Measurement of Processes in Open Source Software Development

Parminder Kaur*
Hardeep Singh**

Abstract

Purpose: *This paper attempts to present a set of basic metrics which can be used to measure basic development processes in an OSS environment.*

Design/Methodology/Approach: *Reviewing the earlier literature helped in exploring the metrics for measuring the development processes in OSS environment.*

Results: *The OSSD is different from traditional software development because of its open development environment. The development processes are different and the measures required to assess them have to be different.*

Keywords: *Open Source Software (OSS); Free Software; Version Control; Open Source Software Metrics; Open Source Software Development*

Paper Type: *Conceptual*

Introduction

Free software [FS], term given by Richard Stallman, introduced in 1984, can be obtained at zero cost i.e. software which gives the user certain freedoms. Open Source Software (OSS), term coined by Eric Raymond, in 1998, is software for which the source code is freely and publicly available, though the specific licensing agreements vary as to what one is allowed to do with that code . In the case of FS, only executable file is made available to the end user, through public domain and end user is free to use that executable software in any way, but the user is not free to modify that software. The alternative term Free/Libre and Open Source Software (FLOSS) refers to software whose licenses give users four essential 'freedoms':

- To run the program for any purpose,
- To study the workings of the program, and modify the program to suit specific needs,
- To redistribute copies of the program at no charge or for a fee, and
- To improve the program, and release the improved, modified version **(Perens, 1999; 2004)**.

The free software movement is working toward the goal of making all software free of intellectual property restrictions which hamper technical improvement whereas OSS users do not pay royalties as no copyright exists, in contrast to proprietary software applications which are strictly

* *Department of Computer Science and Engineering, Guru Nanak Dev University, Amritsar-143005, India. email: parminderkaur@yahoo.com*

** *Department of Computer Science and Engineering, Guru Nanak Dev University, Amritsar-143005, India. email: hardeep_gndu@rediffmail.com*

protected through patents and Intellectual Property Rights (IPR's) **(Asiri, 2003; Wheeler, 2003)**. OSS is software for which the source code is publicly available, though the specific licensing agreements vary as to what one is allowed to do with that code **(Stallman, 2007)**.

Open Source Software Development

Open Source Software Development (OSSD) produces reliable, high quality software in less time and with less cost than traditional methods. **Adelstein (2003)** is even more evangelical, claiming that OSSD is the *"most efficient"* way to build applications. **Schweik and Semenov (2003)** add that OSSD can potentially *"change, perhaps dramatically, the way humans work together to solve complex problems in computer programming".* Even when there is a great level of exaggeration, OSS can be used as an alternative to traditional software development. **Raymond (1998)** compares OSSD to a *"bazaar"* – a loosely centralized, cooperative community where collaboration and sharing enjoy religion status. Conversely, traditional software engineering is referred to as a *"cathedral"* where hierarchical structures exist and little collaboration takes place.

- **Problems with Traditional Development**

Traditional software development projects suffer from various issues such as time and cost overruns, largely unmaintainable, with questionable quality and reliability. The 1999 Standish Group report revealed that 75% of software projects fail in one or more of these measures, with a third of projects cancelled due to failure. In addition, systems often fail to satisfy the needs of the customer for whom they are developed **(Sommerville, 1995)**. These failures are ascribed to:

- Inadequate understanding of the size and complexity of IS development projects coupled with inflexible, unrealistic timeframes and poor cost estimates **(Hughes & Cotterell, 1999; McConnell, 1996)**.
- Lack of user involvement **(Addison & Vallabh, 2002; Frenzel, 1996; Hughes & Cotterell, 1999; McConnell, 1996)**.
- Shortfalls in skilled personnel **(Addison & Vallabh, 2002; Boehm, 1991; Frenzel, 1996; Hughes & Cotterell, 1999; Satzinger, Jackson & Burd, 2004)**.
- Project costs are further increased by the price of license fees for software and tools required for application development as well as add-on costs for exchange controls.

- **Benefits of Open Source Software**

The benefits with OSS **(Feller & Fitzgerald, 2001; FLOSS Project Report, 2002)** are as follows:

- Collaborative, parallel development involving source code

sharing and reuse
- Collaborative approach to problem solving through constant feedback and peer review
- Large pool of globally dispersed, highly talented, motivated professionals
- Extremely rapid release times
- Increased user involvement as users are viewed as co-developers
- Quality software
- Access to existing code

Despite these benefits, perceived disadvantages of OSS are:

- In the rapid development environment, the result could be slower, given the absence of formal management structures **(Bezroukov, 1999; Levesque, 2004; Valloppillil, 1998).**
- Strong user involvement and participation throughout a project is becoming problematic as users tend to create bureaucracies which hamper development **(Bezroukov, 1999)**.
- OSS is premised on rapid releases and typically has many more iterations than commercial software. This creates a management problem as a new release needs to be implemented in order for an organization to receive the full benefit **(Farber, 2004)**.
- The user interfaces of open source products are not very intuitive **(Levesque, 2004; Valloppillil, 1998; Wheatley, 2004)**.
- As there is no single source of information as well as no help desk therefore no 'definitive' answers to problems can be found **(Bezroukov, 1999; Levesque, 2004)**.
- System deployment and training is often more expensive with OSS as it is less intuitive and does not have the usability advantages of proprietary software.

➢ **Open Source Software Development Models**

There are several basic differences between OSSD and traditional methods. The System Development Life Cycle (SDLC) of traditional methods have generic phases into which all project activities can be organized such as planning, analysis, design, implementation and support **(Satzinger, Jackson & Burd, 2004)**. Also, open source life cycle for OSSD paradigm demonstrates several common attributes like parallel development and peer review, prompt feedback to user and developer contributions, highly talented developers, parallel debugging, user involvement, and rapid release times.

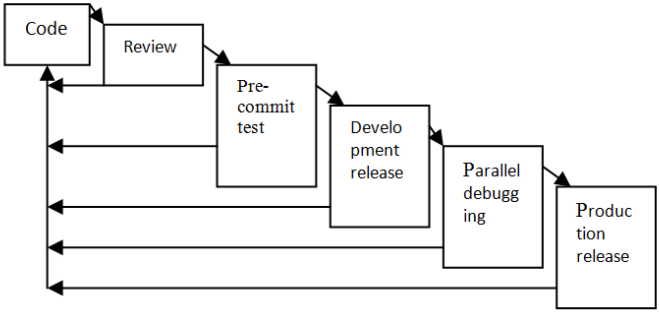
Vixie (1999) holds that an open source project can include all the elements of a traditional SDLC. Classic OSS projects such as BSD, BIND

and SendMail are evidence that open source projects utilize standard software engineering processes of analysis, design, implementation and support.

Mockus, Fielding & Herbsleb (2000) describe a life cycle that combines a decision-making framework with task-related project phases. The model comprises six phases like *roles and responsibilities, identifying work to be done, assigning and performing development work, pre-release testing, inspections, and managing releases*.

Jorgensen (2001) provides a more detailed description of specific product related activities that support the OSSD process. The model **(Fig. 1)** explains the life cycle for changes that occurred within the Free BSD project.

Fig.1: Jorgensen Life-Cycle, 2001



Jorgensen's model is widely accepted **(Feller & Fitzgerald, 2001; FLOSS Project Report, 2002)** as a framework for the OSSD process, on both macro (project) and micro (component or code segment) levels. However, flaws remain. When applied to an OSS project, the model does not adequately explain where or how the processes of planning, analysis and design take place.

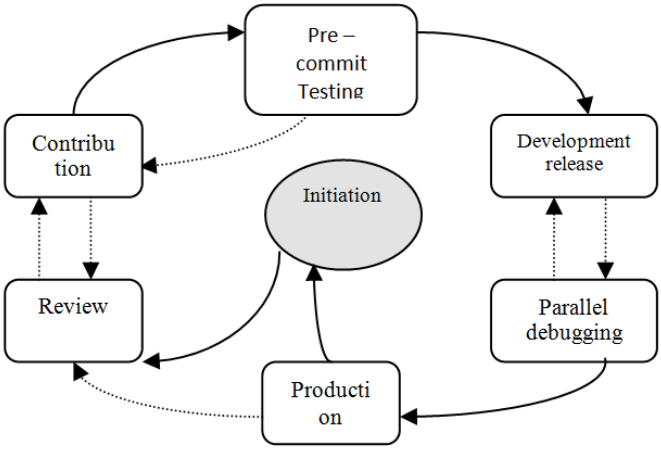
Schweik and Semenov (2003) propose an OSSD project life cycle comprising three phases: *project initiation; going 'open'; and project growth, stability or decline.* Each phase is characterized by a distinct set of activities.

Wynn (2004) proposes a similar open source life cycle but introduces a *maturity phase* in which a project reaches critical mass in terms of the numbers of users and developers it can support due to administrative constraints and the size of the project itself.

Roets, *et al.* (2007) expands Jorgensen life-cycle model and incorporates aspects of previous models, particularly that of **Schweik and Semenov (2003)**. In addition, this model attempts to encapsulate the phases of the traditional SDLC **(Fig. 2)**. This model facilitates OSS development in terms

of improved programming skills, availability of expertise and model code as well as software cost reduction.

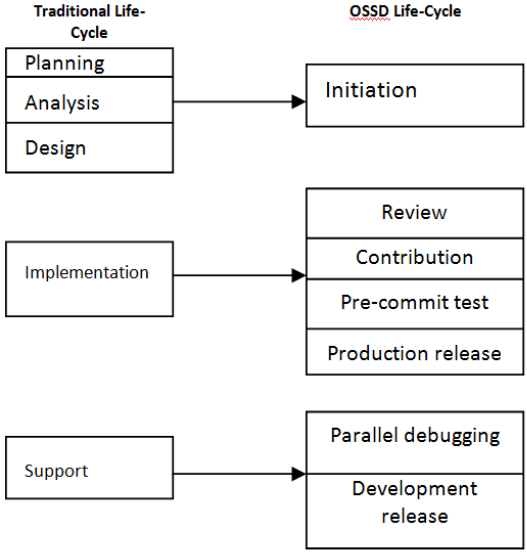
Fig. 2: Roets, et al. life-cycle model of OSSD projects, 2007



➢ **Comparison of Traditional Life-Cycle with OSSD Life-Cycle**

Fig. 3 compares different phases of traditional software development life-cycle with OSSD life-cycle mentioned in **Fig.2**.

Fig. 3: Comparison of Traditional Life-Cycle with OSSD Life-Cycle



Initiation phase of OSSD life –cycle combines three phases i.e. planning phase, analysis phase and design phase of traditional software development life-cycle. As it is suggested that it may be more important to get design right prior to actual programming so that all developers are working towards a clearly defined common purpose. Implementation phase combines the different aspects like review, contribution, pre-commit test and release of production. As multiple users as well as skilled personnel are involved in OSSD, parallel debugging and different versions of one piece of code can be grouped together with support phase of traditional software development life-cycle.

Proposed Metrics for the Selected Model

Keeping in view OSSD life - cycle model proposed by **Roets, *et al.* (2007)**, a following set of metrics is proposed to keep a check over the generation of multiple processes in OSSD.

- **Total Number of Contributions**

Under the considered model, a large number of users in an open environment contribute towards the development of the project. This metrics assesses the total number of contributors for the projects. This number can be a number of unique contributors or some contributors may be associated with multiple projects. However, this metric is related to the number of contributors for a given project irrespective of their affiliations to other projects.

- **Average Domain Experience of Contributors**

A particular project is developed on a specific domain. Usually the contributors having some expertise and experience in that domain area contribute to the project. This metric helps in evaluating the average experience of all the contributors taken together and can be represented as

Cumulative Experience of Contributor i.e.

$$E = \sum_{i=1}^{n} e_i$$

[Where e_i is the experience of an individual contributor in that domain]

Average Experience of Contributors

i.e. $E_{avg} = E / N$

[Where N is total number of contributors]

This metrics tends to measure the extent of support to the development of a project by the contributors. It is safe to assume that greater the average experience of a contributor, more robust development of the project would be.

- **Average Time for a Completion of a Version of Project**

Quick versioning is the essence of OSS development. However, different versions at different rates of time depending upon the factors like number of contributors, their experience, nature of project, complexity of the project etc. The average time for a completion of a version of project can be calculated as:

$$\textbf{Average Time i.e. } T_{avg} = T_{total} / N_{version}$$

[Where T_{total} is total time taken to develop all the versions and $N_{version}$ is the total number of versions generated]

Greater T_{avg} would indicate software development processes resulting from various factors like low number of contributors, their lack in experience or complexity of the project etc.

- **Bugs Track per Version**

Quality of OSS is always a question. However, with proper bug tracking mechanism and tools in place, the bug tracking can be made very effective and the quality of OSS can be enhanced. The number of bugs tracked per version is an indication of quality and reliability of the product. Hence, this measure can be put to an effective use for enhancing the quality of the final product.

- **Patch Accept Ratio**

Every contributor sends a patch for the enhancement of the product. However, it is not necessary that every patch sent by the contributor(s) is accepted for updating the product. The Patch Accept Ratio i.e. P_{ratio} can be defined as:

$$\mathbf{P_{ratio}} = \frac{\text{Total no. of patches accepted}}{\text{Total no. of patches accepted}}$$

A high Patch Accept Ratio shall effectively argue for a high competence of contributors and reverse is true for the less patch ratios.

- **Number of effective Reviews Received**

In addition to the development of patches, some contributors send their reviews about the products in making. A large number of effective reviews indicate that some functionality was not taken care of by either the developing contributor or the mentors. Greater the number of effective reviews more is the gaps in the development process. Therefore, the number of effective reviews can result in an effective developmental methodology.

Conclusion

The OSSD is different from traditional software development because of its open development environment. The development processes are

different and the measures required to assess them have to be different. This paper attempts to present a set of basic metrics which can be used to measure basic development processes in an OSS environment. However, these need to be validated and established by using them on large number of systems.

References

Addison, T., & Vallabh, S. (2002). Controlling software project risks - an empirical study of methods used by experienced project managers. In *Proceedings of the 2002 annual research conference of the South African institute of computer scientists and information technologists on Enablement through technology (SAICSIT '02) (128-140).* Republic of South Africa: South African Institute for Computer Scientists and Information Technologists. Retrieved from http://dl.acm.org/citation.cfm?id=581506.581525

Adelstein, T. (2003). *How to misunderstand open source software development*. Retrieved from http://www.consultingtimes.com/ossedev.html

Asiri, S. (2003). Open source software. *SIGCAS Computers and Society, 33* (1), 2. doi: 10.1145/966498.966501

Bezroukov, N. (1999). *Open source software development as a special type of academic research: Critique of vulgar Raymondism. First Monday*. 4 (10). Retrieved from http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/696/606

Boehm, B. (1991). Software risk management: principles and practices. *IEEE Software, 8*(1), 32-41.

Farber, D. (2004). *Six barriers to open source adoption*. Retrieved from http://www.zdnetasia.com/six-barriers-to-open-source-adoption-39173298.htm

Feller, J., & Fitzgerald, B. (2001). *Understanding open source software development.* London: Addison-Wesley.

FLOSS Project Report. (2002). *Floss Project Report: Free/Libre and open source software (FLOSS): Survey and study*. Retrieved from http://www.infonomic.nl/floss/report/

Frenzel, C. (1996). *Management of Information Technology (2nd ed).* Cambridge, MA: CTI

Hughes, B., & Cotterell, M. (1999). *Software project management (2nd ed)*. Berkshire, United Kingdom: McGraw-Hill.

Jorgensen, N. (2001). Putting it all in the trunk: Incremental software development in the FreeBSDopen source project. *Information*

Systems Journal, 11(4), 321-336. doi:10.1046/j.1365-2575.2001.00113.x

Levesque, M. (2004). *Fundamental issues with open source software development*. *First Monday*. 9 (4). Retrieved from http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/1137/1057

McConnell, S. (1996). Avoiding Classic mistakes. *IEEE Software, 13*(5), 111-112. doi: 10.1109/52.536469

Mockus, A., Fielding, R. T., & Herbsleb, J. D. (2000). Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology, 11*(3), 309-346. doi:10.1145/567793.567795

Perens, B. (1999). The open source definition. In M. Stone, S. Ockman & C. Dibona (Eds.), *Open sources: Voices from the open source revolution.* Sebastopol, California: O'Reilly & Associates.

Perens, B. (2004). *The open source definition*. Retrieved from http://opensource.org/docs/def_print.php

Raymond, E. (1998). *The cathedral and the bazaar*. *First Monday*. 3 (3). Retrieved from http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/1488/1403

Roets, Minnaar., *et al*. (2007). *Towards Successful Systems Development Projects in Developing Countries.* In *Proceedings of the 9th International Conference on Social Implications of Computers in Developing Countries*, São Paulo, Brazil, May 2007.

Satzinger, J. W., Jackson, R. B., & Burd, S. D. (2004). *Systems Analysis and Design in a Changing World(3rd ed)*. Boston: Course Technology.

Schweik, C. M., & Semenov, A. (2003). *The institutional design of open source programming: Implications for addressing complex public policy and management problems*. Retrieved from http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/1019/2426

Sommerville, I. (1995). *Software Engineering (5th ed).* Harlow: Addison-Wesley Longman Limited.

Stallman, R (2007). *Why "Free Software" is better than "Open Source"*. Retrieved from http://www.gnu.org/philosophy/free-software-for-freedom.html

Valloppillil, V. (1998). *Open source Initiative (OSI) Halloween I: A (new?) software development methodology*. Retrieved from http://www.opensource.org/halloween/halloween1.php#comment28

Vixie, P. (1999). Software Engineering. In M. Stone, S. Ockman & C. Dibona (Eds.), *Open sources: Voices from the open source revolution*. Sebastopol, California: O'Reilly & Associates.

Wheatley, M. (2004). *Open Source*: *The myths of open source*. *CIO, March 01, 2004.* Retrieved from http://www.cio.com/article/32146/Open_Source_The_Myths_of_Open_Source

Wheeler, D. A. (2003). *Why open source software/free software (OSS/FS)? Look at the numbers!* Retrieved from http://www.dwheeler.com/oss_fs_why.html

Wynn, Jr. D. E. (2004). Organizational structure of open source projects: A life cycle approach. In *proceedings of the 7th Annual Conference of the Southern Association for Information Systems*, Savannah. Retrieved from http://sais.aisnet.org/2004/.%5CWynn1.pdf