Comprehending co-evolution of OSS projects: Analytical methods and tool support

Dr M.M. Mahbubul Syeed.

1. Introduction

Open Source Software (OSS) development has become a powerful mechanism for developing and distributing IT applications. A number of IT pioneers are now putting serious interest and investment in favor of OSS movement which supports OSS to gain substantial market credibility and legitimacy [1].

Often an OSS project consists of a wide range of components, coming with a large number of versions reflecting their development and evolution history. Broadly, such components can be classified into two dimensions: software artifacts and the community surrounding it. Software artifacts consist of, for example, source code repository, bug reports, mailing list, and change logs; whereas the community can be classified into developer and user community.

OSS development is a dynamic process, thus the software keep changing over time. These changes constitute different form of contributions, such as modification, improvement and extension to the software (e.g., a bug fix, reporting a bug, or adding new functionalities). This continuous process of change and modification is termed as software evolution [12]. As described in [2], the successful evolution of an OSS has been made possible (among other factors) by their attraction of large communities of both developers and users. In such communities, users often initiate the need for changes and the developers implement it in the form of contributions [3]. In OSS realm, such contributions drive the software evolution, which in turn re-define the role of these contributing members and change the social dynamics of the OSS community. Thus, the evolution of the software and the community in an OSS project are mutually dependent and need to evolve amicably for the long term sustainability of an OSS project. This interdependent nature of evolution of the code and the community in OSS projects can be effectively termed as co-evolution of the code (i.e., software) and the community.

To measure the extent to which an open source project is successful has often been evaluated empirically by measuring endogenous characteristics. Such attempts focus on either the software evolution or the community evolution. A little attention is paid to date to study the pattern of co-evolution under one platform. This lagging in current research on OSS evolution shapes the necessity of this thesis.

2. Motivation

Study of the evolutionary pattern of OSS projects can be classified into four facets: software evolution, prediction, community evolution, and co-evolution [10]. In brief, software evolution explores the evolutionary behavior of OSS systems and derives patterns of evolution to evaluate it against the laws of software evolution. The prediction studies define and examine models to simulate the evolution of OSS projects. Community evolution studies explore the social dynamics of the developer and the user communities, whereas co-evolution studies and explores the interdependency between the code and the community during the evolution of the project. Thus, study of co-evolution would enlighten the maintainability, sustainability, and quality issues of the OSS project more effectively than studying them in isolation.

Our recent literature review (SLR) [10] on state-of-the-art of the evolution studies of OSS projects reported that around 80% of the work focuses only software evolution, with a minimal focus on the community and co-evolution. Although such studies provide valuable insights of OSS evolution, yet the evolution phenomenon of OSS remains incomplete without understanding the structure and evolution of the associated communities and their co-existence. In favor to this statement, in [4], it is argued that a project without a sustainable community cannot survive long-term and the collaboration pattern within this community drives the system evolution.

None-the-less, the growing dominance of OSS in different application domains provides a viable alternative to the commercial organizations and governments than that of in-house software. To mention a few, Linux, Ubuntu, Apache, Firefox, LibreOffice, Android, OpenGPS, Chrome, Bind, VLC, FFMpeg, Eucalyptus have initiated their ascendancy on respective application domains. Yet, an OSS project is volunteer driven, and has unconventional

community (i.e., organizational) structure and practices. Hence to adopt an OSS, the commercial organizations and business community often like to assess and understand the quality, reliability, maintainability and also survivability issues of an OSS project. This can be achieved most effectively through the analysis of the coevolution of the code and the community [6]. Also volunteer programmers might be interested in entering a project that has high chances to survive than to fail and be abandoned. Similarly, universities incorporating OSS related courses in their curriculum are encouraging students to participate in a successful, to be continued, project.

Current research focusing co-evolution of OSS projects mainly proposes a metric set [5] to evaluate the co-evolution of OSS projects, and performs a case study [7] on OSS projects to define collaboration models for the co-evolutionary behavior. But these results are primitive and require further investigation and verification for external validity.

Also there exists a need for a generic platform to automate the analysis of OSS evolutionary behavior, as most of the studies to date are empirical studies with either quantitative or qualitative data analysis [10]. This will provide a hands-on tool to the users of OSS projects to assess the evolutionary behavior from different perspectives.

3. Objective

The primary target of this research is to investigate the extent to which the two dimensions of OSS projects, e.g., software and the community, influence and overlap each-other during the evolution of the project, and ascertain their implications on OSS projects sustainability, survivability, maintainability and quality issues. The fundamental point of argument of this research can be stated as follows,

"*The evolution of the Open Source Software (OSS) project is constrained by the non-orthogonal evolution of the software and the community surrounding it*".

Under the hood of this focus the study examines the following set of research questions.

- What are the endogenous and exogenous drivers in the OSS projects that constitute and drive the co-evolution?
- What correlations can be established between the co-evolutionary pattern (i.e., drivers) of OSS projects and their evaluation criterion, e.g., long term sustainability, maintainability and quality?
- What kind of metrics and collaboration models can be defined for modeling and generalizing the co-evolution of sustainable OSS projects?
- How to analyze, conceive and exhibit the co-evolutionary pattern of OSS projects?

4. Research Methodology

For this research, case study research method is applied. This section put a detail discussion on the rationale for the selection of this research method, and on the protocol defined for carrying out the case studies.

Case Study research method

Case study can be defined as an empirical method aimed at investigating contemporary phenomena in their context [15][16][17][18]. This method provides a flexible means of studying phenomena when the boundary between the phenomenon and its context is unclear [17].

The suitability of case study as a research method in the domain of software engineering has gained substantial acceptance due to the followings [15], first, software development is a heterogeneous process consisting of individuals, groups and organizations, and has social and political issues. There exist several contemporary areas and research questions in software engineering that can be effectively investigated and reported through case study. Second, it is often hard to clearly state the boundary between phenomenon and its context in software engineering research [19]. In this regard, case study offers a flexible approach that does not need a strict boundary between the studied object and its environment. Third, case study contains essence of many other research methods which make it possible to apply such research methods under the case study framework [15]. For example, survey, systematic literature review, interviews, observations, and archival analyses can be performed as a part of case study research.

Relevance of case study method for this research

The target population of this research is the Open Source Software (OSS) Projects and their uses. It is already known that OSS projects are complex to analyze and comprehend due to their unconventional organizational structure and practices [6]. Also the project setup (e.g., communication mechanisms, data management tools and strategies) often varies significantly from project to project [9]. Case study research method as a flexible type, most effectively cope with such complex and dynamic characteristics of OSS projects [15]. Also, in a case study the results are derived from a clear chain of evidence (e.g., following a case study protocol), with either qualitative or quantitative data analysis. Research data for this method is collected from multiple sources in a planned and consistent manner. These characteristics of case study research method effectively increases the validity, traceability and reproducibility issues of the findings in OSS project studies.

This case study research can be categorized as a combination of exploratory and explanatory/ confirmatory case study [20], as the target of this work is to investigate and explore the evolution of the code and community dimensions in OSS projects in order to identify interdependency and influence of each other, and ascertain their implication on the quality parameters of OSS projects. For data collection and analysis we used quantitative data analysis.

Design of the Case Study

Following the suggestions and guidelines to performing case study research provided in [15], we first derived a case study research protocol. This protocol constitutes the plan of actions that must be accomplished in sequence for carrying out a case study. The correctness and precision of each action in the protocol is assessed and ensured against the study checklists provided in [15]. The study protocol is presented in Figure 1. The phases in this protocol are described below,

Define the objective and the research questions: The objective and the research questions for this research is defined and discussed in section 3.

The case and the selection strategy: Considering the focus of this research we stick to the evolving and well established OSS Projects in which the population of the developer community is over the critical mass [23].

Methods for data collection: To measure the evolution of OSS projects, current research work collects and utilizes OSS project data mostly from two sources. First source is the historical data repositories that are maintained and make available by the OSS project itself (e.g., source code repositories, change logs, mailing list archives, bug reporting systems, registered user information system, user forums, wiki entries). The other source is the third party sources that collect, organize and made available OSS projects data for public use and research. (e.g., source-forge, ohloh) [10]. For this research, the former approach is followed, i.e., the data is collected directly from project sources. Because collecting data from original sources is most flexible as the data that is most suitable for the research questions under investigation can be collected and analyzed [15]. Nevertheless, this form of data collection (termed as third degree data collection in [21]) is criticized due to the validity and completeness of the data for the research. Because the data has been collected and recorded for another purpose than that of the research study [22]. We tried to mitigate this issue by utilizing those data sources of OSS projects that are most popularly explored for evolution studies and are well accepted by the research community.

We also collected the research data from different data sources. For example, developer's contribution in a project is derived from the data collected from source code repository, change logs, email entries from development branch and bug reporting systems. In case study research, this form of data collection is termed as triangulation and is important for strengthening the conclusions derived from the collected data [15].

Finally, to keep the relevance of data collection (e.g., what data to collect for what purpose) we use the Goal-Question-Metric method (GQM) [22]. Within this method, we first formulate the research goals, which is followed by the formalization and formulation of research questions, and finally derived metrics based on the questions. Selective data is collected against the metrics. Thus the quality of the collected data is controlled and unnecessary data collection is restricted [15].

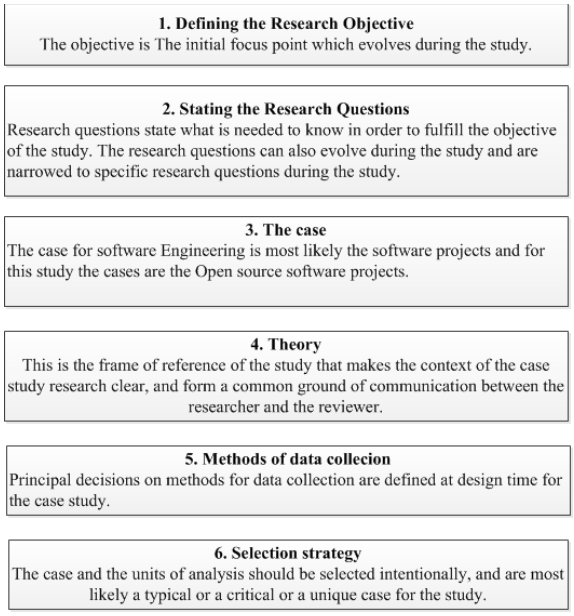

Figure 1: Case Study Protocol

5. Current Progress

The thesis work is now just over three and a half years in running. All the works done so far is carried out by the author in a close tie with his PhD supervisors. Current progress of this work produced two journal publications and five conference publications. Two more journal publications are currently under review and three conference papers are under correction phase. Following is the brief description on current progress of the work.

Literature reviews were performed at the beginning of this thesis work. The main concern was to have a hands-on guide that identifies the main contributions in the field of OSS project evolution studies and distill recommendations for future research. Two full scale literature reviews (SLR) were conducted, first one concerns the predictability of OSS project evolution and the other, covers OSS evolution concerning software, community and co-evolution. This work produced two articles, one conference article [8] and one journal article [10].

An exploratory ***Case study*** was conducted for existential identification of the co-evolutionary behavior in OSS projects. Based on this work a journal article was published [6]. This study posted five research questions and were investigated through mining OSS repository data (e.g., FFMpeg and Eucalyptus) and analyzing them through comprehensive set of methods and meta-models and corresponding tool implementation.

Two full scale ***empirical researches*** on the pattern of socio-technical congruence were conducted through quantitative analysis of the OSS evolution data. Socio-technical congruence is the approach used frequently to study the co-existence and bilateral influence of the product and the organization in forming a sustainable evolution of the project. We studied BSD group OSS projects and publications [24] [25] were made out of these works.

Constructive research. In order to analyze and comprehend evolutionary patterns of OSS projects, two tools **Binoculars** and **Pomaz** is implemented. These tools provide novel approaches to conceptualize the socio-technical dependencies within OSS projects using graph based data representation and visualization. Two conference articles were published based on these works [9][25]. A video demonstration of **Binoculars** tool can be found in [11].

6. Conclusion

Open Source Software provides a viable alternative of its commercial counterpart. Yet, users of such OSS projects want to confirm the sustainable evolution of the software and its surrounding community, due to its unconventional and heterogeneous organizational structure and practices. A plethora of studies were carried out to uncover evolutionary behavior of OSS projects, mostly focusing either the code or the community. But our point

of argument is that, in OSS domain, both the code and community has their impact on each other and should evolve amicably for the sustainability of the both. This research aims to put an intensive exploration on this issue, and identify facts behind it quantitatively.

References

1. Grewal R, Lilien GL, Mallapragada G (2006) Location, Location, Location: How Network Embeddedness Affects Project Success in Open Source Systems. Management Science 52:7: 1043–1056

2. Beecher K, Capiluppi A, Boldyreff C (1997) Identifying exogenous drivers and evolutionary stages in FLOSS projects, JSS, 82:5, 739-750

3. Mockus A, Fielding R T, Herbsleb J (2002) Two case studies of open source software development: apache and mozilla, TOSEM, 11:3, 309–346

4. Weiss M, Moroiu G, Zhao P (2006) Evolution of open source communities, Open Source Systems, IFIP, 203:2132.

5. Wang Y, Guo D, Shi H (2007) Measuring the Evolution of Open Source Software Systems with their Communities, ACM SIGSOFT Software Engineering, 32:6

6. Syeed M.M., Altonen T, Hammouda I, Systa T (2011) Tool Assisted Analysis of Open Source Projects: A Multi-Faceted Challenge, International Journal of Open Source Software and Processes, 3(2), 43-78.

7. Nakakoji K, Yasuhiro Y, Nishinaka Y, Kishida K, Yunwen Y (2002) Evolution Patterns of Open-Source Software Systems and Communities, In Proceedings of the international workshop on Principles of software evolution, 76-85.

8. Syeed M.M., Kilamo T, Hammouda I, Systa T (2012) Open Source Prediction Methods: a systematic literature review, In Proceedings of 8th. IFIP International Conference of Open Source Systems, Springer.

9. Syeed M.M. (2012) Binoculars: Comprehending Open Source Projects through graphs, In Proceedings of 8th. IFIP International Conference of Open Source Systems, Springer.

10. Syeed M.M., Hammouda I, Systa T (2013) Evolution of Open Source Software: a systematic literature review, Journal of Software.

11. Binoculars Demo (2012) http://www.youtube.com/watch?v=cMoYq6JOpQE.

12. Scacchi W. (2003) Understanding Open Source Software Evolution: Applying, Breaking, and Rethinking the Laws of Software Evolution, Applying, Breaking, and Rethinking the laws of software evolution, John Wiley and Sons Inc.

13. Hasselbring W. (2006) Research Methods in Software Engineering, GITO mbH Verlag.

14. Yin R.K. (1984) Case study research: Design and methods, Newbury Park, CA:Sage.

15. Runeson p., Martin H. (2009) Guidelines for conducting and reporting case study research in software engineering, Empirical Software Engineering, 14(2), 131-161.

16. Benbasat I, Goldstein DK, Mead M (1987) The case research strategy in studies of information systems. MIS Q 11(3):369 - 386 doi:10.2307/248684

17. Yin RK (2003) Case study research. Design and methods, 3rd edn. London, Sage 18. Robson C (2002) Real World Research. Blackwell, (2nd edition).

19. Shull F, Basili V, Carver J, Maldonado JC, Travassos GH, Mendonca M, Fabbri S (2002) Replicating software engineering experiments: addressing the tacit knowledge problem, Proceedings on ESEM pp 7-16.

20. Robson C (2002) Real World Research. Blackwell, (2nd edition). 21. Lethbridge TC, Sim SE, Singer J (2005) Studying software engineers: data collection techniques for software field studies. Empir Softw Eng 10(3):311-341.

22. Van Solingen R, Berghout E (1999) The goal/question/metric method. A practical guide for quality improvement of software development. McGraw-Hill.

23. Scacchi, W. (2004) Free and open source development practices in the game community, Software, IEEE , 21:1,59-66.

24. Syeed, M.M., Hammouda, I. (2013) Socio-Technical congruence in OSS Projects: Exploring Conway's law in FreeBSD OSS evolution, Proceedings of 9th International Conference of Open Source Systems (OSS), Springer.

25. Syeed, M.M., Hammouda I. (2013) Exploring Socio-Technical Dependencies in Open Source Software Projects - Towards an Automated Data-driven Approach, Academic Mindtrek Conference, ACM.