Volume 5, Issue 5, May 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Open Source Software: A Review of Characteristics That Made it Popular and Successful

Prasad Devarasetty[1], Dr Satyananda Reddy Ch[2]
[1] Department of Computer Science and Engineering, DVR & Dr HS MIC College of Technology
Vijayawada, Andhra Pradesh, India
[2] Department of Computer Science and Systems Engineering, Andhra University
Visakhapatnam, Andhra Pradesh, India

Abstract: ***Open Source Software is an alternative to proprietary software. It is being popular day by day which has brought about an increase in research. It has been successful as the research in this area is growing rapidly and results published support this argument. Research is in progress to identify the characteristics unique to this phenomenon. There are a large set of characteristics that drive to the success of Open Source software. Some of them include the type of the development model, the phenomenon of forking that facilitates to achieve sustainability, the role of contributors and the attractiveness of OSS, the behaviors of Commits, and how the training phenomenon facilitates the acceptance of the OSS. This study elaborates how the five characteristics facilitate to the success and popularity of the Open Source Software.***

Keywords: Open Source software, Sustainability, forking, commits contributors, acceptance of Open source software

I. INTRODUCTION

Software plays an important role in our society. With this increasing popularity of software, its evolution and its style of development, it is categorized into two types, Open Source Software and Proprietary Software. Open Source Software is defined as the one which is developed by a set of people and its evolution is due to various collaborators who spread all over the world and the right to ownership and enhancement is not specific to a single user. In contrast proprietary software is that category where the development is done under strict supervision with a set of closed people [3]. The licensing mechanism of Open Source Software is quite different from that of the proprietary software [13]. Here we try to present some of the reasons for the success and popularity of Open Source Software. We begin with a small discussion on the type of process model which OSS communities follow in general, the increasing role of Contributors i.e. the people who use and contribute to the development of the Open Source Software. This increase in the number of contributors or the increased activity of the contributors may lead to the release of various versions of the software. These contributors modify the existing software by contributing parts of it. These small contributions are called commits. After a commit completes the client revision control software informs of the new revision number and each successive commit increases the revision number by one. The Open Source Software code that is available may be forked to start a new development effort, and this right to fork the open source code is at the core of the Open source Licensing [16]. Code forking represents the single greatest tool available for guaranteeing sustainability of the Open Source Software. The last and important factor presented in this paper is related to the human behavior, the effects of training of these solutions and illustrates how this facilitates the adoption of the Open Source Software. The following sections describe in brief about the characteristics of the open source software which made it popular and successful.

II. THE DEVELOPMENT MODEL

The development model followed by the Open Source Software should confirm the release of constraints on the user, who has the flexibility to do necessary changes and customize the software and release it as a new version. This lifecycle model may be called as perpetual development lifecycle [4] which comprises the following major activities.

1. Continuous and steady development of the system adding new features all the time and this activity of adding new features is public
2. The development is done based on anticipated user needs and explicit user feedback.
3. Significantly when new functionality is accumulated, the continuous development is interrupted to prepare for a major release of a new production version.

Moreover there will be more common minor releases in each version which reflects bug fixes or security patches which enables several production versions in parallel.

In general the Open Source Software project begins with a personal need of a single developer who has a vision and tries solutions for his specific requirements. Then he starts discussion with his friends and colleagues about the possible solution and making the codebase. This code is then made available to others which attracts the attention of

other user developers and inspire them to contribute to the project. The contributors and the initiators collaborate with each other by sharing their ideas and enrich the product. These collaborators or co-developers help in rapid code improvement and effective debugging. As the project grows it attracts more and more contributors and the feedback helps in getting better understanding of the issue and possible strategies to evolve it. Now the projects community is established and will react to future changes in the same was it emerged originally.

Greger J. Ruthfuss in his research classified the various stages of Open Source Product development as Planning, Pre-Alpha, Alpha, Beta, Stable and Mature. The planning stage is where no code is written, it is only an idea having no concrete form. The second stage in the classification is the pre-alpha stage where the preliminary source code has been released where the code is not expected to compile or run. The next sate is the Alpha where the released code takes some shape and active work to expand the feature set of the application continues, and when the feature set is complete the beta stage is said to be achieved. Though the feature set is complete it may contain some faults in the beta stage. The software is said to be achieve stable state where it is useful and reliable enough for daily use. During this phase changes are applied very carefully with intent to increase the stability of the product. When no significant changes happen over a long time, and only minor issues remain, the project enters the next stage i.e... The mature stage. In this mature stage little or no development occurs. The project may remain in this final stage for a longer duration before it slowly fades into the background unless it gets forked to emerge as new software. This source code for mature projects remains available indefinitely, and may serve educational purposes.

The following figure illustrates the OSS development process proposed.

The development process of an Open Source Software project consists of the following visible phases out of which the first being the problem discovery followed with the identification of volunteers who strive to identify the solution and develop code and test it. After coding, the code is subject to changes when it is being shared to a number of contributors and this change is to be reviewed to perform final commit and documentation proceeding to the final phase, Release management where the post release activities are performed.

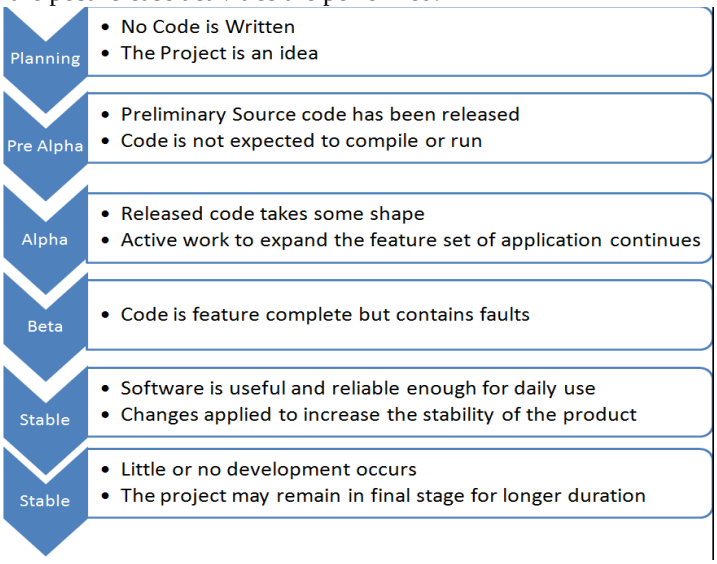

Fig 2.1: Open Source Development Model

III. FORKING AND ITS EFFECTS ON SUSTAINABILITY

Code forking represents the single greatest tool available for guaranteeing sustainability in Open Source Software. The term sustainability is defined as the possibility of an open source program to continue to serve the needs of its developers and users.

Fork is by meaning "Divide into branches that go separate ways". In the Software the word fork evolves from a system call which causes a running process to split itself into two identical copies that diverge to perform different tasks. Free and Open Source Software may be legally forked to create a totally new version of the software. Though there are many reasons for the projects to get forked the most common reason is the desire to modify the original program to better address a specific need [28]. The forks may also be planned temporary divergences intended to test new ideas and features with the intention of later integrating effective improvements back into the original. The right to fork code is built into the very definition of what it means to be an open source program. The Free Software Foundation's Free Software Definition (FSD) states that users have the freedom to "run, copy, distribute, study, change and improve the software. The possibility of forking any project affects the governance and sustainability of all Open Source Software. There could be several reasons for performing fork activity on the software. The following sections describe three levels that govern forking activity.

i. **Software Level**: By nature the software industry dictates that software cannot be stable and steady a longer period of time. Without continual adaptation and changes the program will be progressively become less satisfactory [11] and hence the key component of software sustainability can be identified as its ability to change and evolve. The revenue models of companies in the software marketplace often welcome elements such as system versioning encourage repurchases of a newer version of the same system, or vendor lock-ins that limit the customer choice to

certain providers of system or product [19]. Newer versions of programs my introduce compatibility problems with earlier operating systems or programs for example the lack of backward compatibility in Internet Explorer, Microsoft Office… More over some programs also cause compatibility issues with earlier versions of the program. By the very definition of open source the right to improve a program, the right to combine many programs, and the right to make a program compatible with other program and versions are its fundamental rights. In essence at the software level the right to fork protects against planned obsolescence, versioning and vendor lock-in and further forking and updating counters the disuse due to decay.

ii. **Community Level**: The Open Source Project can be forked at any point of time due to the shared ownership of the open source. Hence no person or group has a "magical hold over the project. One of the major concerns among the communities of Open Source is that a commercial vendor attempting to privatize a project's source code. To quote an example the 2008 acquisition of MySQL [29], an open source relational database management system by sun Microsystems and subsequent acquisition of Sun by Oracle is an example of a case involving community concern over potential hijacking. Researchers argued that such series of acquisitions would lead to the collapse of both MySQL and the open source movement at large. Responding to this the other researchers in their research noted that while open source companies can be bought, open source communities cannot. The original MySQL developer Michael Widenius forked the MySQL code and started a new version under a different name, MariaDB, due to concerns regarding the governance and future openness of the MySQL code. Another example to quote the community concerns regarding governance led to a forking of the OpenOffice to begin LibreOffice. This project emphasized the importance of a "transparent, collaborative and inclusive" government [30]. Analysis of the LibreOffice project indicates that this fork has resulted in a sustainable community with no signs of stagnation. Thus in common forking occurs due to a community's desire to create different functionality or focus the project in a new direction. These forks may be based on the difference in the software requirements or focus of the product. When they address these disparate community needs, different versions can prosper.

iii. Business-ecosystem Level: The right to fork means that any company can duplicate any competitor's open source software distributions and hence the competitive advantage cannot depend on the quality of the code alone. Though Open source software is free it is also increasingly developed and supported for commercial gains. To quote an example Red Hat, the world's open source leader is one such example. While the primary revenue of Red Hat is due to subscriptions and services related to their software reviews publish that the Red Hat's programs themselves are largely based on forks of programs by other developers. This process of combining forked programs is all made possible by forking of existing products and repackaging them as a new release. This feature is also implicitly applied in the hundreds of Linux distributions.

The saying "one man's trash is another man's treasure" is particularly apt for open source software development. To quote an example to this soon after Nokia's abandonment of the MeeGo project in 2011, the Finnish company Jolla announced that it would create a business around its revival and on July 16 2012 it announced a contract with D. Phone, one of the largest cell phone retailers in China and on November 21 they launched Sailfish OS and thus an abandoned project can become a business opportunity. Further forking can also aid companies who choose to use an existing program or develop it for personal use.

Thus it is the right to fork that drives the governance of open source projects and provides the dynamic vigor found in open source computing today and made the Open Source Software to sustain in the market.

IV. CONTRIBUTORS AND THE ATTRACTIVENESS OF THE OPEN SOURCE SOFTWARE

The Open Source Software products comprise groups of developers and users geographically dispersed but connected together through shared values and the internet. Each group of users and developers contribute a unique set of complementary resources to the OSS product. The users provide inputs including bug reports, suggestions of new features and translation of documentation and developers implement new features, fix bugs and deal with sponsors. The developers need users to inform their practices and users need developers to implement their requests. Though the literature primarily singles out developers as the main contributors to the success of Open Source Software [31], users also play an important role in the innovation process of Open Source Software. The third category of contribution is the role of visitors to the corresponding WebPages of the Open Source Software. The visitors who are technically inclined inspects the source code and post a suggestion or refer people to the project supporting the argument that visitors can contribute to the success of Open Source Software, though they may never use or get directly involved in developing the software. In addition to the above mentioned classification of contributors there are passive users, readers, bug reporters, bug fixers, peripheral developers, active developers, core members and project leaders.

Though the research in OSS gives a wide classification of contributors it is argued that visitors, users and developers are the key resources to the Open Source Software and attracting, retaining and inducing them to contribute are the core challenges for the success of Open Source Software [32]. The users who are spread all over the world provide relevant problems to be solved such as bug reports and enable to network externalities that not only increase the popularity of the project but also attract new and sustain developers contribution. The visitors of the software's webpage represent the brand exposure and commercial success contributes indirectly by enhancing the ranking of the webpage.

The Attractiveness of the OSS project is related to the popularity and visibility of a project, which increases the motivation of individuals to showcase their abilities and improve their reputation within both developers and business communities. The more the OSS community focuses on improving the attractiveness the more will be contributor's activity towards the software. The Free Open Source projects attractiveness significantly influences the likelihood of task

completion and the time for task completion. New users, visitors and developers are likely to choose the more attractive project in comparison to the others.

Today as the size of the Open Source Software developer and user community is large enough to contribute to the development and improve the attractiveness of it. Further the increased attractiveness of the project drags attention of the contributors' support the "Rich gets Richer" phenomenon.

V. COMMITS AND THEIR BEHAVIOR

Software development consists of making code contributions, also known as commits to a source code repository that hosts the project. This open source collaborative development among software developers from all over the world is a significant factor to the success of OSS projects [33]. In such multi developer works revision control and version control are essential whose main role is to track and provide control over changes to the source code. The developers generally use the tools such as CVS (Concurrent Version System), SVN (Subversion) and Git to manage and maintain different types of files stored in a repository that hosts the OSS project. After each commit the software informs us of the new revision number and each successive commit increases the revision number by one [21]. The commits can be classified into two types the first being the project level commits which are contributions towards the enhancement of the project which drive release of the new version of the project. The type is the file level commits which are performed at the file level. Further based on the size the commits can be classified into single commits, Aggregate Commits and Repository Commits.

A Single commit is a single individual developer contribution in which case a software developer makes a code contribution that ideally deals with exactly one semantic issue. Ex: fixing a bug. These single individual developer commits would be typically of size 1 – 100 Source Lines of Code. These constitute the major portion of the commit population as the process of development of open source software attracts contributors all over the world who are sparsely distributed can commit directly and as the commit size grows they cannot directly commit code but rather have to channel it through a committer. This is the reason why small commits dominate in the Open Source Software development phenomenon.

The Aggregate commit or the aggregate developer contribution is the case, a developer or a team of developers contributes a consolidated set of commits they had been working separately. Ex: Working in another repository. These aggregate commits can be particularly focused contributions like fixing a particular bug, implementing a feature... These represent larger chunks of work and may be the result of a single developer or a team of developers delaying commits to finish a complex feature or a development process using distributed configuration management systems. A typical aggregate commit could range from 101 to 10000 SLoC.

Repository commits or Repository Refactorings results when the contributors branch a whole library of a project. These repository commits may be the result of a large copy and paste activity like including an existing library, Branching or forking an existing project or the merging of separate repositories in distributed configuration management systems. A typical repository refactoring would be more that 10000 SLoC. The Open Source Software development process encourages all the above classification of commits liberally and probably this could be the reason for the enormous growth and evolution of the open source software and further the success and popularity of it.

VI. THE TRAINING AND ITS EFFECTS ON THE ACCEPTANCE OF OSS

The freedom to access the Open source software is divided into four essential parts [34]. i) The freedom to run the program for any purpose; ii) The freedom to study how the program works, and change how it does the computing as the user wishes; iii) The freedom to redistribute copies so that we can help our neighbors; iv) The freedom to distribute copies of our modified versions to others. The OSS licenses differentiate in the degree of restrictions imposed on the ability of the user to distribute modified versions [36] based on the concrete OSS. Various research studies produced results supporting the statement that the education and training of individuals have a relevant role on the development of the OSS solutions. The motivational factors of the OSS adoption can be divided into three groups; technological, economic and socio-political. Similarly in an enterprise environment three groups of motivation spread out are Organizational, technological and environmental [37]. Some other studies have analyzed how these organizations adopt these solutions. Six different ways in which the organizations adopt these are i) Deploying OSS products in their operational environment as end users. ii) Using OSS case tools in software development iii) Integrating OSS components into their own software systems iv) Participating in the development of OSS products controlled by another organization v) providing their own OSS products and relating to a community around the product vi) Using software development practices, often associated with OSS communities within a company or Consortium of companies [38].

Though there are a number of reasons and ways and reasons to adopt the OSS solutions this adoption would be far easier and effective if the end users have received training on these solutions and this will have a positive impact in the acceptance towards the OSS solution [35]. Training and educational programs may foster a feeling of self efficacy and emphasize the user friendliness towards the technology. OSS companies' and communities recommend the design of the user training program to increase the usage intentions of a technology. The OSS users recognize the usefulness of software when it is ready to user and hence when considering the OSS as utility solution users with technical expertise and training OSS appear to be important. Thus it is evident that in an educational environment based on the trainers support is an important factor which influences positively and directly on the usage behavior. Thus training the OSS before it comes to adoption to the industrial and commercial purposes also plays a key role in the popularity and success of the Open Source software.

VII. CONCLUSION

Open source software has a wide impact in the software industry and community. The popularity of the open source software is due to many reasons. We tried to present a few important characteristics that are at the back end to drive to its success. The open incremental development model that allows a number people to collaborate and contribute to the enhancement of the software is one among the key characteristics of the Open Source Software. The right to fork the Open Source Software is another one that leads to sustain the software over a period of time. The three different levels at which the forking can be performed are discussed quoting different cases. Further the contributors which could be the developers, users and even the visitors of the projects websites their role, and how the attractiveness influences them in the success and popularity of the Open Source Software is also discussed. The attractiveness of an Open Source Software project is related to the popularity and visibility of the project, which increases the motivation of the individuals to showcase their abilities and improve their reputation within both developers and business communities. The behavior of commit is another characteristic which is not overlooked in this study. The code contributions that are given by the collaborative developers and contributors across the world are also known as commits. The characterization of these commits, The project level and file level is done and based on their size they are classified into Single Commits, Aggregate Commits and Repository Commits which are being encouraged and supported by the OSS Community and this could be one among the reasons for its success. Finally the training component which has the direct influence on the acceptance and popularity of the OSS is reviewed, training of the OSS before its adoption to the industrial and commercial needs plays a key role and acts proportionately to the success and popularity of the Open Source Software.

This study outlined only a few characteristics of the Open Source Software that have been driving to its success. Further research should focus on the evidences that show to what extent these influence the OSS paradigm. Investigation is also needed to identify and evaluate various other characteristics that differentiate the Open Source Software development and the proprietary development phenomenon. Our focus is also on developing a measure considering the above and other characteristics yet to be investigated in detail to evaluate the performance and success of the Open Source Software.

REFERENCES

[1] Software Evolution – Tom Mens, Serge Demeyer-Springer-2008
[2] Three Strategies for Open Source Deployment: Substitution, Innovation and Knowledge Reuse- Jonathan P.Allen: IFIP International Federation for Information Processing
[3] Open-Source Vs Proprietary Software Jean-Michel Dalle, Nicolar Jullien
[4] The Linux kernel as a case study in software evolution: Ayelet Israeli,Dror G. Feitelson
[5] The Past Present and Future of Software Evolution- Michael W.Godfrey, Daniel M. German
[6] Open Source Approach in Software Development- Advantages and Disadvantages- Jovica Durkovic, Vuk Vukovic. Lazar Rakovic- Management information systems Vo.3 No.2
[7] " On Understanding Laws, Evolution and conservation in the large program Life cycle", Journal of Systems and Software.
[8] Bennet. K. H Rajlich, V.T: Software Maintenance and Evoluton: A Roadmap In: The Future of Software Engineering.
[9] Raymond. E.S: The Cathedra and the bazaar: musings on Linux and open source by an accidental revolutionary.
[10] Software Evolution: An Empirical study of Mozilla Firefox: Anita Ganpati, Dr. Arvind Kalia, Dr. Hardeep Singh. Int. J. Computer Technology & Applications, Vol 3.
[11] M.M. Lehman, "Programs, Life Cycles and Laws of Software Evolution", Proceedings of the special Issue of Software Engineering, IEEE, Vol.68, No.9, 1980
[12] Has open source software been institutionalized in organizations or not? Josianne Marsan, Guy Pare, Michael D. Wybo
[13] From Proprietary to open source-Growing an open source ecosystem Terhi Kilamo, Imed Hammoda, Tommi Mikkonen, Timo Aaltonen- Journal of Systems and Software -2011.
[14] Open source software:The effects of training on acceptance M. Dolores Gallego, Salvador Bueno, R. Jose Racero, Jan Noyes- Computers in Human Behavior
[15] Dynamics of Open-Source Software Developer's Commit Behavior: An Empirical Investigation of Subversion Yutao Ma, Yang Wu, Youwei Xu
[16] Code Forking, Governance, and Sustainability in Open Source Software Linus Nyman and Juho Lindman-Technology Innovation Management Review- January 2013.
[17] Users' perception of open source usability: an empirical study- Arif Raza, Luiz Fernando Capretz , Faheem Ahmed
[18] Sustainability in Open Source Software Commons: Lessons Learned from an Empirical Study of SourceForge Projects- Charles M. Schweik- Technology Innovation Management Review January-2013
[19] The attraction of Contributors in free and open source software projects- Carlos Santos, George Kuk, Fabio Kon, John Pearson.
[20] The Commit size distribution of Open source software-OliverArafat, Dirk Riehle-Proceedings of the 42nd Hawaii international Conference on System Sciences-2009
[21] Dynamics of Open-Source Software Developer's Commit Behavior: An Empirical Investigation of Subversion-Yutao Ma, YangWu, Youwei Xu.

[22] Open source software : The effects of training on acceptance- M. Dolores Gallego, Salvador Bueno, F Jose Racero, Jan Noyes
[23] Understanding the open source software development- Feller.J & Fitzgerland B
[24] Exploring the determinants of the OSS MARKET POTENTIAL: The case of the Apache webserver Lakka S., Michalakelis, C., Varoutas, D., & Martakos, D Telecommunications Policy,36(1),51-68.
[25] The evolution of the OSS governance a dimensional comparative analysis. Noni, I D Ganzaroli, A & Orsi, I… Scandinavian Journal of Management
[26] Open source software licenses strong-copyleft, non-copyleft or somewhere in between? Sen, R., Subramaniam, C., & Nelson, M.L, Decision Support Systems
[27] Understanding the Differences between Proprietary & Free and Open Source Software D Prasad, Dr.Ch.SatyanandaReddy Int. Journal of Engineering Research and Applications (IJERA)Vol. 3, Issue 4, Jul-Aug 2013, pp.2295-2299.
[28] To Fork or Not to Fork: Fork Motivations in SourceForge Projects
Linus Nyman and Tommi Mikkonen International Journal of Open Source Software and Processes (IJOSSP), 2011, Volume 3, Issue 3.
[29] www.mysql.com
[30] Sustainability of Open Source software communities beyond a fork: How and why has the LibreOffice project evolved? Jonas Gamalielsson, Björn LundellThe Journal of Systems and Software.
[31] Open Source Software User Communities: A Study of Participation in Linux User GroupsRichard P. Bagozzi, Utpal M. Dholakia,Management science
[32] The attraction of contributors in free and open source software projects carlos Santos, George Kuk, Fabio Kon, John Pearson. Journal of strategic Information Systems March 2013.
[33] Free/Libre Open Source Software Development: What we Know and What We Do Not Know. ACM Computing Surveys, Feb2012.
[34] www.gnu.org
[35] Open source Software: The effects of training on acceptance M. Dolores Gallego, Salvador Bueno, F. Jose Racero, Jan Noyes. Computers in Human Behavior: 2015
[36] Sen, R., Subramaniam, C., & Nelson, M. L (2011), Open Source Software licenses strong-copyleft, non copy-left or somewhere in between? Decision Support Systems.
[37] Qu, W. G., Yang, Z., & Wang, Z(2011). Multilevel framework of open source software adoption. Journal of Business Research.
[38] Hauge, O., Ayala. C., & Conradi, R (2010). Adoption of open source software in software- intensive organizations-A Systematic literature review. Information and Software Technology
[39] What's a Typical Commit? A Characterization of Open Source Software Repositories. Abdulkareem Alali, Huzefa Kagdi, Jonathan I. Maletic.
[40] en.wikipedia.org/wiki/Committer

