linux and the Open Source Revolution*, The Penguin Press, 2001.

[7] C. Dibona, M. Stone, and S. Ockman, *Open Sources: Voices from the Open Source Revolution*, O'Reilly & Associates, 1999.

[8] A. Mockus, R. T. Fielding, and J. Herbsleb, "A Case Study of Open Source Software Development: The Apache Server," Proceedings of ICSE 2000, pp. 263-272, 2000.

[9] M. W. Godfrey and Q. Tu, "Evolution in Open Source Software: A Case Study," Proceedings of International Conference on Software Maintenance (ICSM'00), 2000.

[10] B. J. Dempsey, D. Weiss, P. Jones, and J. Greenberg, "A Quantitative Profile of a Community of Open Source Linux Developers", SILS TR-1999-05, 1999.

[11] H. Wang and C. Wang, "Open Source Software Adoption: A Status Report," *IEEE Software*, March/April, pp. 90-95, 2001.

[12] J. Feller and B. Fitzgerald, "A framework analysis of the open source software development paradigm," Proceedings of 21st International Conference on Information Systems, pp. 58-69, 2000.

[13] "SourceForge", http://sourceforge.net/.

[14] "Geocrawler", http://www.geocrawler.org/.

[15] "The Free Software Definition - GNU Project - Free Software Foundation (FSF)", http://www.fsf.org/philosophy/free-sw.html.

[16] "The Open Source Initiative: History of the OSI", http://opensource.org/docs/history.html.

[17] "Why Free Software is better than Open Source", http://gnu.metagensoft.com/philosophy/free-software-for-freedom.html.

[18] E. S. Raymond, "Homesteading the Noosphere", http://tuxedo.org/~esr/writings/homesteading/homesteading/.

[19] "Categories of Free and Non-Free Software", http://www.gnu.org/philosophy/categories.html.

[20] "K Desktop Environment Home", http://www.kde.org/.

[21] N. Slack, S. Chambers, C. Harland, A. Harrison, and R. Johnston, *Operations Management*, 2nd ed, Financial Times Pitman Publishing Series, 1998.

[22] A. Aoki, K. Hayashi, K. Kishida, K. Nakakoji, Y. Nishinaka, B. Reeves, A.Takashima, and Y. Yamamoto, "A Case Study of the Evolution of Jun: an Object-Oriented Open-Source 3D Multimedia Library," Proceedings of 23rd ICSE Conference, Toronto, Canada, pp. 524-532, 2001.

[23] R. N. Britcher, *The Limits of Software: People, Projects, and Perspectives*, Addison Wesley, 1999.16

[24] T. Bollinger, R. Nelson, K. M. Self, and S. J. Turnbull, "Open-Source Methods: Peering Through the Clutter," *IEEE Software*, July/August, pp. 8-11, 1999.

[25] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, Addison Wesley, 1998.

[26] "Linux Documentation Project", http://www.linuxdoc.org.

[27] "Mozilla Developer Documentation", http://www.mozilla.org/docs/.