Fall September 13, 2012

Proceedings of the OSS 2012 Doctoral Consortium

Klaas-Jan Stol
Charles M Schweik, *University of Massachusetts - Amherst*
Imed Hammouda

Tampereen teknillinen yliopisto. Ohjelmistotekniikan laitos. Raportti 23
Tampere University of Technology. Department of Software Systems. Report 23

Klaas-Jan Stol, Charles M. Schweik & Imed Hammouda (eds.)

Proceedings of the OSS 2012 Doctoral Consortium, September 13, 2012, Hammamet, Tunisia

TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

Tampereen teknillinen yliopisto. Ohjelmistotekniikan laitos. Raportti 23
Tampere University of Technology. Department of Software Systems. Report 23

Klaas-Jan Stol, Charles M. Schweik & Imed Hammouda (eds.)

Proceedings of the OSS 2012 Doctoral Consortium, September 13, 2012, Hammamet, Tunisia

Tampere University of Technology. Department of Software Systems
Tampere 2012

ISBN 978-952-15-2993-1
ISSN 1797-836X

Preface

Open Source Software (OSS) is an active research area and the OSS conference as its premier publication venue has reached its eighth edition this year. While OSS has been studied for over a decade, and a wide variety of aspects of OSS have been studied, the research area is still relatively young. Like any emerging field of study, s the research community learns more about the fundamental research questions, the field itself evolves and new questions arise, which present a fruitful ground for novice researchers.

To facilitate these new researchers with an arena to present and receive feedback on their research, the OSS conference has organized a Doctoral Consortium for several years. The principle objective of the consortium is to provide doctoral students the opportunity to present their research at various stages of production – from early drafts of their research design to near completion of their dissertation – in a forum where they can get critical and helpful feedback from a community of interested scholars and other students as they work to finish their degree.

This volume contains the 10 papers selected by the consortium organizing committee for presentation at the Doctoral Consortium, which was held at the eighth International Conference on Open Source Systems, at Hammamet, Tunisia in September 2012. The papers provide a look into the research-in-progress by Ph.D. students who study a variety of aspects of OSS. The students represent countries all over the world, from Japan to India, and from the U.S.A. to Tunisia, which indicates that there is a global research interest in this topic, and that OSS plays an important role worldwide.

We believe this year's doctoral consortium has been very successful, due to the kind cooperation and support of the audience and, of course, the presenting students. In particular, we would like to express our sincere gratitude to Björn Lundell, Gregory Madey, and John Noll, who have acted as a de-facto panel during the consortium. We also are greatly appreciative to the U.S. National Science Foundation, who provided travel support (award # 1132151) for two U.S. Doctoral students and one U.S. Faculty participant, along with funding for the consortium's facility rental. Of course, any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the U.S. National Science Foundation.

Limerick, Ireland
November 2012

Klaas-Jan Stol
Charles M. Schweik
Imed Hammouda

Doctoral Consortium Organizing Committee Chairs

Cornelia Boldyreff	University of East London, UK
U. Yeliz Eseryel	University of Groningen, the Netherlands
Charles D. Knutson	Brigham Young University, USA
Charles M. Schweik	University of Massachusetts, USA
Klaas-Jan Stol	Lero—the Irish Software Engineering Research Centre, University of Limerick, Ireland

Contents

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

M.M. Mahbubul Syeed

Tampere University of Technology, Finland
mm.syeed@tut.fi

Abstract. Open Source Software (OSS) projects are currently becoming a viable medium for developing and distributing quality software. Yet users of OSS projects, specially from commercial domain want to assess and ensure the effective management of evolution in OSS projects. A plethora of empirical studies concerning either the evolution of the OSS or the community was conducted in last decade. But a little attention was paid to explore the co-evolution of the code and community of OSS projects. In OSS era the code and the community are mutually inclusive as the contributions of the community members drive the system evolution, which in turn redefine the role of contributing community members and change the social dynamics within the community. Thus it has profound impact on the sustainability of OSS projects.
Due to the interdependent nature of evolution of the code and the community of OSS projects, this PhD research would primarily based on the following proposition, *The evolution of the Open Source Software (OSS) project is constrained by the non-orthogonal evolution of the software and the community surrounding it*.

Keywords: Open Source, Evolution, Co-evolution, open source community

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. Both these dimensions are potentially inclusive.

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.

This paper describes the current plan for the research intended for the authors doctoral thesis. The organization of the paper is as follows. In Section 2 the motivation of the work is stated. Research objective and corresponding methodology is presented in section 3 and 4, respectively. Section 5 gives an overview of the current progress on the topic and finally, concluding remarks are presented in section 7.

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, Libre-

Office, 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 co-evolution 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][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 work 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 long term sustainability?
- 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.

4.1 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. Thus, 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.

4.2 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.

As described in section 3, the target of this work is to investigate the evolution of the code and community dimensions in OSS projects in order to identify interdependency and influence on each other, and ascertain their implication on the quality parameters of OSS projects. Thus this case study research can be categorized safely as a combination of exploratory and explanatory/ confirmatory case study research

[20]. For data collection and analysis we used quantitative data analysis as discussed in section 4.3.

4.3 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].

In what follows, the description of the study protocol, and a discussion on the data collection method and analysis technique.

Case study Protocol The study protocol is presented in Figure 1.

Fig. 1. Case Study Protocol

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 explored mostly the historical data 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), or from sources which collect and maintain OSS project data for research (e.g., sourceforge, ohloh) [10]. For this research, the former approach is followed, i.e., the data is collected directly from project sources. Because it gives most flexibility as the data that is most suitable for the research questions under investigation are collected and analyzed [15]. Nevertheless, this form data collection (termed as third degree data collection in [21]) is criticized due to the validity and completeness of the data for 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 several different data sources. For example, developer's contribution in a project is derived from the data collected from source code repository, change logs, 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 are controlled and unnecessary data collection are restricted [15].

5 Current Progress

The thesis work is now close to one and 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 the research and the supported publications mainly lay a foundation for the thesis work and provide a road map to complete the whole picture with further research. In what follows, the description on the current progress and the publications made for this research.

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. One article is published based on the first review [8] and the article based on second review is under correction phase [10].

Case study 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 (shown in Table 1), and

were investigated as follows, (a) a set of methods were defined for exploring the relations of the code base and community structures of OSS projects in order to answer the research questions, (b) a meta-model was proposed to model both the code and community dimension of the OSS projects, and support the implementation of the methods and a graph based visualization [6], (c) a prototype tool, Binoculars was build which provide the implementation of the methods and a graph based visualization, and (e) the applicability of the proposed methods and the tool was assessed with two large scale OSS projects, namely FFMpeg and Eucalyptus [6]. In this study a repository was build based on the proposed meta-model. This repository contains the data extracted from the OSS repositories (e.g., version control systems, mailing list, bug reporting systems, user information systems).

Table 1. Research Questions

	Research Questions
Q1.	Where does the expertise lay within the developer community?
Q2.	Whom (developer) users should contact to solve an issue?
Q3.	How many people work on each software component? Who are they?
Q4.	Does the inclusion structure of the code base conform to the organizational structure?
Q5.	What are the critical issues related to the performance of the software?

Tool implementation. In order to analyze and comprehend evolutionary patterns of OSS projects, a tool Binoculars is implemented. This tool relies on graph based data representation and visualization [9], and was build on top of an extensible meta model [6] that captures and models the essence of OSS projects for such analysis. Figure 2 presents the main interface of Binoculars.

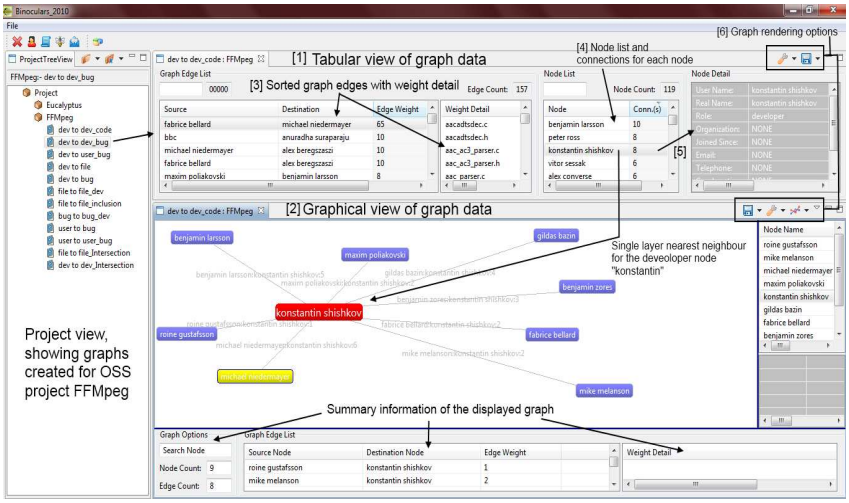

Fig. 2. User interface of Binoc-ulars

The meta-model represents both the code and the community dimensions through mining the repositories produced by OSS projects and derive their inter-relationships using mathematical graphs [6]. The principal argument in favor of this graph based representation and visualization [9] is that graph structures are most suitable for analyzing data that exhibits inherent relationships. In this context, the repository data produced by OSS projects exhibit strong relationships among them due to common work space sharing and exchange of information. For example, community members often share many technical competencies, values, and beliefs over online discussion forums. Similarly, code artifacts have interrelationships due to architectural dependency as well as due to contributions from multiple community members. Thus, OSS projects can be effectively comprehended through graph based representation and visualization. A validation of this claim is done with two example OSS projects, namely FFMpeg and Eucalyptus and the results are reported in [6].

Visualization mechanism of the tool Binoculars offers the followings: provides a two-way visualization of a graph, e.g., tabular and graphical (Fig.1 items 1,2). Tabular view provides complete graph information consisting of (a) Graph with nodes and (weighted) edges; (b) Node list with degree count for each node; (c) description of each node; (d) Summary data on graph; and (e) Options to render a graph (Fig.1, item 3,4,5,6, respectively). Thus user can get complete graph data with detail information in real time for large graphs with thousands of nodes. Then, depending on the option selected for rendering a graph, a modified (or abstracted) version of the graph (in tabular view) can be viewed in graphical form. As shown in Fig.1 item 2, a single level nearest neighbor graph showing the developers to whom developer Konstantin has direct communication in FFMpeg project.

Other options for rendering a graph includes (Fig.1, item 6), customization based on (a) given range of edge weights, (b) selected set of nodes or edges from the original graph, (c) a given attribute value (e.g., gio-location= america). None-the-less, searching, sorting, zooming, and saving graph data in XML format can also be performed.

A video demonstration of the current implementation of the tool can be found in [11].

6 Future Direction

Current and future direction of this thesis work can be summarized as follows,

First the evolution data will be extracted from the OSS project repositories, e.g., SVN/CVS code repositories, change logs, mailing list, bug reporting systems, and user information system. This data will be fed into the Binoculars repository, which was build on top of the meta-model. These repository data will then be used to run the case studies.

Second define and implement methods as plug-ins to the Binoculars tool to evaluate the hypothesis. These methods will utilize the repository data for quantitative analysis and will explore their inter-relationships to identify the co-evolutionary pattern if exists.

Third the tool will be released as an open source project. This would benefit the OSS community in assessing their projects with the tool, and to tailor it according to their need.

Fourth the evolution data extracted from the study projects will be released publically for further research. The parsers build for this data extraction will also be made publically available.

7 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, 2012.
9. Syeed M.M. (2012) Binoculars: Comprehending Open Source Projects through graphs, In Proceedings of 8th. IFIP International Conference of Open Source Systems, Springer, 2012.

10. Syeed M.M., Hammouda I, Systa T (2012) Evolution of Open Source Software: a systematic literature review, Manuscript under writing process.
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.

QA practices and FLOSS communities

Adina Barham
Hitotsubashi University, Graduate School of Social Sciences, 2-1 Naka,
Kunitachi, Tokyo, 186-8601, Japan
adina.barham@yahoo.com

Abstract. The number of FLOSS projects that include a quality assurance step in the development model is increasing which suggests that a new layer may be emerging in the classical "onion model". The existence of a team of people performing quality assurance tasks is impacting the communities' structure and might affect the information flow within the project and implicitly its sustainability. Due to the fact that the community, an essential resource for any FLOSS project, can determine the success or failure of the project, it is important to study how these changes affect the overall internal dynamics. Although FLOSS communities have been extensively studied, questions that concern quality assurance teams, such as how this group of contributors interacts and fits with the rest of the community, are yet to be answered. This research aims to take a step towards answering these questions by analyzing the dynamics of various quality assurance teams through the perspective of social network analysis. In order to have an accurate picture of how quality assurance interacts with the community mailing list data, issue tracker data as well as other available communication data is taken into consideration. The conclusions reached so far, although provisory, suggest that quality assurance activity may not be increasing steadily over time but dependent on other factors. In addition, data collected so far points to a highly connected network that contains a large group of members communicating over various channels.

Keywords: quality assurance, test, social network analysis, information flow

1 Introduction

In recent years open source software has become a viable choice for a wider range of users, overcoming its initial status as a tool used only by experts and hackers. This phenomenon has led to higher expectations from end-users which translates into a greater need for responsible management, productivity over time, ease of maintenance, availability of support, increased quality and other features that now drive the success of FLOSS projects. This paper investigates whether and to what extent this change is affecting the way FLOSS communities develop software.

It is not a surprise anymore when an open source project's community decides to adopt methodologies and policies that point more towards a hybrid development

model than towards the bazaar model. This hybrid model seems to adopt development methodologies from traditional FLOSS development such as heavy community involvement, as well as proprietary development such as the quality assurance (QA) phase. QA is a rigorous process that implies a series of elaborate steps necessary to ensure a certain quality standard. As a part of the result, formal quality assurance practices are becoming more and more present in open source software projects which might suggest that Linus' law [10] is not applied as easily and as often considering today's strict market requirements. Furthermore, it is safe to assume that as a project matures so does the testing process around it, a truism for both open source and proprietary software [11].

Even though QA practices are becoming more and more present in FLOSS projects, their success or failure depends greatly on actual community development [7], in other words on the project members. Furthermore, characteristics of the community such as its size [15] are important factors influencing the quality of a software product. We therefore need an up-to-date understanding of communities' structures and dynamics.

2 Background and Motivation

Open source software development has evolved substantially to keep with the standards imposed by the continuously growing user base and the needs of the market. This implied refining the development process and pushing it towards a more sustainable model. But what does the term sustainable actually mean in this context? The Brundtland Commission's report defines sustainable development as development that “meets the needs of the present without compromising the ability of future generations to meet their own needs” [21]. In the context of open source software, this includes raising the quality standards of products by implementing more complex processes and rigorous methodologies.

The importance of quality in open source is recognized as an important issue that needs to be further studied, a fact illustrated by current academic research that covers a wide range of QA related issues. This research has addressed such topics as the ways in which QA practices used in traditional software development are adapted in FLOSS projects [1], the kind of QA practices implemented in FLOSS development [3], the challenges faced in maintaining acceptable QA levels and sustainability [4], the factors that affect sustainability [26], and the tools and techniques that can be used to measure quality assurance [13]. In addition, researchers have analyzed existing literature to suggest practices, methodologies and lessons learned that can be used to improve quality [2], [20]. QA is also a significant trend in research programs funded by various organizations and governments such as the Qualipso project (Trust and Quality in Open Source Systems – largest open source initiative funded by the the European Commission as part of its Information Society initiative) [25] or Qualoss [23]. However, little research has been done on how the community, the most important resource of FLOSS projects, is being affected by implementing QA practices and how the communities’ internal dynamics are changing.

Another important trend in current research is the use of social network analysis to examine the communities that drive FLOSS. For example, a study by Masmoudi et

al. [18] of the communication patterns between core and periphery in the case of the Firefox project concludes that core members seem to engage with the periphery more frequently than one would expect. Another study that partially confirms this behavior is the one conducted by Oezbek et al. [17]; by performing a social network analysis on a number of open source projects they reach the conclusion that core members are more active in communicating with the periphery via e-mail than expected and that "Core member status may be qualitatively (rather than just quantitatively) different and the transition of individual mailing list participants towards ever higher participation is qualitatively discontinuous." Furthermore, although social network analyses of FLOSS communities have focused on a number of aspects ranging from structure and dynamics [12], [14] to migration within the hierarchy of FLOSS projects [9], none of these studies have sought to link QA with the rest of the community. This research aims to fill that gap.

In this research it has been assumed that formal quality assurance is present within a FLOSS project if members of the project's community form a layer that is responsible for the QA process and is easily identifiable (meaning that information associated with the QA team such as web pages, wikis, mailing lists, forums and so on can be easily found). In a previous preliminary study, in order to check the extent to which QA is present among popular FLOSS software, I accessed the first 50 applications from http://www.ohloh.net [24] that were ranked as most popular by number of downloads. The conclusion was that approximately one third have easily identifiable QA. Further analysis is necessary in this regard, in order to get a more general idea about current trends and due to the fact that QA might not be as easy to identify as previously thought. For example, it is possible that a project has a clearly defined QA step but it might be defined in a different way, has a different name or is performed by other parties.

By analyzing the QA teams within FLOSS organizations we can take a first step towards clarifying the QA position within the open source community and develop these findings into a framework that can be applied further in FLOSS projects. Due to the particularities of each project, producing a foolproof recipe for success is impossible but a study of this kind should provide important lessons.

3 Research Questions

Q1: How does a QA contributor fit into the community?

Although recent research has defined more than three layers in the onion model [6], [9] it is generally accepted that a project's community can be split into: active users, co-developers and core developers. This research aims to investigate the extent to which QA is a step on the road from end-user to developer, or whether QA has become established as a separate category of contributor. Previous research has shown that participants ascend and descend the organizational hierarchy as well as move laterally to other tracks but it has also been suggested that many QA tasks are assimilated by other roles [9]. It would be also interesting to note, if all projects taken into consideration share the same structure with respect to the QA contributors.

Q2: What are the QA activities performed in FLOSS projects and what are their characteristics?

QA activities can vary based on technical difficulty, for example users may provide automated test tools which might mean that QA might be divided into two subgroups based on activity type. Another aspect that should be investigated is how participants' activities evolve overtime in the sense that users may begin with 'easier' activities such as commenting on the issue tracker or sending e-mails considering that most FLOSS communities are meritocracies. In later phases, users can become more active by posting bugs, editing their status or even providing fixes. On the other hand, members of the periphery also perform some QA tasks such as posting bugs on the issue tracker; it has been noted that for the case of Firefox the percentage is 20 to 25% [18] and it would be interesting to compare and see the percentage of periphery involvement for other projects. Contributors generally use mailing lists, IRC channels, issue trackers and other channels for project related communication and it is logical to assume that some members will be more active than others but it would be worthy to investigate if participants active in one environment are active in others as well or there is some kind of distribution. Studying this aspect will show the extent to which there is a core of people within the QA group who are performing most tasks.

Q3: What are the communication patterns between QA members as well as with other project participants?

The goal of analyzing the communication patterns between QA members is to find the central figures within the community and observe their evolution over time within the project. Another issue would be comparing central figures based on different data sets, for example central figures based on e-mailing activity and central figures based on issue tracker activity. Social networks are not static and the structure is continuously changing over time [8], which means that participant position within the network is dynamic. For this reason it would be useful to learn the path, or paths, which the central members took to get into that position. As previous research has shown, information access correlates with productivity and participants who have better access to information are able to contribute more efficiently [19]. Hence information flow and who controls it is very important for the success of a project, and for this reason QA interaction with other layers of the community should not be ignored. Where bridges exist between the QA team and other teams, the information flow will be compromised temporarily if these bridges are broken for one reason or another.

4 Data and Research Method

In order to study communication and activity patterns quantitative research methods are employed and mainly social network analysis techniques because they allow easy discovery of communication patterns, community evolution over time as well as identifying user group affiliation and migration paths.

Projects' participants will be represented as nodes (vertices) while interactions will be represented as edges (arcs). Defining interactions between members could pose some difficulties in the sense that data will be collected in different formats and in some cases thread-based analysis could be difficult to implement. An alternative solution would be quotation-based analysis in which participants quote e-mails to which they are replying [16]. In other words, in quotation-based analysis a relation exists once one member quotes another member's message in his/her e-mail. In thread-based analysis a relation exists once a member replies to the sender of the previous message by including the contents in his/her e-mail. In the case of thread-based analysis, relations are created between members who are exchanging messages on a certain subject or topic. Previous research successfully used both quotation-based analysis as well as thread-based analysis, for example, Crowston et al. [6], [12] defined a link between two developers as a reply (or follow up) to the previous message posted on the projects' bug trackers. The number of e-mails sent from one member to the other will be represented as weight (value) of the graphs' arcs.

When examining central figures within the community it is important to pay attention to the dynamic attributes of the network structure and analyze the network evolution using time frames in order to avoid biased results [5]. Take for example, the centrality value: this is used to determine the importance of a node within a network and can be calculated, among other ways, by counting the node's connections to other nodes. The centrality value can be easily biased by using an aggregated network[1] due to the fact that if a leader (a node with a large number of connected edges) leaves a project and is replaced with another leader then the centrality level is maintained for the central member but in case we collapse the two states of the graph we will obtain a lower overall centrality value. Another aspect that needs attention is how long a relationship lasts between two members based on exchanged e-mails or comments. However it is probably safe to assume that after exchanging a certain number of e-mails/comments, a relationship is maintained or can be easily restarted at a later date.

Examining only one communication channel does not help paint an objective picture of a communities' structure and for this reason, in this phase of the research QA mailing lists and issue tracker commenting activities will be analyzed. Future research phases will include other communication environments such as forums and code commits with the associated comments.

5 Mozilla

Due to the fact that it is a mature and successful project offering a considerable amount of data, Mozilla was chosen as a first project in order to help draw preliminary conclusions and create a framework for further research. Mailing list data was downloaded in July 2011. At that time, according to the Mozilla Quality Assurance (QMO) website [22], there were 5 sub teams, each with its own web forum: Web QA, Desktop Firefox, Browser Technologies, Automation, Services[2].

[1] An aggregation of all the states that the network has been through to reach the current state

[2] The structure of the Mozilla Quality Assurance teams is dynamic considering the fact that the Services team was ulteriorly dropped and that since data was collected, other changes have also taken place.

The activity on the web forums was low compared to the mailing list activity and for that reason the mailing lists[3] were analyzed instead. Web QA, Desktop Firefox, Browser Technologies, Services teams used the mozilla.dev-quality mailing list while the Automation team used the Mozmill developer mailing list.

Bugzilla data was acquired in February – March 2012 using a web crawler that accessed pages associated with each bug (the actual bug page and the change history page associated with each bug) and stored the issues directly into a PostgreSQL database. The next phase consisted of data cleaning both data sets by including activities such as removing double posts, removing spam or normalising name formats. As expected, the QA mailing list was created much later than the issue tracker. For the purpose of analysing activity levels data should be normalised but at this stage of the research it is still unclear how to proceed with data that will be analysed using social network analysis techniques. If activity pertaining to a certain period of time will not be taken into account then it will be possible to omit migration of certain members between layers of the community.

4.1 Mailing List Data

The Mozilla.dev-quality mailing list data contains 2535 e-mails exchanged between February 2006 and June 2011 while the Mozmill developer mailing list data contains 1155 e-mails exchanged between October 2008 and July 2011. The traffic and number of users is higher on the Mozilla.dev-quality mailing list than the Mozmill developer mailing list (see Table 1)[4]. This is to be expected as the Mozmill developer list is addressed to more technically aware users since it is dedicated to an automated testing tool produced by the Mozilla community primarily to test their own products.

Table 1. Mozilla.dev-quality: 2006/17/2-2011/6/30, Mozmill developer 2008/10/1-2011/7/21

	Mozilla.dev-quality	**Mozmill developer**	**Total**
Topic	1042	313	1299
Messages	2535	1155	3689
Thread initiators	199	47	233
Distinct author	293	61	332

4.2 Bugzilla Data

The issue tracker (Bugzilla) data set covers all Mozilla products from 1998 to 2012 containing 687,221 bugs with 5,834,507 associated comments. Bug ids range from 0 to 724,339; the collected bugs thus represent 94.87% of the id range. The remaining 5.13% were not collected because they were not publicly available or due to bad html that could not be parsed.

[3] Two mailing lists are at the disposal of the QMO teams: mozilla.dev-quality which contains more general discussion topics and Mozmill developer which contains more technical discussions topics that are mostly about the Mozmill testing tool.

[4] The difference in the total is due to cross posting and users belonging to both lists

Approximately 4,400 distinct project members were identified as assigned to fix bugs. Without getting the data associated with code commits it is not safe to assume that these members were also the members that posted the bug fix, but it is safe to assume that they are code committers. These users are also active when it comes to posting bug comments as well as sending e-mails on the QA mailing lists. After cross-referencing members active on the mailing lists and code committers, 883 bugs were found, most of which pertain to Firefox.

4.3 Data analysis

From all the e-mails exchanged 152 (4.12%) were sent by authors who had sent only one e-mail throughout the period taken into consideration for this research. On the other hand 135,466 bugs (19.70%) were posted by members who had posted only one bug throughout the period taken into consideration. Most activity levels have steadily increased over time; however activity levels on the QA mailing lists do not seem to be linked to time progression. As a result it would be useful to find out if the activity peak in 2009 is linked to interior events or external factors. Developer activity includes bugs to which they are assigned, comments posted on the issue tracker as well as e-mails send on the QA mailing lists.

Table 2. Activity levels on a yearly basis

	2006	**2007**	**2008**	**2009**	**2010**	**2011**
Comments	328846	335323	467087	528199	658030	703857
Bugs	42015	41995	56785	60880	78089	78896
E-mails	343	361	556	1307	739	384
Dev bugs	119571	123234	174742	177776	227123	226555
Dev Comments	258458	271679	375729	449539	541707	561853
Dev e-mails	196	286	343	953	500	264

If we consider that 11 e-mails (average number of e-mails sent) is the lower limit for highly active users then Pareto's principle is somewhat applicable in the sense that only 16.8% of the users sent more than 11 e-mails and 17.69% of users receive more than 11 replies. Following the same principle, only 4.39% of users show a higher than average activity posting more than 39 comments and 9.25% more than 6 bugs.

In this phase of the research, due to the fact that data collection and cleaning took longer than anticipated, social network analysis techniques could not be applied to the whole data set. Instead interaction was analyzed between active members on the mailing list (more than 10 e-mails sent – 55 users) and 10 members fairly active on the issue tracker. The resulting network does not depict relations between all QA members and its role is only to offer a sample of the interaction patterns within the community. After eliminating loops (replies to themselves) this sub-network had a number of 1433 participants with 2593 connections; 933 of these connections were

formed by more than one interaction. The average degree is 3.16, which means that the average number of connections a member has is approximately 3.

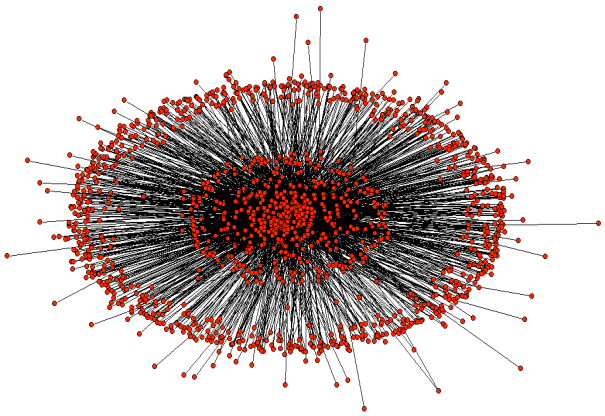

Fig.1. Social Network - Mailing list and 10 active members on the issue tracker

6 Ubuntu

Ubuntu was chosen as the next case study for this research as it is yet another mature and successful FLOSS project. As opposed to the Mozilla projects, Ubuntu has a dedicated QA team [28] as well as a Bug Squad [27]. The Bug Squad's tasks include assigning bugs to packages, ensuring that bug reports are adequate, finding duplicate bugs, replicating bugs and so on.

As a first step, the mailing list archive associated with the QA team was downloaded using a web crawler and stored in a local PostgreSQL database. The dataset contains e-mails from October 2007 to August 2012. Using a similar method, the mailing list archive associated with the Bug Squad team, which contains e-mails spanning from July 2006 to August 2012, was downloaded. In total 5786 e-mails were exchanged between 1153 members.

At this point of the research mailing list data associated with Ubuntu QA and Bug Squad was not cleaned. Thus the results presented in this paper are only provisionary.

Table 3. Activity levels on a yearly basis

	2006	2007	2008	2009	2010	2011	2012
QA mailing list	-	30	324	321	693	486	588
Bug Squad mailing list	206	515	536	475	1017	435	160

If we consider that 5 e-mails (the average number of e-mails sent per user) is the lower limit for highly active users, then we conclude for Ubuntu as well that Pareto's principle is somewhat applicable in the sense that only 16.86% of the users sent more than 5 e-mails and 11.79% of users received more than 5 replies.

In this phase of the research, due to the fact that data was not cleaned thoroughly, final conclusions regarding communication patterns between Ubuntu QA and Bug Squad members cannot be drawn. However a preliminary analysis was conducted using Pajek by integrating both mailing lists. After eliminating loops (replies to themselves) this sub-network had a number of 931 participants with 2196 connections; 383 of these connections were formed by more than one interaction. The average degree is 4.7, which means that the average number of connections a member has is approximately 5.

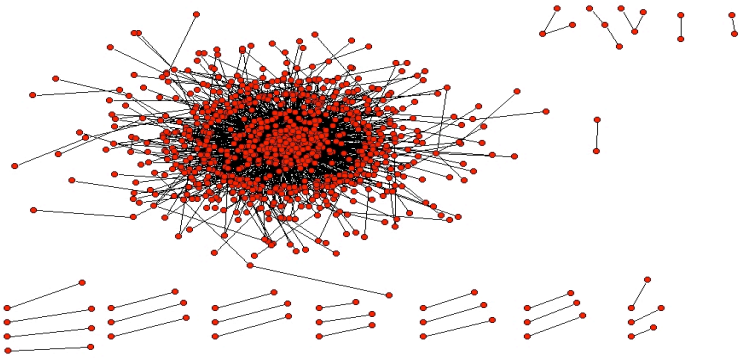

Fig.2. Social Network – **QA** mailing list and Bug squad mailing list

The network formed contains 29 components from which 28 contain less than 3 members. In other words, there is a large connected group of 872 members.

7 Conclusions

Q1: How does a QA contributor fit into the community? Considering the fact that the Mozilla and Ubuntu QA teams have dedicated communication channels, one can draw the conclusion that these teams represent a layer in the community model. However, the question whether QA forms a separate layer in the community can't be answered due to the fact that non-QA mailing lists were not yet analyzed.

Q2: What are the QA activities performed in FLOSS projects and what are their characteristics? At this point of the research a clear definition of the tasks performed by QA members has not been made, evidence such as the existence of a QA mailing list oriented to more technically aware users, in the case of Mozilla, might suggest that there is more than one type of QA task. In addition, tasks performed by Ubuntu's Bug Squad may also be considered as QA activities thus pointing out that a detailed classification of QA activities is necessary.

As expected the activity of members of the community that "hit and run" (open one bug and never contribute again, send one e-mail and never contribute again) is

higher on the Issue Tracker than on the QA mailing list for Mozilla projects. This may suggest that QA mailing list members have a more sustained activity in the Mozilla community. Another difference is that issue tracker activity has shown an increase over time while mailing list data showed a peak level. The same kind of peak in mailing list activity can also be noticed for Ubuntu data. This might suggest that mailing list activity may not be related to time progression but to other variables that need to be identified. On the other hand, for Mozilla the increase in activity on the issue tracker points out the community has grown over the years and these results are to be confirmed during later stages of the research with Ubuntu data.

Q3: What are the communication patterns between QA members as well as with other project participants? Mozilla associated data used for the social network analysis section of this study was performed only on a sample due to time related issues and thus a general conclusion regarding communication patterns cannot be drawn at this point. However, the sample shows no small groups of people working together but a team spanning both mailing lists and issue tracker. In addition, judging from the activity of QA members and code committers on the issue tracker it is safe to say that interaction with other community members has been increasing. This suggests that it is unlikely that there will be participants that control the flow of information, or bridges between the QA team and other layers of the community. Ubuntu associated data used for the social network analysis section of this study was not cleaned and like for the Mozilla data, a general conclusion cannot be drawn. However, one large component spanning both mailing lists can be easily observed.

8 Limitations and Further research

The purpose of this research is not to derive general conclusions that can be applied to all FLOSS projects. It is logical to conclude that by analyzing the structure and behavior of some FLOSS projects, one can't obtain a foolproof method to successfully implement QA practices due to the variety and uniqueness of every FLOSS project. However, the final goal is to create a precedent for further study in this direction in order to come up with a framework that can be applied on a wider scale.

One of the limitations of this study is that the community members might also use other communication channels that are not publicly available. This is one of the reasons why quantitative research should be confirmed with a qualitative follow-up. Results can be confirmed in later stages of the research by designing a questionnaire and asking the community, or at least some of the influential figures of the community. Another reason to go back to the community is correlating data peaks or anomalies with actual situations. These peaks or anomalies can be correlated with both internal events such as a release with an increased risk or an external event such as a new competitor. In addition, when collecting Bugzilla data for Mozilla projects, some of the bugs were not publicly available and it is relevant for this study to know why and if these bugs were deleted as being doubles and so on.

Furthermore, to complete the analysis for the Mozilla projects, in the next phase of research, social network analysis will be applied using time frames and with consideration to time decay affecting connections between members of the community. The results presented in this paper are only provisory due to the fact that

data collecting took a longer time than initially expected and processing an amount of data of this size also takes a considerable amount of time (For example, exporting all the data in Pajek format including time frames). Furthermore, in order to obtain an objective categorization of community members it is necessary to integrate previously acquired results with code comment data and non-QA mailing lists. It is essential to separate the QA members from developers and track their evolution within the community by monitoring their activity levels in different time frames in different environments.

In addition, to obtain useful data for analyzing the Ubuntu community, a thorough data cleaning is necessary. The next phase of analysis should include downloading issue tracker data, non-QA mailing lists, code comment data and integrate it with the QA and Bug Squad mailing list data. For the same reasons listed for the Mozilla projects, results should be confirmed with questionnaires addressed to the community or at least influential members of the community. After cleaning, processing and analyzing Mozilla and Ubuntu integrated data with Pajek, the role of QA within these communities should be clearer by obtaining information regarding community structure with respect to QA, career advancement and migration between teams.

It is also important to define QA activities that are employed in FLOSS projects and for this reason, projects listed as on www.ohloh.net should be analyzed with respect to QA activities and a clear classification should be made. After this step, it should be clear how each community decided to adapt QA practices to the FLOSS philosophy.

The final step of this research will be the integration of all findings into a framework that can be applied on a wider scale. The viability of this framework should be confirmed by addressing a survey to members of other successful FLOSS projects. Another interesting aspect worth investigating is when did these communities decide to include a dedicated QA and if it is an increasing trend. Whether the quality of FLOSS products have improved or not after the introduction of a formal QA step could represent a valuable assessment for other growing FLOSS communities. And, for this reason further studies should include quality evaluation and measurement of FLOSS products.

References

1. Halloran, T. J., & Scherlis, W. L.: High quality and open source software practices. In: 2nd Workshop on Open Source Software Engineering (2002)
2. Hedberg, H., Iivari, N., Rajanen, M., & Harjumaa, L.: Assuring Quality and Usability in Open Source Software Development. In: First International Workshop on Emerging Trends in FLOSS Research and Development, FLOSS 2007, pp. 2 (2007)
3. Michlmayr, M., Hunt, F., & Probert, D.: Quality practices and problems in free software projects. In: Proceedings of the First International Conference on Open Source Systems, pp. 24–28, (2005)
4. Schmidt, D. C., & Porter, A.: Leveraging open-source communities to improve the quality & performance of open-source software. In: Proceedings of the 1st Workshop on Open Source Software Engineering (2001)

5. Howison, J., Wiggins, A., & Crowston, K.: Validity Issues in the Use of Social Network Analysis for the Study of Online Communities. In: Journal of the Association for Information Systems (2012)
6. Wiggins, A., Howison, J. & Crowston, K.: Social dynamics of FLOSS team communication across channels. In: Proceedings of the Fourth International Conference on Open Source Systems (2008)
7. Kilamo, T.: Essential Properties of Open Development Communities. In: Proceedings of the OSS 2011 Doctoral Consortium (2011)
8. Watts, D. J., A.: Twenty-first century science. In: Nature, Vol. 445, No. 7127, pp. 489-489 (2007)
9. Jensen, C., & Scachi, W.: Role Migration and Advancement Processes in OSSD Projects: A Comparative Case Study. In: Proceedings of the 29th international conference on Software Engineering, pp.364-374 (2007).
10. Raymond, E.S.: The cathedral and the bazaar. O'Reilly (1999)
11. DiBona C., Cooper D., Cooper M.: Open Sources 2.0: The Continuing Evolution. O'Reilly, USA (2006)
12. Crowston K., Howison, J.: The social structure of Free and Open Source software. In: First Monday, Vol. 10, No. 2 (2004)
13. Spinellis D., Gousios G., Karakoidas V., Louridas P., Adams P. J, Samoladas I., Stamelos I.: Evaluating the Quality of Open Source Software. In: Electronic Notes in Theoretical Computer Science, Volume 233, Proceedings of the International Workshop on Software Quality and Maintainability (2009)
14. Mockus A., Fielding R. T, and Herbsleb J. D.: Two Case Studies Of Open Source Software Development: Apache And Mozilla. In: ACM Transactions on Software Engineering and Methodology, volume 11, number 3, pp. 309–346 (2002)
15. Sowe S., Ghosh R., & Haaland K.: A Multi-Repository Approach to Study the Topology of Open Source Bugs Communities: Implications for Software and Code Quality. In: 3rd IEEE International Conference on Information management and engineering, IEEE ICIME (2011)
16. F., Burkhardt J-M, Sack W.: Thematic coherence and quotation practices in OSS design-oriented online discussions. In: Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work (2005)
17. Oezbek C., Prechelt, L. & Thiel F.: The Onion has Cancer: Some Social Network Analysis Visualizations of Open Source Project Communication. In: Psychology, Section 4, 4-9, (2010)
18. Masmoudi H., den Besten M. L., De Loupy C. and Dalle J. M.: 'Peeling the Onion': The Words and Actions that Distinguish Core from Periphery in Bug Reports and How Core and Periphery Interact Together. In: Fifth International Conference on Open Source Systems (2009)
19. Aral, S., Brynjolfsson, E. & Van Alstyne, M.: Productivity Effects of Information Diffusion in E-mail Networks. In: Proceedings of ICIS 2007 (2007)
20. Aberdour M.: Achieving Quality in Open Source Software. In: IEEE Software, pp. 58-64 (2007)
21. Burtland Comission: The Bruntland Report. United Nations, (1987)
22. https://quality.mozilla.org/
23. http: //www.qualoss.org/
24. https://www.ohloh.net/
25. http://www.qualipso.org/

26. Chengalur-Smith I., Sidorova A., Daniel S.: Sustainability of Free/Libre Open Source Projects: A Longitudinal Study. In: JAIS 11 (2001)
27. https://wiki.ubuntu.com/BugSquad
28. http://qa.ubuntu.com/

Automated mapping between Computer Aided Design program formats

Toby Borland
School of Computer Science, Brunel University
Toby.Borland@brunel.ac.uk
WWW home page: http://www.brunel.ac.uk/siscm/disc/

Abstract. Commercial software frequently uses incompatible data formats. In the case of Computer Aided Design or CAD programs used to specify manufacturing design data this causes significant economic loss transferring designs between manufacturers and suppliers. The diversification and scale of production enterprise is limited by the ability to transfer data between different production system formats. The objective of this research is to create an automated means to determine, preserve and translate the semantic content of CAD models between different CAD programs, effectively generating a common open standard. To date, determining rules for command translation has been a laborious manual task. Many features have no exact equivalent between programs, requiring a sequence of commands for one program to create the equivalent output of a single command within another. An automated test for command equivalence is possible if the geometric output produced by program commands is numerically tested for equivalency with the geometrical output of a different program.

1 Introduction

The weak link in Open Hardware is software; no FLOSS CAD software is at a production stage. This is due in part to an absence of a de facto standard or dominant format for parametric CAD systems. The approach presented in this paper is to extract common formats from proprietary software.

The interoperability of computer data limits any efficiency gained from computational automation. In the case of manufacturing software, the exchange between designers, engineers, suppliers and logistics is fraught with expense and delay resulting from incompatible software (Gerbino 2003, Horst et al., 2010). Computer Aided Engineering software vendors differentiate their products for development and marketing purposes, it is both difficult and uncompetitive to coordinate common data interface standards alongside product development (Ball et al., 2008).

The semantic structures within modern CAD programs are represented as parametric "features", commonly used to specify geometrical form within CAD programs (Shah & Mäntylä, 1995). Features represent a high level of geometrical abstraction corresponding to an descriptive engineering vocabulary that allow reasoning of shape abstraction, e.g. flange, tolerance, assembly (Brunetti & Grimm, 2004). CAD features have come to represent the associated behavior, constraints and intent of a physical feature. The feature concept encapsulates the semantic associations of a physical instance or process. In order to create a design, the relationship between geometric feature elements and their associated parameters specify the required geometry. It is the information within this "design intent" that is lost in the transfer of a model between one system to another. Unlike model surface geometry, "features" specifying semantic design intent are not readily transferred between programs (Kim et al., 2008).

Part of the issue of mapping feature semantics between different application is the absence of a one-to-one or bijective mapping, in many cases the action of a single feature in one application is only replicated by a sequence of several features in another. This complex mapping requires machine readable structures that can be readily extended. Current research has adopted ontologies to define and structure semantic mapping information (Ciocoiu et al., 2001).

Traditionally, ontologies are engineered to be useful within a specified conceptual domain. This approach relies on applications adhering to, and correctly implementing an existent ontology. In the case of Computer Aided Engineering applications, there is little incentive for developers to adopt prescribed ontologies although many exist. It is therefore necessary to determine semantic similarities between CAE programs to populate an ontology. This represents a significant expenditure of labour. Current research efforts use unsupervised lexical semantic matching tools for the purpose of automating this task. The drawback of this approach is the difficulty of verifying high probability matches using generalised statistical methods.

The research undertaken here uses the property of geometry-based programs to output models that can be numerically compared. Two identical geometries produced by two different sequences of feature operations in two different CAD programs infer that the different feature operations produce identical results. This property can be used to machine-check mapped features. A brute force combinational search of application features and their myriad parameters against those of a different application has an impractically large search space. Several statistical methods are used to partition this search space toward efficient dimensions.

2 Production data interoperability strategies

Enterprise efficiency is dependent on effective transmission of information. In the case of manufacturing, design and process complexity is limited by the attendant demands of organisational complexity (Sanchez & Mahony, 1996). The

manufacturing design and production process requires the interaction of diverse specialties. Product Data Management is the integration of information between such requirements as Computer Aided Design, Computer Assisted Manufacture, Product Lifecycle Management, Computer Aided Process Planning, Supply Chain Management and Enterprise Resource Planning. These multiple perspectives of product data requirement present difficulties in maintaining unambiguous semantic content in formats (Yang & Miao, 2007). The challenge of distributing a consistent semantic format across diverse systems is a requisite for formalising expert knowledge, preserving design rationale and automating design checking (Fowler et al., 2004).

Computer Aided Design or CAD programs will support the translation of surface geometry from other vendor formats but lose "design intent" associated with the imported geometry (Lei et al., 2007). The rapid pace of software evolution and the absence of a common format provides little incentive for program vendors to consider interoperability with software from competitors or legacy versions of their own software. The leading cause of inefficiency within design engineering is considered to be the incomplete transfer of data between different CAD formats (Kubotek, 2006; Tassey, 1999).

One solution is to rely on one vendor to supply and integrate all the required software. This is less effective if the supply chain does not use the same software; occasionally manufacturers demand their suppliers use the same software. This solution is considered too expensive and inflexible for the majority of manufacturers (Young, 2009; Ray & Jones, 2006).

Extensive standards coordination efforts are undertaken by the International Standards Organisation. ISO 10303, or "Standards for the Exchange of Product Model Data" is an ongoing initiative to standardise the format of product information. This process requires the consensus of representatives from across the manufacturing spectrum, constituting a significant coordination effort (STEP Handbook, 2006). The scope of this initiative has made it difficult to maintain consistency in specifications and to anticipate future requirements (Ball et al., 2008; Kosanke, 2005; Gielingh, 2008). Consequently, commercial software implements a reduced subset of these standards (Gerbino, 2003). Both these approaches require manufacturers to commit to either shared software or shared standards if they wish to share information across systems.

3 Development of semantic feature ontologies to share product data

Representation of information structure, semantics and relationships in a machine-interpretable ontology preserves a consistent interpretation of data. An ontology is a formal description of the types, properties and relationship of information within a specific domain. Where the ISO 10303 group of standards are non-logic-based ontologies suited for human interpretation, a logic based formal

ontology is suited to machine reasoning and discovery (Bittner et al., 2005). Artificial Intelligence research into automated means of organising, maintaining and providing access to large heterogeneous data repositories have created widely used formal ontologies that are well supported (Nieto & Auxillo, 2003). Feature semantic mapping is rarely bijective; ontologies are a sophisticated approach to structuring complex semantic relationships where simple bijective mapping of CAD features is limited, such as the case where a feature can be defined in a number of different ways.

The parameters that specify a feature are not necessarily unique. As an example, a circle can be specified as tangential to three lines or as passing through three points. A formal structure is required to provide logical consistency between both specifications of the circle example. It is this ability of formal ontologies to provide a machine-readable reasoning framework that allows automated data interpretation (Lee et al., 2005).

Manufacturing ontologies have been developed as a means of preserving, organising and reasoning information across heterogeneous design and process domains (Ciocoiu, 2001). It is acknowledged that existing applications rarely have explicit ontologies and that different application domains have native ontologies (Yahia, 2009). The command set for a mature CAD application can be seen to have an internally consistent "domain ontology". To transfer information between applications is to create a semantic mapping between domain ontologies. The mapping between domain ontologies is invariably characterised as mapping to an intermediate "shared" or "top" ontology. The drawback of a single ontology approach using a shared ontology is the difficulty in specifying a global shared vocabulary sufficiently general to represent diverse domain associations, yet economical enough to be computationally effective. (Ciocoiu et al., 2000; Wache et al., 2001).

An alternative and useful perspective is that of a single ontology performing as an "interlingua", the mapping between two application domains form the structure of a single ontology (Uschold & Gruniger, 1996). Lee describes cases of syntactic heterogeneitys between CAD feature domains that can be resolved via simple syntactic mapping. More complex instances of structural and semantic heterogeneitys require the definition of mapping axioms to process symmetric mapping. Seo defines a "layered" ontology based on the shared ontology of two CAD programs. A "top-down" approach is specified for new application areas; source ontologies that describe individual CAD systems inherit a subset of the shared ontology. In the case of providing the same construct for existing applications, a "bottom-up" creation of a shared ontology is derived from the bridging of ontologies local to the specific CAD applications (Seo et al., 2005). Bridging is manually defined by rules or axioms, resolution in the case of semantic inequivalence between applications is not defined.

Many of the ontology models propose specific ontology architectures, or prescriptive concepts which are then verified as a necessarily limited research experiment (Sudarsan et al., 2005). Multiple examples of interlingua ontologies have

been created based around CAD feature taxonomy, (Seo, 2005; Lee, 2005; Dartigues, 2007; Yang & Miao, 2007).

3.1 Current research into bottom-up interlingua ontologies for CAD format mapping

Practical interlingua ontologies do not currently exist between manufacturing applications. Because software evolves faster than standards are agreed, it appears unlikely that the introduction of a top-down ontology to connect heterogeneous application data will be adopted by commercial software applications. The task of generating or extracting a shared ontology from heterogeneous domain ontologies is significantly more difficult than defining a top ontology in advance of creating applications. The construction of a top ontology is subject to similar difficulties as a standardisation approach, it is difficult to anticipate the scope required for greatest utility. Generic top-down ontologies specified for CAD data have had limited success (Yahia et al., 2009). The issue of a useful granularity and generality specified within a top-down sharing ontology is described as "minimal commitment" by Gruber, a bottom-up interlingua ontology satisfies this constraint. It can be stated that the bottom-up approach generates an ontology that is necessarily relevant and computationally economical (Gruber, 1995).

By definition, a practical bottom-up interlingua ontology must be extensible. The research summarised demonstrates that an interlingua feature ontology can symmetrically map simple CAD models between commercial applications. The ontologies created have been limited in scope as the number of features between commercial CAD systems is extensive and dynamic. Techniques that can automate the discovery and identification of features for populating an interlingua ontology are a major step towards practical information exchange systems.

McKenzie-Veal creates a simple ontology from a CAD application in an automated fashion using a series of translations. A CAD file is exported to a neutral STEP file, the STEP file is translated to XML which is further processed in Protégé to reveal the classes and relationships of the CAD ontology. This procedure is then reversed, files are sent to the CAD file to observe discrepancies between the the original and the reversed file (McKenzie-Veal et al., 2010).

Existing ontology or schema extraction techniques have been used to identify semantic equivalence between CAD features. Yeo uses a weighted hybrid method identifying type similarity, syntactic similarity and concept similarity. Statistical regression methods (SVM, FEMFIT) are used to assign rank to matching findings (Yeo, 2009).

Hanayeh uses lexical semantic correlation of the API feature names with the output feature graph tree. The method relies on the tested application exporting the output feature graph in a neutral format (XML:RDF in this instance) (Hanayeh, 2008; Altidor, 2009)

The research outlined above employs automated methods to discover semantic similarities between features within different CAD programs. To date, the means

employed have used general methods derived from lexical semantic tools or have relied on existent translation. Methods of unsupervised lexical semantic discovery are an active research area, the statistical approaches employed are not at a commercial stage of development. The following methodology describes a fundamentally different approach to machine-checking that uses the geometric properties inherent within CAD applications.

4 Automated discovery of feature equivalence between CAD applications

A method is proposed to improve the success of automated interlingua ontology generation in the case of mapping CAD features between systems. This method is extensible in a more generalised form to the wider set of manufacturing applications.

If two CAD systems represent the same model, it can be stated that the minimum sequence of commands to create one CAD model are the mapping of the minimum sequence of commands to create the second CAD system model if the output geometry is identical. This assertion can be extended by stating that a command that alters a model in one CAD system maps to a command that creates the exact same alteration of a model in a second CAD system. Determining command equivalence can be inferred from matching corresponding alterations in respective CAD models. If a sample of points are taken from features on the surfaces of both models, a numerical comparison can detect identical models. This means of automated ontology extraction validation supporting feature mapping axioms upholds semantic mapping associated with features. Where CAD programs use features to embody design intent, a complex mapping between features of differing CAD systems constitutes semantic mapping.

A problem is immediately evident with this method. The proposed method is essentially a brute-force combinational search, the search space associated with hundreds of feature commands each having multiple parameters is very large. Techniques to reduce this search space are necessary.

4.1 Feature parameter normalisation

Feature parameters are chosen to minimise variability. A feature is accessed through an API as a function with a set of parameters. Discovery mechanisms exist on operating system platforms to allow third-party programs to pass parameters.

In the example of a feature function with four boolean parameters, there are 24 permutations possible. In the case of integers or floating point numbers there is a number of variations limited only by the machine precision and libraries used. Hence a strategy must be used to reduce the permutations associated with test input.

The immediate solution is to use a single number such as 0.0 or 1.0 as numerical parameters, normalising the input data. Certain features require a range of numbers,

for instance a vector defined between two points, or a sphere defined as co-tangent to three Cartesian points. It is then appropriate to use a sequence of numbers that will generate a model that has points that are simple to extract and normalise, for example;

```
unknown_feature(0.0, 0.0, 0.0, 1.0)   //first pass
unknown_feature(0.0, 0.0, 1.0, 0.0)   //second pass
unknown_feature(0.0, 0.0, 1.0, 1.0)   //third pass
:
unknown_feature(1.0, 1.0, 1.0, 1.0)   //2^4 pass
```

Once there is correlation identified between two parameters of two feature functions, a second pass to test for linear correlation identifies Cartesian coordinate inputs.

A similar technique can be used to identify numerical output. A returned set of points that correspond to coordinates taken from two CAD models being compared may have identical relative values but differing absolute values. This indicates translation or rotation. Data is normalised to account for variations in absolute position, orientation or scaling. This step is simplified with a judicious selection of input parameters as described earlier.

4.2 Identified feature recursion

The third method is to reuse features that have been successfully identified to create more complex models in a recursive search. Many commands are specific operations on features. If the atomic features of the CAD systems are matched first, their discovery directs the order of models generated. For example, a hole feature cannot be created by itself, a solid must first be generated for a hole to be removed from it. This order of feature discovery is central to the creation of an ordered ontology; one of the outcomes of this research is to identify an optimal method of ordering the search space to minimise the lower bound of search combinations.

4.3 Hybrid semantic matching

Hybrid semantic matching techniques represent the range of existing matching methods developed for ontology discovery. Linguistic matching, instance-based matching, structure-based matching, graph matching, typeface matching and document link similarity are effective techniques to analyse the CAD Application Programming Interface documentation (Bernstein, 2011). Lexical analysis can re-order the search sequence in descending probability. A union of the set of words within the function names of both applications under investigation can be probed for synonyms using the WordNet corpus, a minimum path length to a common

taxonomy defines the correlation between command, feature or function names of different CAD programs.

4.4 Numerical point assignment for equivalence testing

Numerical comparison requires determining an equivalent selection of points from both CAD systems under test. All commercial CAD systems will return Cartesian coordinate values from model boundaries. The points selected must efficiently characterise the nature of the model created. It would require an infinitude of points to create exact equivalence, so the problem becomes one of finding sufficient points to create a statistically significant match. To determine points through feature recognition would require a cognitive understanding of the feature, unavailable to a machine search. To define points in space and query whether they lie on the surface of the CAD model does not take the model precision into account. A robust method of specifying points is to create a three dimensional grid or lattice and determine what points on the model surface intersect this grid. Several other means exist, such as specifying points of constant or locally maximum curvature, or exporting a neutral STEP geometry model.

Because a computer is a finite precision machine, the numeric values corresponding to coordinates of a model vary in precision. The summation of a least-mean-squares difference between matrices of individual values gives a probability of two sampled CAD models being equal.

4.5 General considerations and limitations in scope

In the case of all commercial and open source CAD programs examined, feature creation is a subset of listed CAD API commands. The parameters that define a feature are equivalent to the parameters in the CAD API function. Modern feature-based CAD programs define geometry via a directed acyclic graph of feature commands. An ontological structure of CAD features is equivalent to a structure of commands.

This method is currently limited to extraction of feature equivalency between CAD systems. Abstract processes such as exporting a document cannot be tested through geometrical comparison of a model. The validation method requires that both test outputs exist within a domain tractable to automated reasoning; extensions of the method beyond comparative geometry are beyond the current scope of this research.

The mapping between feature-based CAD systems as formulated within an interlingua ontology can be represented as the union of feature concepts between two systems. An automated ontology extraction cannot ascribe meaning, or qualia to instances; the ontology is a compendium of statistical matches.

An algorithm that increases the complexity of search parameters to determine semantic mapping cannot be guaranteed to complete in polynomial time. A practical

implementation of this method cannot be considered to be truly automatic; the time to reach an individual mapping within the search space governs the decidability of the mapping.

An ontology that is derived from existing formats can form a basis for a standard that represents existing concepts, but perhaps more significantly, can identify incompatibilities between existing formats. The method outlined here generates an ontology that is both extensible and of minimal commitment. This approach to CAD feature mapping is an alternative to other means of specifying interoperability that is not reliant on either vendor or user adoption.

5 Future directions

The method of using the transitive properties of program output to determine functional equivalency can be generalised. Where the output of a program has a isomorphic mapping to a formal system, the output can be machine checked for equivalency. Where the input search space of programs can be partitioned to computationally practical sets, a combinational approach of checking program output equivalency against a set of input parameters is a means of automating mapping discovery.

Access to manufacturing production efficiency is proportionate to access to the software that modern production relies on. Where standardisation initiatives and market forces have failed to commoditize manufacturing production software, an automated mapping process creating a post hoc standard may succeed.

Bibliography

Altidor, J., Wileden, J., Wang, Y., Hanayneh, L., Wang, Y., (2009), "Analyzing and Implementing a Feature Mapping Approach to CAD System Interoperability", 2009 ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE2009)(DETC2009-87539).

Ball, A., Ding, L., Patel, M., (2008), "An approach to accessing product data across system and software revisions", in *Advanced Engineering Informatics*, pp.222-235.

Barber, S., (2010), "Experience in Development of Translators for AP203 Edition 2 Construction History", in *Computer-Aided Design and Applications*, 7(4), p.565, (online) http://www.cadanda.com/CADandA_7_4_565-578.html (retrieved December 9, 2010).

Bernstein, P.A., Rahm, E., (2001), "Generic Schema Matching , Ten Years Later", in *Proceedings of the VLDB Endowment*, Vol.4, No.11. (online) http://www.vldb.org/pvldb/vol4/p695-bernstein_madhavan_rahm.pdf (retrieved September 30, 2011).

Bittner, T., Donnelly, M., Winter, S., (2005),"Ontology and semantic interoperability", in *Large-scale 3D data integration: Problems and challenges*, (Eds.) D. Prosperi and S. Zlatanova, CRC press (Taylor & Francis).

Brunetti, G. & Grimm, S., (2005), "Feature Ontologies for the Explicit Representation of Shape Semantics" Related Research Areas, in *International journal of computer applications in technology*, Vol.23, No.2/3/4, pp.192-202.

Ciocoiu, M., Nau, D., Gruninger, M., (2001), "Ontologies for integrating engineering applications", in *ASME Journal of Computer Informatics Science & Engineering*, Vol.1, No.1, pp.12–22.

Dartigues, C., Ghodous, P., Gruninger, M., Pallez, D., Ram, S., (2007), "CAD/CAPP integration using feature ontology", in *Concurrent Engineering* Vol.15, No.2, pp.237-249.

Fowler D. W., Sleeman D. H., Wills G., Lyon T., Knott D., (2004), "The Designers' Workbench: Using ontologies and constraints for configuration", in *Proceedings of the 24th International Conference on Innovative Techniques and Applications of Artificial Intelligence (AI2004)*, Cambridge, UK.

Gielingh W., (2008), "An Assessment of the Current State of Product Data Technologies", in *Computer-Aided Design,* Vol. 40, No.7, pp.750–759.

Gerbino S., (2003), "Tools for the Interoperability Among CAD Systems", Proc. of the XIII ADM – XV INGEGRAF International Conference on Tools and Methods Evolution in Engineering Design, Cassino, June 3.

Gruber T.R., (1995), "Toward Principles for the Design of Ontologies Used for Knowledge Sharing", in *International Journal of Human Computer Studies* Vol.43, pp.907-928.

Horst, J. A., Hartman, N., Wong, G., (2010). "Metrics for the Cost of Proprietary Information Exchange Languages in Intelligent Systems", in *Proceedings of the 2010 Performance Metrics for Intelligent Systems (PerMIS) Workshop* pp.82-86 Baltimore, September 28-30.

ISO 10303-203: 1994, Industrial automation systems and integration – Product data representation and exchange – Part 203: Application Protocol: Configuration controlled 3D design of mechanical parts and assemblies.

Kim, J., Pratt, M., Iyer, R., Sriram, R., (2008), "Standardized data exchange of CAD models with design intent", in *Computer-Aided Design*, Vol. 40, No.7, pp.760–777, (online) http://linkinghub.elsevier.com/retrieve/pii/S0010448507001625 (accessed September 9, 2010).

Kosanke, K., (2005), "ISO Standards for Interoperability: a comparison", First International Conference on Interoperability of Enterprise Software and Applications, INTEROP-ESA'2005, Geneva.

Kubotek USA (2006), "The 2006 CAD interoperability survey results: a Kubotek USA study of the design and manufacturing marketplace," (online) http://www.kubotekusa.com/company/inter opsurvey/index.asp (retrieved September 30, 2010).

Lee, Y.S., Cheon, S.U., Han, S.H., (2005), "Enhancement of CAD model interoperability based on feature ontology", in *Journal of Ship and Ocean Technology SOTECH* Vol. 9, No. 3, pp.33-4.

Lei, M., Jovanovic, V., Qiong, L., & Tomovic, M. (2007). Data exchange among heterogeneous CAD/CAM systems from a PLM perspective. *Proceedings of 2007 International Conference on Comprehensive Product Realization*, Beijing, China.(online), http://www2.tech.purdue.edu/cimt/Courses/cimt311/cad_interop.pdf (retrieved July 30, 2012).

Mckenzie-veal, D., Hartman, N.W., Springer, J., (2010), "Implementing Ontology-based Information Sharing in Product Lifecycle Management", 65th Midyear Conference Engineering Design Graphics Division of ASEE October 3 – 6, Houghton, Michigan.

Nieto, M.A.M., (2003), "An overview of ontologies, Technical report", Center for Research in Information and Automation Technologies and Interactive and Cooperative Technologies

Lab, Universidad De Las Americas Puebla, (online) http://www.starlab.vub.ac.be/teaching/ontologies_overview.pdf (retrieved September 30, 2011).

Patil, L., Dutta, D. & Sriram, R., (2005), "Ontology-Based Exchange of Product Data Semantics", in *IEEE Transactions on Automation Science and Engineering*, Vol.2, No.3, pp.213-225. (online) http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1458714 (retrieved September 30, 2011).

Ray, S. R., Jones, A. T., (2003), "Manufacturing interoperability", in *Concurrent Engineering: Enhanced Interoperable System,* Proceedings of the 10th ISPE International Conference, pp.535-540.

Rothman A., (2006), "Airbus Vows Computers will Speak Same Language After A380 Delay", Bloomberg News, (online) http://www.bloomberg.com/apps/news?pid=newsarchive&sid=aSGkIYVa9IZk (retrieved September 14, 2010).

Seo, T.S., Lee. Y, Cheon, S.U., Han, S., Patil, L., Dutta, D., (2005), "Sharing CAD models based on feature ontology of commands history", in *International Journal of CAD/CAM,* Vol.5.

Sanchez, R., Mahoney, J.T., (1996), "Modularity, flexibility and knowledge management in production and organization design", in *Strategic Management Journal,* Vol.17, pp.63-76. Shah, J. J., Mäntylä, M., Parametric and Feature-based CAD/CAM, John Wiley & Sons, Inc., 1995. ISBN: 0-471-00214-3, pp10-18.

STEP Application Handbook ISO 10303 Version 3 (2006), *pp14 also pp145, 155* SCRA, North Charleston, SC 29418, (online), www.uspro.org/documents/STEP_application_hdbk_63006_BF.pdf (retrieved September 30, 2010).

Sudarsan, R., Fenves, S.J., Sriram, R.D., Wang, F., (2005), "A product information modeling framework for product lifecycle management", in *Computer-Aided Design*, Vol.37, No.13, pp1399–1411.

Tassey, G., (1999), "Interoperability cost analysis of the US automotive supply chain", Research Triangle Institute final report, RTI Project 7007-03.

Uschold, M., Gruninger, M., (1996), "Ontologies: principles, methods and applications", in *Knowledge Engineering Review,* Vol.11 No.2 93–136. p17.

Wache, H., Voegele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H. and Huebner, S., (2001), "Ontology-Based Integration of Information - A Survey of Existing Approaches", in *Proceedings of the IJCAI-01 Workshop on Ontologies and Information Sharing*, Seattle, WA, August 4-5, pp.108-118.

Yahia, E., Yang, J., Aubry, A., Panetto, H., (2009), "On the use of Description Logic for Semantic Interoperability of Enterprise Systems", in *On the Move to Meaningful Internet Systems: OTM 2009 Workshops* (Meersman R., Tari Z., Henerro P. (Eds)), 4th IFAC/IFIP

Workshop on Enterprise Integration, Interoperability and Networking (EI2N'2009), Vilamoura, Portugal, Springer Verlag, Lecture Notes in Computer Science, LNCS 5872, ISBN 978-3-540-88874-1.

Yang, Q.Z., Miao, C.Y., (2007), "Semantic Enhancement and Ontology for Interoperability of Design Information Systems", in *Computer Engineering*, pp.169-176.

Yeo, I., (2009), "Matching methods for semantic interoperability in Product Lifecycle Management", PhD dissertation University of Michigan. (online) http://deepblue.lib.umich.edu/bitstream/2027.42/64796/1/yeoil_1.pdf (retrieved September 30, 2011).

Young, R.I.M., Gunendran, G., Chungoora, N., Harding, J., Case, K., (2009), "Enabling interoperable manufacturing knowledge sharing in PLM", in *Proceedings of the 6th International Product Lifecycle Management Conference*, University of Bath, Bath, UK.

Yusuf, Y.Y., Sarhadi, M., Gunasekaran, A., (1999), "Agile manufacturing: the drivers concepts and attributes", in *International Journal of Production Economics*, Vol.62 No.1-2, pp.33-43.

Dynamics and organization in a digital collective: the case of an open educative publishing project

Clément Bert-Erboul
clement.bert-erboul@ed.univ-lille1.fr
Faculté de sciences économiques et sociales, Université Lille 1
CLERSE CNRS UMR 8019

Abstract. Starting from a fieldwork investigation based on mailing lists study and a set of interviews and observations, we show the successions of contributors' commitments and withdrawals in a French digital collective using free license on a publishing project. Hobby for someone, job action for others, the digital collectives are an example of the interference initiated by the use of a personal computer between work and leisure. Our investigation focuses on a digital mathematics teacher's: Sésamath. We show, through this example, how a digital contributors' collective with employees and volunteers, amateurs and professionals, regular or transient, is born, grows and maintains its existence, over a collective process of fact, a succession of events. We contribute to the debate on the issues of coordination and production in those organizations in which the constraints of the labor market or bureaucratic hierarchy do not weigh on contributor's commitment.

Keywords: nonprofit organizations, digital collectives, open source software, open education.

1 Introduction

The group's activities on the Internet are different from the traditional associative activities because of their contributors' reaction speed and expanse of their social network. Often the "free" organizations are study only once they are formed, by taking the examples of the operating systems deriving from the famous Linux core operating system. In these studies, the dynamics of the collective have often a detached anecdotic meaning of the analysis at time t of the organization. Our analysis is interested in the constitution and the maintenance of the activities of one of these digital collective: Sésamath. By studying this French association of mathematics teachers (Organization registered as a non-profit association in accordance with the French law passed in 1901), we clarify the social ordeals which frame these "distant collectives" in a grey area between personal and professional

uses of computer. In the case studied, contributors use their free time in the construction of on-line teaching contents.

We define “process” to analyze the social dynamic, as successions of associations and ordeals quite close to the observed phenomena of agreements, enrollments and betrayals by the actor network theory when it studies scientific processes (Callon 1986). Thus, the actors' commitment within the group is a set of coherent behaviors which could evolve during time according to the situations (Becker 1960). The possible evolution of commitment situations obliges the individuals to redefine their commitment “motivations”. Alone or collectively contributors can stop recognizing as coherent the behaviors of the group, and leave it. In that case we will name it "withdrawals ". During these ordeals we observe negotiations between all or some one of the group. At each step, the collective can be started again or disappear following the departure of members.

Organized around the technical transformations of the Internet and microprocessing, the “free” movement preaches the “free licenses” property licenses allowing copy, study, modification and redistribution of contents, and prohibits the individual appropriation of work collectively realized (Stallman and al. 2011). The “free” movement, results from the software developers former practices (Weber 2004, Levy 1984), which have been adapted to the publishing field (Lessig 2004). In the French education sphere, the Sésamath association transposed the practices of free software development towards the writing of secondary school papers textbooks. This digital collective, composed by voluntary mathematics teachers, produced and broadcast teaching contents on the Internet under free license.

Without seeking to dissociate the digital collective from their agents, we focus on the free licenses' effects in the medium-term implications in the organization. The example of Sésamath provides us elements to build a kind of an “ideal type” of the digitals collectives’ process. We present the stages of the summarizing the commitment process. The process includes four stages: 1) the collectives construction carried out by heterogeneous groups, 2) the group's technical and organizational amendments to use free licenses, 3) the construction of a name carried as a label and 4) development of an economic model.

2 Ordeals of digitals collectives

Unlike the traditional productive organizations studied, the sustainability of a digital collective is not a primary objective for its members. In a digital collective, the contributor's livelihoods are not threatened by the disappearance of the online group. To a volunteer, leaving Sésamath is neither abandoning his salary nor being excluded from a dynamic professional socialization. In the case of a digital collective the contributor is not participating in a “community”. The traditional sociological meaning of that term (Tönnies 1944) announces stable social relationships based on emotional ties or traditional, family, tribal, or religious bonds. The term of “virtual community”, often used by early Internet sociologists (Rheingold, 1993) is

misleading. Our approach in terms of "collective" rather than "community" (Proulx and Latzko-Toth 2005) indicates that contributors have a common activity without the members of the group being in a cohesive, or a "mechanical" and "coercive society (Durkheim 1950),"or totalitarian "(Goffman 1972). The use of the term "collective" reflects the freedom of each individual to enter or leave the group.

We determined that the Sésamath collective's process is marked by four ordeals. Sésamath beginins met a birth ordeal because the first contributors were in different fields. The group underwent a legal ordeal in use free licenses on teaching contents. In a third point we show that the group also faces an ordeal of using to finding new contributors. Finally we studied the economic ordeal when the group's members decide on the management of their resources.

2.1 The birth ordeal: where digital collective come from?

Sociologists have on a long time insisted on the social construction of technical choices. We insist on the fact that all that is digital is not constructed in a social vacuum (Fligstein 2001). Unlike to an **autarkic** vision, we think that the digital collectives do not appear ex nihilo. The example studied in the education field shows relations between heterogeneous actors, where we find teachers, inspectors, and publishers. These relationships lead us to deconstruct the subversive label which marked actors of the Internet. Our data show that the digital collectives do not start from a struggle and they are not built against an existing actor. Fork and struggle can appear later but the beginning of a digital collective is a merging of heterogeneous groups of actors. Our point of view is that digital collectives are constructed by a set of variable collaborations independent from commercial competition.

2.1.1 A brief history of Sésamath

Today (in 2012) the Sésamath association counts 79 active members, including 6 employees, plus 100 regular contributors. All are or were mathematics teachers in France. The Sésamath association allows a publishing company to make paper textbooks, exercise books and software which are available free licensed on-line for free. The Sésamath's mathematics textbook accounts for 15% of the French mathematics secondary school textbooks market[1]. Digital teaching collectives were also constituted around other disciplines (in Biology, English, History and Geography and Letters). However these collectives do not have the glory that Sésamath had in the Technologies of Communication and Information for school. Our work emphasizes elements to understand this difference in durability.

In our study, Sésamath appears following the connection between a pedagogical research laboratory and teachers having activities on the Internet. Between 1999 and

[1] The French textbook market is quite open. Anybody can sell a textbook. There is no institutional label or a rule making. It's the teacher's pedagogic choices which referee between sold textbooks.

2002 the teachers' mailing lists, the personals web site, and a research program of the Academic Institute of Masters Formation (IUFM) have converging activities with individuals participating in these three activities. As the email extract shows below, Web mathematics teachers claim recognition from their establishment. The teachers wish that their on-line activity be taken into account by the National Education ministry as a part of their teaching work.

"[…] it appears urgent to me, like to many others, to federate the actions and to support the mathematics teachers investing themselves for their discipline. The individual actions carried out by the teachers of mathematics, in particular on the Internet must be recognized and supported (by important resources) by our institutions. They bring much to those who seek information, a dialog, etc… and contribute to the continuing training of teachers […] More personally, I wish to know if you want " to link" Sésamath with the list "the maths in secondary school list" which I manage. The list could be one of a Sésamath's components (like other sites)". **Email extract from an teachers' mailing list organizer to a Sésamath's contributor January 2002**

The first agreement from educational establishments with the teachers present on the Internet is found in 2002. The Sésamath's contributing authors published with the National Documentation Teaching Center[2] (CNDP) a paper exercise book. However, these initial arrangements fail three years after their application and the on-line activities of the teachers will never be paid directly by the National Education ministry as a salary. On the one hand in 2005, the exercises' books head author withdrawals Sésamath because he was motivated by the hope of money. On the other hand local officials of CNDP decide to exit from the paper exercise books project for intern political reasons and because paper exercise book was not a profitable project. Concerning the collective's on-line activities, Sésamath closed its principal collaborative website because of the pressure from the academic establishment and the websites users. Inspectors take a dim view of the access to the "massive ready to think" provided by the website. Teachers criticized the too uneven quality contents. Following this website closure, some Sésamath contributors implicated in this project left the collective.

After these failures Sésamath rebounded and created a website articulating paper and digital contents. This new presentation was carried out in partnership with two digital school publishers. Paper textbook and exercise book marketing generate new flow of economic and symbolic resources. The textbooks were greeted in French "free" protagonists with a gold Lutèce [3] prize and Sésamath receives a UNESCO prize. These distinctions led the collective to participated to the educational policy debates.

[2] CNDP is the French Education's Ministery publisher and works with regional relays.

[3] The Lutèce prize are decreed every year by the National Free software Industry federation during the "Paris Capitale du Libre "demonstration

2.1.2 An outsider in the school publishing

We are aware that school mathematics contents Publisher Company did not await Sésamath to diffuse their own contents on Internet . However the collective modified the field gravity center (Bourdieu 1999) by using new productions rules to write school textbooks. The Sésamath's slogan says "Teachers' tools, made by teachers". The communication tools diversification allow new actors to produce, to evaluate and use collective contents. The digital collectives entrance on the publishing's market, delete some economic and social monopolies. In the case of Sésamath, the teachers' mailing lists and personal websites have shorted-circuit the IUFM researchers, the CNDP's officials and the inspectors of Academy for the teaching contents production. Previously, this panoptical monopoly was acquiring over rounds of inspection. In our case of study the aggregation of individuals' experiences through new means of communication on the publishing's market have overflowed the legitimacy of the inspectors. Dynamics convergence obtained is expressed by this contributor.

"The idea of Sésamath at the beginning is really simple: here, we have a media, and this media allow an important grouping of people, and if we leave these people in an isolated way, we don't have this mass phenomenon. If the whole is undoubtedly gathered, if we have a visibility together, that will increase the total motivation, and it will allow to others to come to incorporate itself... finally it is a snowball phenomenon." **Sésamath's contributor Interview February 2009.**

During first actors turnover of we see that the collective do not disappear. After test of digital pedagogical experiences a professional claim movement appears concerning the recognition of on-line voluntary teachers' activity by the National Education. Thereafter members of the collective engaged a commercial logic with publishing companies. The succession of associations and activities lead the collective to centralize specific digital, symbolic and economic resources. In the next part we will see how web teacher protect them.

2.2 The legal ordeal: be or not to be free

All the authors do not protect their works under free licenses because of social relations, or economic model, or technical or legal compatibility. The collective choice of the use of the licenses has various effects during time. In a first time teachers has chosen free licenses to react to a separation between unpaid and teaching activities. Then the free license has a medium-term social dimension and allows broadcasting information. On long range the free license seems to frame contributors' coordination.

2.2.1 The school discipline inequality front of the free licenses

The free licenses use by teachers modifies the linked relations until 2001 between websites' administrators and contributors, the user lists subscribers, the IUFM's officials and the Academy's inspectors. The legal formalization of the free licenses breaks the hierarchical report binding the teachers to their guardianship institution. The contents' copyright collectively realized cannot be unilaterally changed by the administration. This legal provision obliges the institutions to treat the authors teachers according to a different register of the hierarchical mode as this contributor of Sésamath claims it.

"We cannot ask an institutional support and reject the members of the institution… Where the territory should be "marked", it is that they have (like all other teachers besides) the right to express criticisms and restrictions, but that must be with respect for our work and without hierarchical point of view. " **Sésamath's contributor Email January 2001.**

However the amazing feat realized by the Sésamath's contributors partly depends on the properties of the mathematical contents. The pooling of digital production is framed by two important limits. On one hand authors have to make a digital copy of contents and on other hand they need possess copyright to pose a free license on these contents. All objects do not lend themselves to digitalization for practical or legal constraints. The similarity between the software and mathematics textbooks was already underlined by Richard Stalleman himself, the founder of the free licenses. The legal and technical proximity between a piece of source code and another contents facilitates the use of free licenses. The mathematical tools are not patentable, the exercises are written with the digital format and their authors are free to determine the restrictions of copyright which they wish apply to their works. In 1989, when the first free license is published, the Public General GNU License, "treated a program much like an algebra textbook: its author can claim copyright on the text but not on the mathematical ideas of algebra or the pedagogical technique employed to explain it" (Stallman and al. 2011).

In other fields the use of free license is much more delicate for practical matters. To carry out a history textbook, authors need access to spaces belonging to private owners (Pinçon and Pinçon-Charlot 2007) which are difficult to penetrate for private individuals wishing to widely broadcast the reproduction of these domains. The transposition of the "legitimate culture" to a mass media attenuating the social distinctions seems to meet practical barriers which reveal social hierarchy. The digital reproduction of the paintings requires the agreement (often invoiced) from the owners to place the painting in good conditions to digitalizes it. There, without support from the owners, a large scales digitalization seems impossible for a distant volunteers group.

2.2.2 The free license's organizational effects

In addition to the social and technical sides attached to the released objects, free license frames the volunteers' organization. If we do not deny the charismatic effect from some contributors, the contractual relation instituted by the licenses stay the most largely shared. To transform content's circulation rules , all the authors are contacted to validate the change wished on the license. The contributors in activity but also the former members which no longer contribute can give their veto on the transformations brought to their collectives contents broadcasting rules.

First used by teachers to react to a power balance with their hierarchy, the free licenses prove to be particularly compatible with the mathematical contents. However, the distinctions between legitimate culture defended in the historical places and the mass culture available on Internet limit the dynamic ones of constitutions of the digital collectives. Over time, the licenses prove to have an organizational characteristic and maintaining the contributors in the collective in spite of the end of their activities. In the next part we will see how the collective attracts these contributors.

2.3 The use ordeal: how make a name?

With the rise of micro computer and the Internet , write and diffuse mathematics exercises is not only reserved to teachers in their classrooms. Publisher companies, on-line education companies, pupils' parents, and single teachers edit mathematical exercises on Internet. Lost in the users' mass of web editors tools, information producers and diffusers announce themselves on Internet by their names. Initially dedicated to the secondary school teachers, Sésamath attracted also student and their parents, and concerned teachers by the primary education. Moreover, collective realizations of Sésamath are concerning also publisher companies, chiefs of establishments and persons in charge of teaching policy. This widening of public represents part of the collective's social resources. Indeed, the Sésamath collective is maintained with aggregation in a regular way of new contributors. The teachers join the collective, after they made personal adaptations to contents or during seminars or teaching formations where committers meet inside their professional time.

2.3.1 The new collective's commiters

These new participants' arrivals counterbalance the tiredness, the disinterest and the discords which justify the abrupt or progressive departures of free contributors from their commitment and their withdrawals. The entry and exit contributor's movement is framed in an organization in concentric circles (Demazière and al. 2003) with porous borders. A historical core and associative in charge members forms the center. These individuals have collective's administrative responsibilities, but also assume the collective's animation and organization. A group of co-opted contributors will widen the size of the collective around the core. Users extend

around these two first circles and a fuzzy limit distinguishes public and the largest crowd from the other Net surfers. The contributors' integration is progressive. On one hand, this movement is observable with the evolution of indicated addresses by the contributors to the server of mailing lists. On other hand, the progressive withdrawals can be shown by unsubscription to the various lists by the moderators.

To stay despite these seldom announced departures, the collective regularly hire new contributors. The co-optation of contributors integrates newcomers and their contributions under a name centralizing specific digital, symbolic and economic resources. The mark - Sésamath - is a kind of a "investment of form" (Thévenot 1986) which facilitate to net surfers to identify what is that teachers' digital collective. This centralization objectifies the heterogeneous contents, and incorporates free contributors to enter and leave the group. Thus, the generations of contributors follow one another without attaching the contents to a personal name or an individual competence. By using the free licenses, the first members of Sésamath opened editions rights to outside contributors from the first historical core.

2.3.2 Commitment and hiring in a digital collective

Thus, the activity within Sésamath is related to a dialog between the individuals and the collective. Contributors voluntarily commit or are hired by co-optation. They can leave gradually from the collective or get out from the group after a brutal collective ordeal. Teachers engage in Sésamath because they defend the political ideas of the group or, because it wishes to carry out new exercises. On its side, the Sésamath collective hires an experts committee made up of teachers to ensure his social resources. We noted that digital collectives are in competition to attract contributors. But like there is an entry movement in the group there is an exit flow. The contributors leaves Sésamath by lassitude or following a rupture of tacit agreements after the rise of new actor.

The pooling of the separated teachers gave a symbolic value to the collective's members' carried out contents. Recruitment within an experts committee, the professional recognition, the updated contents accumulation by an active group and regularly remodeled, and the recourse to the free licenses, distinguishes the name from Sésamath compared to the other on-line mathematics actors. The centralization of heterogeneous contributions, published under free licenses within a collective work bearing a name, defuses the opportunist uncertainty behaviors since individually the contributors cannot take and close the whole of the contents of the collective. In the next part we will understand how all these contributors stay together.

2.4 Economic ordeal: how to have an economic model?

The actors' heterogeneity and the digital collectives' legal construction influence the sharing of digital, symbolic and economic resources accumulated mainly by

nonprofit volunteers' activity. Three factors seems important to consider the digital collective economics. The technical architecture requires a collective and centralized funding to distribute the realized contents beyond the contributors' circle. Individual motivations involve often professional implications. The commitment is also influence by a collective dynamic with a form of rationality sometimes in opposition with the technical requirements or individual choice.

2.4.1 The digital collective's technical structure

The collectives' economic model which realizes free licensed contents was yet analyzed by Michel Gensollen. This French economist insists on the material aspects of their organization and their file sharing processing system (Gensollen 2004).

1. Each contribution is unique. The exercises from contributors treat too different teaching points and cannot be compared between them according to their lengths or their difficulty level. This uniqueness is reinforced by the label reference enabling to position each exercise in the whole of the textbook. This ranking is used to entitle the emails on the mailing lists. Every five years these titles reappear when the group upgrades the textbook.
2. The digital contents are not rival goods. The exercises' electronic format and the use of free license make them available free on Internet without harming another user. This characteristic is extended by the association's technical and pedagogical maintenance which makes the contents imperishable. This temporal availability increases the attractiveness of the stored contents. More content is read and used, more it is improved in successive versions by his users.
3. The digital collectives are made up around a decreasing return to scale. The production of an exercise is not very grasped in resources. On one hand, a computer, some software and an Internet connection are technically sufficient to individually start the production. On the other hand, providing the whole contents for every student in every school requires more and more work from the association's members. Since 2005 in response to the increase in activity the digital collective has hired employees, to help the digital collective's contributors. The increasing contents success on Internet requires to programming powerful and complex servers.
4. The aggregate contents are complementary. The exchanged textbook exercises are a part of a biggest educational corpus tools including dynamic geometry software, and websites. Taking part in the contribution of one of these tools enables a contributor to come into contact with the whole of the contributors of the collective and to diversify the shapes of its own contents. For instance a paper exercise can be translates by an animating web application by a collective's programmer

2.4.2 Individuals' motivations

Analyze collective's equipment organization is not sufficient to look at the collective's organization's processes. On other way, studies on Instrumental

commitment in the digital collectives insist on the actors' inscription in a home consumption dynamics and on formation. Within this framework, collective's membership is often an important symbolic and social investment to progress in the professional life (Lerner and Tirole 2002). Indeed, the enhancement of the on-line individual activities by peers is a resource which the contributors do not find, neither on the job market, nor within the hierarchy. However, our fieldwork shows that the individual and technical activities are framed by social processes reflecting the "sociopolitical relation which have course within the profession and in its relation with the remainder of the society" (Convert and Chantraine-Demailly 2007).

2.4.3 The collective's dynamic

In the case of Sésamath, commitment and withdrawals are influenced, neither by a technology matter (there is not an automated system of mailing lists registration), nor by a question of subsistence, since all the contributors live from their teaching of mathematics in schools. Contributors were initially remunerated to write the paper edition, but the economic model of the collective was quickly directed towards a mixed organization articulating nonprofit volunteer authors and economic remunerations for communication and technical development activities. This choice follows the collective's separation with the national education in 2001. With the incomes from the textbooks' sale, Sésamath finances employees to develop and maintain its massive web collaboration spaces and to promote its contents to users. However this mode of redistribution of the resources is the result of negotiations where several approaches are confronted. Within Sésamath, the choice to preserve or not employees, and thus to have a complex technical architecture to maintain on-line the contents stock was the subject of a conflict between contributors. Following the decision to increase the remunerated activities in the collective for programming, some members (as that quoted below) left the group.

"Be paid part-time by Sésamath returns us a distorted idea of Sésamath, making us forgets that the others are unpaid volunteers and that they cannot act within the same times. The ditch is thus even larger with a full-time job that explains why I proposed to remove it. Live Sésamath, eat Sésamath, sleep Sésamath does not help necessarily Sésamath because one forgets which sacrifices made by those but without having time released to do it. Also, I am also in favor to decrease the number of the employees. If the money to spend is still too important, I prefer by far to support members' meetings and their material equipment." **A Sésamath's paid member Email June 2009**

The previous quotation of the Sésamath's employee wishing that the collective do not maintain his employment shows that the recourse to the instrumental motivations cannot be extrapolated. The ascetic morals shared by contributors since the break between the collective and the national education established a nonprofit organization rules. The commitment lead by instrumental aspirations is not the only engine of the collective action. Moreover, the symbolic resources from commitment in the collective is sometimes negative because of power struggles in a bureaucratic

organization (Crozier 1963, Alter 1993). In the Sésamath case, teachers are sometimes in front of inspectors opposed to the educational tools of the collective. Furthermore, the importance of commitment is often disproportionate compared to the professional repercussions. Sometimes commitment is prolonged even if the possibilities of professional progression are exhausted because of retirement.

Finally studying a digital collective as Sésamath is to study a perimeter unceasingly redefined by distant contributors' commitments and withdrawals. These contributors' back and forth pass rhythm of the stages in the organization of Internet groups. The first stage underlines the possibility of connection of heterogeneous actors opened by Internet. The second stage indicates the legal tendencies linking the contributors between them. The third stage insists on the part played by contributors' centralization around a common noun. Lastly, the fourth stage shows that the economic springs are important but not sufficient to explain the durability of the collective and that the preceding three stages are supporting it.

3. Conclusion

Neither coordinated by a bureaucratic logic, nor by labor market rules, the Sésamath organization was built on other criteria. The stages whereby the collectives realize, or not their process are validated or must be restarted through a succession of commitments and withdrawals. Our ethnographic work mainly carried out on Sésamath group reveals four stages of organization. We confirm that the groups on the Internet do not appear ex-nihilo. Indeed, the initial contributors approach come from already existing organizations. We show that the organizations adaptation to the free licenses' constraints requires a round of negotiations influenced by broader social structures such as the opposition between legitimate culture and mass culture. We signal that the process of a collective's construction needs symbolic, economic and digital resources accumulation under a reorganized name which attracts the contributors and identifies the group to the users. The use of symbolic resources as in the second stage is influenced by broader social structures. Finally we insist on the fact that the economic model and the use of the technical tools of a collective depend mainly on volunteers' commitment influenced by external representations and social constraints.

Of course, this study is not exhaustive, but seeks the typical elements of an association building digital collectives. The evolution of Sésamath reveals that the collective is not attached to individualities but to a set of actors who can leave the group. The characteristic of the producers of free licensed resources give a special place in the Internet's uses social sciences studies.

References

Alter N. 1993. Innovation et organisation : deux légitimités en concurrence. Revue française de sociologie:175-197.

Becker HS. 1960. Notes on the Concept of Commitment. American Journal of Sociology 66(1):32-40.

Bourdieu P. 1999. Une révolution conservatrice dans l'édition. Actes de la recherche en sciences sociales:3-28.

Callon M. 1986. Éléments pour une sociologie de la traduction. L'Année Sociologique 36:169-208.

Convert B, and Chantraine-Demailly L. 2007. Les Groupes Professionnels Et L'Internet : L'Harmattan.

Crozier M. 1963. Le phénomène bureaucratique: essai sur les tendances bureaucratiques des systèmes d'organisation modernes et sur leurs relations en France avec le système social et culturel: Université de Paris. Faculté des lettres.

Demazière D, Horn F, and Jullien N. 2003. Le travail des développeurs de logiciels libres. La mobilisation dans des communautés distantes. Cahiers Lillois d'Economie et de Sociologie 46:271-194.

Durkheim É. 1950. Les règles de la méthode sociologique: Presses universitaires de France.

Fligstein N. 2001. Le mythe du marché. Actes de la recherche en sciences sociales:3-12.

Gensollen M. 2004. Economie non rivale et communautés d'information. Réseaux 124(2):141-206.

Goffman E. 1972. Asiles: études sur la condition sociale des malades mentaux et autres reclus: Ed. de minuit.

Grossetti M. 2004. Sociologie de l'imprévisible: Dynamiques de l'activité et des formes sociales: Presses Universitaires de France.

Lerner J, and Tirole J. 2002. The Simple Economics of Open Source. The journal of industrial economics, 50(2):197-234.

Lessig L. 2004. Free Culture: How Big Media Uses Technology and the Law to Lock Down Culture and Control Creativity: Penguin Press.

Levy S. 1984. Hackers: Heroes of the Computer Revolution: O'Reilly Media.

Pinçon M, and Pinçon-Charlot M. 2007. Les ghettos du Gotha: comment la bourgeoisie défend ses espaces: Seuil.

Proulx S, and Latzko-Toth G. 2005. Mapping the Virtual in Social Sciences: On the Category of "Virtual Community".

Rheingold H. 1993. The virtual community: homesteading on the electronic frontier: MIT Press.

Stallman RM, Williams S, and Masutti C. 2011. Richard Stallman et la révolution du logiciel libre: Une biographie autorisée - Une initiative Framasoft: Eyrolles.

Thévenot L. 1986. Les investissements de forme. In: Piore MJ, editor. Conventions économiques: Presses universitaires de France.

Weber S. 2004. The Success of Open Source: Harvard University Press.

Examining FLOSS Community Cohesion: An analysis of communication and collaboration patterns in open source

Darren Forrest
School of EECS, Oregon State University, Corvallis OR 97331, USA
forrestd@onid.orst.edu

Abstract. The success of Free/Libre Open Source Software (FLOSS) is driven by the efforts of individuals all over the globe, many of whom are volunteers. Larger FLOSS projects often have a mix of volunteer and paid developers, as their coordination needs and workload can be higher. This influx of paid developers can come at a price, with the community changing and possibly fracturing. It is therefore important to map and understand the evolving dynamics of such mixed communities, and to identify best practices. I propose to study FLOSS project structure, communication history, and social network structure to determine the characteristics that foster a more diverse, inclusive, and interconnected community. This research will include a longitudinal multiple case study covering several large FLOSS projects with known high levels of paid developer contributors.

Keywords: FLOSS, Community structure, social network analysis, conflict, corporate participation, best practices

1 Introduction

Free / Libre Open Source Software (FLOSS) projects are a major part of our software ecosystem. FLOSS is used in all levels of computing, from backend technical systems to end user software on tablets and phones. Users of all technical levels benefit from the capabilities that FLOSS brings to the table, and much of our development and Internet infrastructure depends on a healthy and vital FLOSS community.

FLOSS development differs from "traditional" closed-source software in a number of fundamental aspects. One is that it is not only possible for anyone to view and use FLOSS code, but that projects depend on an open participation model where anyone can contribute, and where the best ideas win. This development ideology is a key strength, as it enables a large and diverse group of developers to pool resources to benefit everyone.

The distributed organization and development model of FLOSS projects is at the heart of what makes FLOSS successful. This distributed organization model requires

extensive cooperation and coordination, necessitating project maintainers and others in supervisory roles. These individuals manage the code, are responsible for choosing which contributions to incorporate, and who has the ability to submit code. Because of these powers, they have a measure of control over the direction and participation of the project above and beyond any planning or leadership activities [18, 27].

Control of the code, and thus the direction of a project, is important. A project may end up alienating, or neglecting the needs of a subset of their users if these are not represented in the project. This is a very real problem. The code-base of the Linux Kernel for instance has ballooned [28] as hardware manufacturers add support for high-performance hardware. While the rapid growth of the code-base may be of only minor concern to those running data-centers, it is a serious concern for those wanting to run Linux on netbooks. Code, though important, is not the only way to contribute to a project. Bug reporting, documentation, mailing list discussions, mentorship, or governance are also important.

It is important to understand how participation in these different activities contributes to the health of projects, and how different organizations exert influence on projects. This knowledge is not just important to the projects, but to potential adopters or developers. Understanding who is guiding the project is crucial to many decisions about whether to invest in a project. Having broad support and transparency is important; the highly public fork of the OpenOffice project serves as an example of how mistrust and conflict related to project governance can lead to serious rifts. The Linux Foundation recognizes the importance of such information as well and does a yearly report on its contributor base [13].

2 Background

2.1 FLOSS Participants and Motivations

Numerous studies have found that a majority of FLOSS developers are European or North American, Caucasian, male, and average 27 years of age [6, 7, 15, 23]. Other studies have looked at what drives people to contributing to FLOSS. These surveys found that a majority of individuals participate for altruistic and personal betterment reasons. This dynamic changes, however, when we take into account that FLOSS communities are not made up of only volunteer individuals. David et al. found that more than 40% of developers in major FLOSS projects are paid to do so [23]. Hars and Ou put this figure at 50% [7]. This suggests that the grassroots volunteer nature of FLOSS may no longer be true.

With this change it is reasonable to expect some change in what drives participation in FLOSS. Bonaccorsi and Lamastra found that the motivations for individuals and for firms were different. While individuals were primarily driven by personal and altruistic reasons, firms where driven by technical and financial motives

[2]. It stands to reason that individuals who are paid to work on FLOSS will have motivations that are more in line with the motives of their firms.

Forking is a constant concern for FLOSS projects. Sometimes both projects survive the split, other times one or neither project will survive the conflict and loss of developers. At the very least, forking increases the overall cost to the community, as it requires keeping two projects running in parallel, though more diversity is also desirable. Overall, forks are seen as something to be avoided.

2.2 FLOSS Project Data Mining

Most FLOSS studies have focused on studying bug reporting, code commits, and mailing lists. Feature tracking [5], the development process [21], knowledge reuse [14], and negotiations between developers and bug reporters [26] are among the many aspects of FLOSS that have been studied. However, most studies have only focused on one or two of these aspects.

2.3 Organizational Behavior Theory

When studying FLOSS group dynamics, organizational behavior theory provides a suitable basis for identifying relevant factors. Equity Theory has been well established as a framework for describing how individuals work together and the factors that motivate certain behaviors [19, 10, 9]. In this framework, if an individual feels the cost benefit ratio they are subject to to be unequal (positively or negatively) to that of others, they will seek to equalize the difference.

In more recent years, a large amount of research has focused on studying the effects of diversity on group productivity. While it is widely assumed that diversity is a positive force on productivity, the research has been less conclusive [11, 8, 30, 31, 16]. Faultline theory, developed by [17], provides a framework for studying diversity across measures. There is no requirement that faultlines be applied only to demographic differences, any factor that splits a group into subgroups may be considered a faultline [1]. One could look at participant affiliation as one aspect of diversity. For the rest of this paper we will define affiliation and affiliates as people who are employed or directly sponsored for their work on FLOSS.

Central to faultine theory are the concepts of groups and subgroups. A group is the main body of organization. In the case of a FLOSS project this would be the complete project community including users. Subgroups are subsets for which there is a defining characteristic they hold in common (i.e. developers develop code). In this model, individuals have identifying traits (gender, age, race, social position, wage, etc.). These traits make up the group's faultlines and hypothetical subgroups.

Research has shown that faultlines have an effect on group productivity. Groups that are very diverse (high diversity, weak faultlines) or have strong faultlines (low diversity, strong faultlines – i.e. 2 groups) have more conflict and lower morale and

performance. The relationship is U-shaped when comparing number of faultlines to the presence of conflict [29].

Faultlines also provide a framework for discussing the formation of coalitions within a group. Rabbie and Horwitz describe a coalition as a "recurring and salient subgroup" [25]. Coalitions tend to form around those members who see a problem similarly [24]. Research is divided on the inherent strength of coalitions within organizations; Murnighan and Brass found that coalitions form and fade quickly [22], but this may only be true of weak faultline groups [17].

Li and Hambrick researched the difference in faultlines of factional groups. A factional group "consists of members who are drawn from, and are expected, to some degree, to represent, a small number of social entities that exist outside the boundaries of the group" [20]. Representational research states that individuals who are aware of their status are more likely to vocalize the needs and desires of those they represent [12]. In addition, they found that when these groups also share demographic characteristics, the faultline has a greater effect.

Research has found several mitigating factors with regards to faultlines and group member stress. Bezrukova, Spell and Perry found that the presence of faultlines can be a positive influence in dealing with distress [1]. They found that individuals were more likely to seek and find support from their own subgroup and thus better deal with stressors. Faultlines that are not perceived by their members are seen as dormant, whereas, those known to their members are considered active [17]. Activation of faultlines can have a polarizing effect on the group cohesion.

In a FLOSS context, corporate affiliation could have two strikes against it: it provides an alternate identity, and it is a highly visible division within the group. Project communication revolves around email, and email addresses offer a visible reminder of corporate identity.

3 Motivation

FLOSS projects have been found to be highly demographically homogenous. This means that they likely have high strength faultlines. Participation data shows project contributors are near evenly split between those who are compensated for working on FLOSS and those who volunteer. Because of this, there could be two large subgroups within a typical FLOSS project, with a strong faultline between them. Research has shown this to lead to increased conflict within projects. Increased conflict may lead to a decrease in productivity or even developers leaving a project.

Research also shows that mitigation strategies can be used in faultline-based conflicts. Common sense tells us that knowing a potential problem is brewing is more advantageous than ignoring the issue. If a method for determining when conflict is imminent through analysis of project artifacts could heighten awareness, projects could take appropriate measures before the situation becomes critical.

4 Research Questions

Our study seeks to develop an understanding of how individuals interact within FLOSS projects; specifically in relation to the affiliation of individuals, and how those interactions shape the community. Within the context of a faultline framework, determining the perceptions of developers is important. Our research questions are:

RQ1: Given that affiliate participation is common in FLOSS, are like-affiliated members clustered around key activities to the exclusion of other community members?

Example H1: The clustering coefficient between two like-affiliated members will be greater than the coefficient between one of those members and a non-affiliated member.

RQ2: Do community members perceive a distinction between affiliated and non-affiliated project members?

RQ3: Do affiliated members perceive themselves more aligned with their sponsor or the FLOSS project?

RQ4: What are community attitudes towards affiliate participation in the project?

RQ5: Do we see more cliques forming around co-affiliated members than others?

RQ6: Do strong social circles lead to more and stronger cliques over time?

5 Proposed Methodology

Determining how individuals and firms interact in FLOSS communities requires a case study methodology. To this end I propose to study several FLOSS projects. This will involve interviews with key project members, gathering of bug repository, code versioning system (CVS), email addresses and communication patterns, and social network analysis (SNA) of these patterns.

5.1 Case Selection

We plan to study the Linux Kernel, Gcc, Libre Office, Open Office, MySQL, Eclipse, NetBSD, and OpenBSD. All of these projects are fairly large. Our preliminary work has shown that a large number of regular contributors (> 20) are necessary for a meaningful affiliation analysis. Many smaller projects do not have this number of regular contributors. These projects also are more likely to have attracted the attention of corporations and other organizations open to being

involved. Either through internal use or as part of their products, many businesses have a tangible interest in the success of these FLOSS projects and thus likely have developers involved in the project.

It is worth reiterating that the goal in selecting these projects is to select projects with large amounts of industry participation. Each of these projects is highly used by industry and has a large number of affiliated developers. In the case of projects that have undergone a fork we consider the pair as having large industry participation. Also in this selection we are choosing projects where we know industry participation has been an issue in the past and a few projects where there is participation, but that participation does not seem to have caused issues within the community. In this way, by comparing those having issues with those not having issues, for projects that have large levels of industry participation, we should begin to see a clearer picture of patterns and trends as the case study unfolds. Table 1 provides the reason for including each project as well as measures of project size.

Table 1. Project breakout

Project	LOC (millions)	Contributors	Reason for Including
Linux Kernel	24.9	9515	Mixed Industry Participation
Gcc	6.1	482	Mixed Industry Participation
Libre Office	12.8	853	Forked
Open Office	15.1	359	Forked
NetBSD	16.6	408	Forked
OpenBSD	-	-	Forked
MySQL	1.4	1188	Single Dominant Supporter
Eclipse	2.6	306	Single Dominant Supporter

Libre Office, Open Office, NetBSD, and OpenBSD have undergone forking. This means conflict occurred, and an artifact trail may lead up to the split. We will study these projects to understand the characteristics of the interactions before and after a fork occurred.

The Eclipse and MySQL projects are dominated by developers from a single corporation. In these we expect to see different social network characteristics. While these projects may not have forked, there is a community of developers within and from outside the company. It would be interesting to see how these social networks differ from other projects.

Finally, some of our projects have a more diverse set of contributors. In these projects individuals from many affiliations work together to develop software. It may be that these have entirely different network characteristics that can be studied.

The use of CVS and bug tracking systems also plays a role in the selection of projects. Table 2 provides a summary of the projects to be studied as well as the CVS and bug tracking systems they employ. The lines of code (LOC) and contributor counts were gathered from Ohloh.net. The rest of the information was gathered from

project websites. Some data is missing; in the case of OpenBSD this is due to the project not having been analyzed by ohloh.net yet.

Most of the projects use git or SVN for their CVS system. In my preliminary work, I wrote scripts to extract equivalent log data from both git and SVN. Three projects use the CVS system for their code repository. These projects will require additional scripting to obtain the necessary information, but have been included because NetBSD and Open BSD have undergone a fork, and Eclipse is a prime example of a project that is heavily supported by a corporation. MySQL Server makes use of the Bazaar CVS system with Launchpad. While this will also require additional work and may provide some difficulties in normalizing data across the cases studied, MySQL Server is another prime example of a FLOSS project that is supported primarily by one company.

Most of the projects selected also make use of Bugzilla for bug tracking. This has the benefit of providing consistent data fields for analysis. The scripts for extracting email addresses have already been completed in preliminary work. MySQL Server makes use of a system based on bugs.php.net. NetBSD and OpenBSD make use of different bug tracking systems. These two projects may provide an additional challenge in obtaining the necessary data.

Table 2. Project summary information

Project	LOC (millions)	Contributors	CVS System	Bug Tracker
Linux Kernel	24.9	9515	git	Bugzilla
Gcc	6.1	482	git	Bugzilla
Libre Office	12.8	853	git	Bugzilla
Open Office	15.1	359	SVN	Bugzilla
NetBSD	16.6	408	CVS	GNATS
OpenBSD	-	-	CVS	Sendbug
MySQL Server	1.4	1188	Bazaar	bugs.php.net
Eclipse	2.6	306	CVS	Bugzilla

5.2 Interviews of Key individuals

We plan to identify and interview key individuals within each project to find out how they view involvement based on the affiliation of the individual. These perceptions are important to determining the activity of faultlines within the group and thus the magnitude of these. This information along with their insight into the project and interaction patterns will serve as a guide to help us reach conclusions. This may also help us identify additional areas for study within these projects. The interviews will be used to answer RQ2, RQ3, and RQ4.

5.3 Artifact analysis

There are many historical artifacts in FLOSS projects. The availability of these is one reason FLOSS research has been growing in popularity. We plan to extract information from bug report databases, mailing list archives, CVS repositories and IRC metadata.

Bug reports contain information on issues encountered by users. Depending on the project and its practices, bug reports may include feature requests or other information not normally associated with a bug. A conversation may develop around a bug as developers work with users to determine the cause of a bug and reach a resolution that does not introduce new bugs. Because of these conversations, bug reports can provide a measure of how developers interact with users (users would often be a super set of developers here).

While bug reports may involve end users, developers dominate mailing lists. While each project has a different culture and norms, mailing lists provide a forum for discussion in a community. As with most aspects of FLOSS, almost anyone can take part. However, mailing lists have core members, often the project leadership, who reply and make decisions. In this way mailing lists provide a better view into the forward looking aspects of the project; software design decisions and priorities.

The third piece of the puzzle is code. Most projects use some form of CVS that allow multiple developers to work on parts of the code separately, while providing a framework for merging changes into a common project. If bug reports show how projects interact with a larger community and mailing lists how they make decisions, CVS provides a lens into what projects actually do.

Many studies offer snapshots of a community. As such there is opportunity to misinterpret the data by ignoring the evolution of the community. For instance if a developer is highly active within a project for a small window of time and then quits, their importance may be overvalued unless time is factored in. To this end we will seek to conduct a longitudinal study of artifacts with careful attention to timeframes and the context of contributions.

Our analysis will be contributor focused. Each artifact mentioned above provides a measure for the involvement and interaction of individual contributors. We will use a very broad definition of the term contributor; anyone that interacts with the project via the artifacts studied. This is different than other studies seeking to track the activities of developers. We choose this definition because we are studying the community interactions and non-developers play an important role in FLOSS. The data from the artifact analysis will become the basis for our social network analysis.

5.4 Social Network Analysis

While each artifact can provide an interesting view into how a project works, we need an analysis framework. We plan to use social network analysis techniques to develop a better understanding of the structure of the community. By comparing the

connectedness, centrality and clustering we hope to find characteristic structures for each project. Some work in this field has already been done [4].

Social network analysis factoring in time will be vital to fully understanding how members of the community interact with each other, sub-communities, and the larger community. Recently, research has started looking into dynamic network analysis [3]. For this work, DNA will primarily build on the main metrics from SNA, but with additional tools when changes over time are considered.

Social network analysis will be used to answer RQ1 and RQ 5. Dynamic network analysis will be used in answering RQ6.

5.5 Challenges

No research is without challenges. Our main challenge is to gain access to people and information and minimize confounding variables. Participant of project leaders is a struggle in FLOSS research, as they are busy maintaining the project and are not always available to assist. Gaining access to the needed data can be difficult. Turning over email addresses can be seen as a breach of trust, privacy is a key concern.

Factors other than affiliation may have significant impact on the structure of a project. Clustering may be due to team structure or organizational decisions, or the demographics and backgrounds of project members. We will collect information about the structure of each project and compare our results against the organizational structure.

6 Preliminary Results

Some initial work has been done. At OSS 2012 I will present a paper detailing the findings of a study of the Linux Kernel and Gcc. In this paper we studied the contributions to their Bugzilla databases and CVS logs. Mailing list data will be included in future studies. For each project we found the number of unique contributors (based on emails) and the number of times an address was included (in any form) in those contributions. The main finding was that contributors affiliated with organizations are more involved in coding than in bug discussions for the Linux Kernel (Table 3). This trend is less pronounced in the Gcc project (Table 4).

There are a myriad of possible reasons for this, but the data does show a need for further study of the impact this might have on FLOSS projects.

Table 3. Linux Kernel code commits to bug reporting ratio

Domain	Unique Contributors			Contributions per contributor		
	Code	Bug	Ratio	Code	Bugs	Ratio
fujitsu.com	478	9	53.111	18.866	1.333	14.153
google.com	228	5	45.6	18.741	1.400	13.386
sgi.com	203	7	29	42.291	12.857	3.289
kernel.org	367	13	28.231	41.507	70.692	0.587
amd.com	130	9	14.444	35.400	4.111	8.611
infradead.org	83	6	13.833	140.759	42.333	3.325
oracle.com	78	6	13	184.423	5.833	31.617
redhat.com	409	32	12.781	132.770	5.438	24.415
intel.com	571	47	12.149	79.783	29.319	2.721
vmware.com	43	6	7.167	23.442	2.333	10.048
ibm.com	721	115	6.270	43.431	7.530	5.768
suse.com	132	22	6	391.856	22.500	17.416
hp.com	96	18	5.333	54.448	3.778	14.412
mit.edu	45	11	4.091	44.800	4.455	10.056
linux.org.uk	20	5	4	723.850	87.600	8.263
cam.ac.uk	21	6	3.5	52.476	2.667	19.676
mandriva.com	21	7	3	57.857	1.286	44.990
ubuntu.com	25	9	2.778	16.160	2.222	7.273
acm.org	19	7	2.714	24.053	1.714	14.033
debian.org	67	26	2.577	9.299	3.192	2.913
gnu.org	17	7	2.429	36.588	1.571	23.290
helsinki.fi	13	7	1.857	159.154	1.857	85.705
sourceforge.net	54	30	1.8	21.857	1.000	21.857
cmu.edu	11	7	1.571	10.636	1.571	6.770
ieee.org	12	8	1.5	4.583	1.875	2.444
linux.com	17	12	1.417	90.588	6.750	13.420
gentoo.org	44	36	1.222	63.068	2.722	23.170
berkeley.edu	6	5	1.2	4.333	1.400	3.095
ethz.ch	6	5	1.2	3.833	3.800	1.009
cvut.cz	8	7	1.143	22.500	3.000	7.500
hut.fi	11	10	1.1	14.545	4.800	3.030
uio.no	6	6	1	39.000	28.500	1.368
altlinux.org	6	6	1	9.500	3.333	2.850
tudelft.nl	6	7	0.857	6.167	1.714	3.598
cern.ch	5	6	0.833	2.800	1.333	2.101
osdl.org	27	112	0.241	1193.074	43.795	27.242

Table 4. GCC contributions ordered by coder/bug reporter ratio

Domain	Unique Contributors			Contributions per contributor		
	Code	Bug	Ratio	Code	Bugs	Ratio
google.com	38	6	6.333	34.184	2.333	14.652
codesourcery.com	47	8	5.875	43.766	43.875	0.998
redhat.com	150	61	2.459	52.620	15.000	3.508
suse.com	29	13	2.231	221.103	9.077	24.359
apple.com	30	15	2.000	69.300	12.733	5.443
arm.com	14	7	2.000	14.000	1.571	8.912
ibm.com	70	55	1.273	21.314	4.945	4.310
columbia.edu	5	5	1.000	17.200	1.400	12.286
inria.fr	9	11	0.818	4.444	5.273	0.843
st.com	8	10	0.800	12.500	1.900	6.579
intel.com	17	24	0.708	138.765	1.958	70.871
kpitcummins.com	6	10	0.600	1.167	2.200	0.530
gnu.org	104	174	0.598	70.356	145.414	0.484
gentoo.org	5	13	0.385	2.800	3.538	0.791
hp.com	7	19	0.368	25.571	3.947	6.479
sourceforge.net	12	35	0.343	14.500	3.743	3.874
acm.org	7	26	0.269	16.286	2.423	6.721
debian.org	10	46	0.217	19.700	10.478	1.880

Acknowledgements

I would like to thank Andy Ko for his help with the initial scripts for gathering the bug repository data. I would also like to thank Carlos Jensen for his helpful feedback and critiques in the writing of this paper.

References

1. Bezrukova, K. (2010). Violent Splits or Healthy Divides? Coping with Injustice Through Faultlines. Personnel Psychology, 63(3), 719–751. doi:10.1111/j.1744-6570.2010.01185.x
2. Bonaccorsi, A., & Rossi, C. (2003). Why Open Source software can succeed. *Research Policy, 32*(7), 1243-1258. doi:10.1016/S0048-7333(03)00051-9
3. Carley, K. M., Diesner, J., Reminga, J., & Tsvetovat, M. (2007). Toward an interoperable dynamic network analysis toolkit. Decision Support Systems, 43(4), 1324–1347. doi:10.1016/j.dss.2006.04.003
4. Crowston, Kevin, & Howison, James. "The social structure of free and open source software development" First Monday [Online], Volume 10 Number 2 (7 February 2005)
5. Gall, H., Jazayeri, M., & Krajewski, J. (2003). CVS release history data for detecting logical couplings. *Sixth International Workshop on Principles of Software Evolution, 2003. Proceedings* (pp. 13- 23). Presented at the Sixth International Workshop on

Principles of Software Evolution, 2003. Proceedings, IEEE. doi:10.1109/IWPSE.2003.1231205

6. Ghosh, R.A., & Prakash, V.V. The Orbiten Free Software Survey. First Monday, 5(7), July 2000, http://www.firstmonday.org/issues/issue5_7/ghosh
7. Hars, A., & Ou, S. (2001). Working for Free? - Motivations of Participating in Open Source Projects. *Hawaii International Conference on System Sciences* (Vol. 7, p. 7014). Los Alamitos, CA, USA: IEEE Computer Society. doi:http://doi.ieeecomputersociety.org/10.1109/HICSS.2001.927045
8. Horwitz, S. K., & Horwitz, I. B. (2007). The Effects of Team Diversity on Team Outcomes: A Meta-Analytic Review of Team Demography. Journal of Management, 33(6), 987–1015. doi:10.1177/0149206307308587
9. Huppertz, J. W., Arenson, S. J., & Evans, R. H. (1978). An Application of Equity Theory to Buyer-Seller Exchange Situations. Journal of Marketing Research, 15(2), 250–260. doi:10.2307/3151255
10. Huseman, R. C., Hatfield, J. D., & Miles, E. W. (1987). A New Perspective on Equity Theory: The Equity Sensitivity Construct. The Academy of Management Review, 12(2), 222–234. doi:10.2307/258531
11. Jehn, K. A., & Bezrukova, K. (2010). The faultline activation process and the effects of activated faultlines on coalition formation, conflict, and group outcomes. Organizational Behavior and Human Decision Processes, 112(1), 24–42. doi:10.1016/j.obhdp.2009.11.008
12. Kesner, I. F., Shapiro, D. L., & Sharma, A. (1994). Brokering Mergers: An Agency Theory Perspective on the Role of Representatives. The Academy of Management Journal, 37(3), 703–721. doi:10.2307/256707
13. Kroah-Hartman, G., Corbet, J., & McPherson, A. (2008). Linux kernel development: How fast it is going, who is doing it, what they are doing, and who is sponsoring it. *The Linux Foundation Publication* http://www. linuxfoundation. org/publications/linuxkerneldevelopment. php.
14. Krogh, G. von, Spaeth, S., & Haefliger, S. (2005). Knowledge Reuse in Open Source Software: An Exploratory Study of 15 Open Source Projects. *Hawaii International Conference on System Sciences* (Vol. 7, p. 198b). Los Alamitos, CA, USA: IEEE Computer Society. doi:http://doi.ieeecomputersociety.org/10.1109/HICSS.2005.378
15. Lakhani, K., & Wolf, R. G. (2003). Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects. *SSRN eLibrary*. Retrieved from http://papers.ssrn.com/sol3/papers.cfm?abstract_id=443040(ref 19) Lessig, L. (1999). *CODE and other laws of cyberspace*. Basic Books.
16. Lau, D. C., & Murnighan, J. K. (2005). Interactions within Groups and Subgroups: The Effects of Demographic Faultlines. The Academy of Management Journal, 48(4), 645–659. doi:10.2307/20159684
17. Lau, D. C., & Murnighan, J. K. (1998). Demographic Diversity and Faultlines: The Compositional Dynamics of Organizational Groups. The Academy of Management Review, 23(2), 325–340. doi:10.2307/259377
18. Lessig, L. (1999). *CODE and other laws of cyberspace*. Basic Books.
19. Leventhal, G. S. (1976). What Should Be Done with Equity Theory? New Approaches to the Study of Fairness in Social Relationships. Dr. Gerald S. Leventhal, Department of Psychology, Wayne State University, Detroit, Michigan 48202 (free). Retrieved from http://www.eric.ed.gov/ERICWebPortal/contentdelivery/servlet/ERICServlet?accno=ED 142463

20. Li, J., & Hambrick, D. C. (2005). Factional Groups: A New Vantage on Demographic Faultlines, Conflict, and Disintegration in Work Teams. The Academy of Management Journal, 48(5), 794–813.
21. Mockus, A., Fielding, R. T., & Herbsleb, J. D. (2002). Two case studies of open source software development: Apache and Mozilla. *ACM Trans. Softw. Eng. Methodol., 11*(3), 309–346. doi:10.1145/567793.567795.
22. Murnighan, J. K., & Brass, D. J. 1991. Intraorganizational coalitions. Research on negotiation in organizations. In M. H. Bazerman, R. J. Lewicki, & B. H. Sheppard (Eds.), Handbook of negotiation research, vol.3: 283-306. Greenwich, CT: JAI Press.
23. P. David, A. Watermann and S. Arora, "FLOSS-US: The Free/Libre/Open Source Software Survey for 2003." *Technical Report*. Standford Institute for Economic and Policy Research, Standford, USA, 2003. http://www.stanford.edu/group/floss-us/
24. Psathas, G., & Sheldon Stryker. (1965). Bargaining Behavior and Orientations in Coalition Formation. Sociometry, 28(2), 124–144. doi:10.2307/2785646
25. Rabbie, J. M., & Horwitz, M. (1988). Categories versus groups as explanatory concepts in intergroup relations. European Journal of Social Psychology, 18(2), 117–123. doi:10.1002/ejsp.2420180204
26. Sandusky, R. J., & Gasser, L. (2005). Negotiation and the coordination of information and activity in distributed software problem management. *Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work,* GROUP '05 (pp. 187–196). New York, NY, USA: ACM. doi:10.1145/1099203.1099238
27. Scacchi, W. (2004). Free and open source development practices in the game community. *IEEE Software, 21*(1), 59- 66. doi:10.1109/MS.2004.1259221
28. System Size - eLinux.org. (n.d.). Retrieved October 8, 2011, from http://elinux.org/System_Size
29. Thatcher, S. M. B., Jehn, K. A., & Zanutto, E. (2003). Cracks in Diversity Research: The Effects of Diversity Faultlines on Conflict and Performance. Group Decision and Negotiation, 12(3), 217–241. doi:10.1023/A:1023325406946
30. Webber, S. S., & Donahue, L. M. (2001). Impact of Highly and Less Job-Related Diversity on Work Group Cohesion and Performance: A Meta-Analysis. Journal of Management, 27(2), 141–162. doi:10.1177/014920630102700202
31. Williams, K Y, and C A O'Reilly. "Demography and diversity in organizations: A review of 40 years of research." Ed. Barry M Staw & L L Cummings. Research In Organizational Behavior 20.S 77 (1998) : 77-140.

Involving Older Adults in the Design and Development of Free/Open Source Software

Jennifer L. Davidson
Department of Electrical Enginnering and Computer Science,
IGERT in Aging Sciences Program, Center for Healthy Aging Research,
Oregon State Universtiy, Corvallis, Oregon, USA, davidsje@onid.orst.edu

Abstract. An age wave is upon us, with many tech-savvy older adults reaching retirement. We plan to involve older adults in a participatory design exercise with two goals in mind: 1) determine benefits and barriers of involving older adults in software design and 2) determine health metrics they are interested in tracking with software. Subsequently, to explore the engagement of retired programmers in free/open source software (FOSS), we plan to perform retrospective interviews of older adults contributing to FOSS. To gain more information, we plan to observe retired programmers during the process of creating their own FOSS community and contributing to existing communities. Our research contribution will outline best practices for involving older adults in FOSS, with the end goal of promoting an active post-working life and increasing diversity in FOSS. This research will explore benefits and barriers of involving older adults in free/open source software communities, socially and technically.

1 Introduction

The workforce is aging. We are experiencing the first wave of retiring software professionals who have skills that could be leveraged in existing free/open source software (FOSS) communities. FOSS is important to our society as it provides a method for advancing technology at a startlingly rapid rate at a relatively low cost. To remain sustainable, the FOSS community as a whole needs a constant influx of contributors. As shown in one survey, there was a peak in people joining FOSS projects around 1999, and a steady decline in recruiting ever since [7]. While some research has focused on addressing this problem by integrating FOSS projects into the undergraduate curriculum [9], we see an untapped opportunity to involve older adults in FOSS communities. As researchers, we plan to identify the benefits and barriers of older adult involvement and suggest best practices to FOSS communities. As FOSS becomes more widespread through projects like Firefox and Drupal, it will reach a more diverse audience. To create software that meets the need of its users, we argue that we need more diversity in the developer population, as FOSS communities are notoriously homogenous, dominated by 20-something, white males [7]. Determining benefits and effective ways to reduce barriers of involving older

adults will not only help the FOSS community, but also has the potential to help older adults remain active and productive into retirement. Research has shown that the brain is like a muscle: use it or lose it [19]. Researchers have conducted various interventions to maintain brain function in older adults, most of which revolve around learning a new skill such as digital photography or quilting [19]. We propose that involvement in FOSS development may be another way to maintain cognitive activity into retirement.

1.1 Aligning Motivations of FOSS Developers with Older Adults

To better understand why older adults are a good fit for FOSS communities, it is necessary to understand the motivations of FOSS contributors. In a seminal paper about motivations in FOSS, Hars and Ou [8] link work done on what motivates contributors to FOSS communities. Intrinsic motivations include internal factors such as altruism and community identification. Altruism, which they define as "the opposite of selfishness", occurs when contributors participate for the greater good. Contributors want to help the community because they believe in its value and benefits. Community identification represents the need to be a part of a community of like-minded individuals. External factors that contribute to extrinsic motivation are future rewards and personal needs. A future reward is the external factor that motivates contribution like financial compensation for work, awards, and attaining greater respect from the community. Personal needs, though self-explanatory, can be exemplified through a FOSS contributor who fixes a bug because they need the software to work for their use-case. After an empirical analysis, Hars and Ou [8] found that the most important driver for open source contributors were self-determination (housed under intrinsic motivation) and human capital (housed under future rewards). They also found that 27.8% of respondents claimed that community identity was a reason to participate. Similarly, Ye and Kishida [23] performed a case study of the motivations of the FOSS community. By using the Legitimate Peripheral Participation model, they found that learning occurs within the context of social situations, much like the community aspect identified by Hars and Ou [8].

With that knowledge, we can align the motivations of existing FOSS contributors with lives of older adults. In the literature surrounding older adults, we find that active social lives and supportive communities positively affect well-being of individuals [15]. We can link this idea to the "community identity" motivation of FOSS contributors. To take this one step further, Pew studies[1] have shown that participation in online social networks increased to 42% in 2010 (from 22% in 2009) [16]. Participation in online social networks can be easily tied to community identity. Therefore, contributing to FOSS could provide older adults another way to improve their well-being through establishing a strong community identity. While community

[1] Pew studies are performed by the Pew Research Center, which is a nonpartisan, non-profit fact tank that generates publicly available demographic research (http://pewresearch.org/about).

identity seems to be an overlapping motivation, we may see that older adults do not have high motivation of self-determination or human capital, meaning that they may not be interested in gaining new skills to make them more attractive on the job market because they are already in retirement. However, the overlap of community identity shows how older adults should be a good fit for FOSS communities.

Another justification for involving older adults in FOSS communities is the benefit of intergenerational social support for older adults. Chen and Silverstein [2] showed that intergenerational social support positively affected older adults' well-being. Because FOSS communities are dominated by younger participants [7], the intergenerational aspect of the study may prove to be beneficial for both the existing FOSS participants and older adults, if the relationships are positive. Attempting to navigate the complex FOSS joining process and being treated as a newbie might alienate and discourage many retirees. Our goal is to identify ways of helping retirees navigate the joining process more easily, as well as advertise their expertise. One way to do this is to focus on communities that have overlapping interests with older adults.

A 2004 Pew study found that 66% of older adults who use the internet look for health or medical information [6]. This may be why we have seen a slew of health and wellness applications with the proliferation of smart mobile devices. Many of these focus on what doctors want to know about a patient's health and wellness, rather than what an older adult wants to track. In the first phase of our research, we plan to address this disconnect by including older adults in the design of an open source health and wellness application. In the second phase we plan to conduct interviews of older adults participating in FOSS, as well as involving retired programmers in implementing a design from the first phase of our research. Retired programmers will also be asked to contribute to an existing FOSS project.

Our research questions are as follows:

Phase 1:

1. What are the benefits/barriers to doing participatory design with older adults?
2. What impact does critiquing software have on creativity in the development of new software?
3. What health metrics do older adults want to track to aid them in their daily activities?

Phase 2:

1. What do older adults contribute socially and technically to FOSS communities?
2. What can older adults contribute socially and technically to FOSS communities?
3. What social or technical barriers exist that inhibit older adults from contributing to the FOSS community effectively?

Our expected contributions are to:

Phase 1:

- Involve older adults in the process of software development
- Create a list of health metrics that older adults want to track

Phase 2:

- Identify processes to diversify the FOSS community by involving retired programmers
- Enable an active, post-working life for retired programmers

Figure 1 shows a sampling of ways to contribute to a FOSS community. This model is purely for context, to show that older adults will contribute both technically and non-technically. We plan to tackle this research area during two phases—first through **design**, then through **development**.

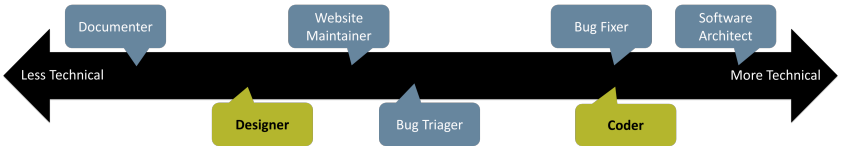

Fig 1. Free/Open Source Contribution Continuum

2 Related Work

The involvement of older adults in software and product development is the main body of research related to our topic.

2.1 Older Adults in Software/Product Development

Researchers have involved older adults in software development in the past. Abeele et al. [1] performed an ethnographic inquiry involving participatory design to develop a model based on the "passions" of the older adults for a digital game. Massimi et al. [17] have conducted participatory activities involving older adults in the evaluation of mobile phones. There has been a large effort in the United Kingdom performing participatory design with older adults. Vines et al. performed a participatory design exercise for a digital payment system called "Cheque Mates" [21]. Also in the United Kingdom, Uzor et al. conducted a participatory design study with older adults to create a fall rehabilitation tool [20]. Lorenz et al. [14] created an application for monitoring the health of older adults, but they followed a user-

centered approach that did not include participatory design sessions. We can use ideas on how to perform participatory design with older adults from this work.

Researchers including Lorenz et al. [14] and Massimi et al. [17] examine usability requirements for older adults in a deficit model. We plan to instead use the lessons from Convertino et al. [4] of positive implications of involving older works in computer supported cooperative work. We plan to use advice from Lindsay et al. [12], who gave suggestions on how to conduct participatory design with older adults. They provide a model with four steps: 1) stakeholder identification and recruitment, 2) video prompt creation, 3) exploratory meetings and 4) low fidelity prototyping sessions. We plan to implement everything except the video prompt creation because of limited resources. We are also limiting the amount of time spent in the design session to investigate if a short time limit will still result in useful designs.

To my knowledge, there has not been any research published that involves older adults contributing to FOSS. This work will be the first of its kind in studying the benefits and barriers of older adult involvement in FOSS communities.

3 Phase 1 – Design

The older adults involved in this phase will participate in high-level software design rather than low-level, FOSS-specific design. We conducted four study sessions for our first research study, and we are confident that this study is feasible. We plan to recruit 32 older adults who are not programmers. We will split participants into groups of 3 to 4 to allow for manageable group interactions. Each group will take part in a participatory design session, which is a qualitative research method used in human-computer interaction to create a low-fidelity paper prototype of a health and wellness mobile application. Our research questions for this study are listed below.

3.1 What are the benefits/barriers of doing participatory design with older adults?

We will perform open coding, a form of grounded theory, to measure anger, excitement, arguments, and agreements. From there, we will answer our research question and learn about the benefits and barriers of doing participatory design with older adults. We will compare our results with published work on participatory design with older adults [12, 20, 21, 22]. Previous work was conducted in a different cultural context, and did not involve tracking health metrics.

3.2 What impact does critiquing software have on creativity in the development of new software?

Half of the groups will critique existing health and wellness applications, followed by a participatory design session. The remaining groups will start with a

participatory design session and then proceed to critique existing applications. We will use a panel of judges and the Creativity Product Semantic Scale [13] to determine the creativity of designs. We plan to determine if critiquing existing applications helps or hinders creativity. This will help us to determine if presenting design examples to new contributors to FOSS when soliciting their help on design problems would be beneficial for the project or not.

3.3 What health metrics do older adults want to track to aid them in their daily activities?

In addition to creating design artifacts, we will administer questionnaires about health metrics. From these we will identify a list of health metrics that older adults are interested in tracking. Notably, this will be a list of what older adults want to track, rather than what doctors want them to track. To conclude Phase 1, we plan to conduct the same study with 3 additional populations (older adult programmers, undergraduate programmers and undergraduate non-programmers) to determine differences across populations.

4 Phase 2 – Development

Phase 2 involves the development of FOSS by retired programmers. Other efforts have attempted to bridge the digital divide by teaching older adults how to use technology [5, 18]. However, these studies do not focus on how to involve older adults in creating FOSS. To our knowledge, no research has been done to investigate the benefits and barriers of involving older adults in FOSS communities.

First, we plan to conduct interviews with older adults who already contribute to FOSS with the goal of learning about their joining process and how they are treated in the community. Because participants may remember the steps to contribution differently than how they actually happened, we plan to recruit older adult programmers to contribute to FOSS for the first time.

There will be two within-subject treatments. Treatment 1 involves creating a FOSS community from the ground up. Retired programmers will work on the designs created by the non-programmers, and from these, start a FOSS community with a focus on developing the health and wellness application.

Treatment 2 involves an “in-the-wild” scenario where retired programmers will be asked to contribute to an existing FOSS project. We expect there to be barriers relating to the generation gap between our subjects and current FOSS contributors. However, we expect older adults, who may be accustomed to rigid software development techniques employed in corporate environments, to provide a set of skills that could benefit the FOSS community. Uncovering these benefits and barriers will also expose shortcomings in the FOSS community that may inhibit older adults from contributing. Therefore, we plan to answer the following questions with Phase 2:

4.1 What do older adults contribute socially and technically to FOSS communities?

As part of our interviews with older adults who already contribute to a FOSS community, we plan to ask them about their history with FOSS, and their role in the FOSS community. Previous work found that people who have contributed to FOSS for a long period of time gravitate toward less technical and more "vision-oriented" roles [10]. It will be interesting to determine if the results from these interviews align with previous research results, as previous work did not take into account the age of the participant, but rather how long they have been active in the community.

4.2 What can older adults contribute socially and technically to FOSS communities?

We plan to take a positive research approach [4], focusing on the constructive aspects of involving older adults in a FOSS community. We will investigate the "gaps" in FOSS communities that the older adults may fill. For example, older adults may have better knowledge of C than other contributors because they spent most of their working career coding in C, while other contributors may be more familiar with Java or Ruby. If the FOSS project requires knowledge of C, this would be a technical asset for the older adult. We expect to find many different assets of older adults and hope to highlight them to encourage older adult participation in FOSS communities.

4.3 What social or technical barriers exist that inhibit older adults from contributing to the FOSS community?

While retrospective interviews will provide some information, we plan to learn more by observing retired programmers while they create and contribute to a FOSS project. In terms of the social implications, we plan to qualitatively examine how FOSS communities view older adults and how older adults view FOSS communities. We are interested in the perception of self as it relates to self-efficacy. We suspect that there will be interesting generation gap issues, as the median age of an open source developer is 27 [7]. Here are some potential scenarios we may witness:

Scenario 1. Older adults are seen as "managers" and either not trusted or looked down upon.
Scenario 2. Older adults are seen as "mentors" and looked to for guidance.
Scenario 3. Older adults see the FOSS community as hap-hazard, uninviting, or not worth their time.
Scenario 4. Older adults see FOSS communities as a welcome change to corporate life and are confident in their ability to contribute.
Scenario 5. Because of the anonymity of some FOSS communities, no one will realize the generation gap and they will be treated as any newcomer.

There are many variations of the above scenarios, but the ones listed are the most beneficial/detrimental in terms of continued participation. **Scenario 5** highlights an interesting point. In our studies, we do not plan to disclose the age of participants. We expect, however, that FOSS members will be able to tell that the participant is older by their vernacular and communication style.

While we have these potential scenarios, we will avoid using them as explanatory data, and instead take an exploratory approach that involves open coding. We plan to use codes that allow us to categorize our data on a continuum of negative to positive experiences, where negative experiences result in the possible discontinued participation of the older adults, and the positive experiences result in the possible continued participation of the older adults. We will cross-reference negative and positive experiences with the self-reported levels of self-efficacy to determine if high self-efficacy and positive experiences are related.

In terms of the technical aspect, we strive to determine if retired programmers have the skills necessary to participate in a FOSS project. We also foresee discovering skills that older adults have that will help to diversify the FOSS community.

During the studies, retired programmers and non-programmers will perform a memory task. Past research has attempted to determine the effect of programming on cognitive abilities in young people [3]. To our knowledge, no research has been conducted with older adults on this subject. Our goal is to start to explore if a lifetime of programming results in better performance, in terms of accuracy and time, in a spatial memory task.

5 Conclusion

Seeking ways to improve older adult involvement in free/open source software (FOSS) communities is a novel idea that has been unexplored. With this research, we hope to increase the diversity of FOSS communities while gaining useful knowledge about how to effectively design and create software with and for older adults. The participatory design sessions with end-user, older adults will give insight into their views about applications that track health metrics. Retired programmers who participate in the Phase 2 will gain experience with FOSS. Overall, our research contribution is to identify benefits and barriers of older adults contributing to FOSS, while creating a prototype of a health and wellness application by and for older adults.

6 Acknowledgements

Thank you to my advisor, Carlos Jensen, to the reviewers, and to my peers in the Human-Computer Interaction group at Oregon State University for their guidance

and feedback throughout this process. This research is funded in part by the NSF IGERT in Aging Sciences Program, DGE No. 0965820.

7 References

[1] Abeele, V. A. V., V. Van Rompaey. 2006. Introducing human-centered research to game design: designing game concepts for and *with* senior citizens. In *CHI '06 extended abstracts on Human factors in computing systems* (CHI EA '06). ACM, New York, NY, USA, 1469-1474.

[2] Chen, X, Silverstein, M. Intergenerational Social Support and the Psychological Well-Being of Older Adults in China. Research on Aging January 2000 vol. 22 no. 1 43-65.

[3] Clements, D. H., D. F. Gullo. 1984. Effects of computer programming on young children's cognition, Journal of Educational Psychology," vol. 76, issue 6, pp. 1051-158.

[4] Convertino, G., U. Farooq, M. B. Rosson, J. M. Carroll. 2005. Old is Gold: Integrating Older Workers in CSCW. Hawaii International Conference on System Sciences. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences* (HICSS'05). ACM, New York, New York, USA, 17a.

[5] Digital Elders "EmPOWER On-line & in the African American Community!," http://digitalelder.org/, accessed June 2012.

[6] Fox, S. 2004. Pew Internet & American Life Project: Older Americans and the Internet. Available online : http://pewinternet.org/~/media//Files/Reports/2004/PIP_Seniors_Online_2004.pdf.pdf.

[7] Ghosh, R. A., R. Glott, B. Krieger, G. Robles. 2002. Free/Libre and Open Source Software: Survey and Study. International Institute of Infonomics, University of Maastricht and Berlecon Research. 1-68.

[8] Hars, A., S. Ou. "Working for free? Motivations of participating in open source projects," System Sciences, 2001. Proceedings of the 34th Annual Hawaii International Conference on , vol., no., pp. 9 pp., 3-6 Jan. 2001.

[9] Jaccheri, L., T. Osterlie. 2007. Open Source Software: A Source of Possibilities for Software Engineering Education and Empirical Software Engineering. In *Proceedings of the First International Workshop on Emerging Trends in FLOSS Research and Development* (FLOSS '07). IEEE Computer Society, Washington, DC, USA, 5-.

[10] Jensen, Chris and Scacchi, Walt. 2007. Role migration and advancement processes in OSSD projects: A comparative case study. *Proceedings of International Conference on Software Engineering.*

[11] LIFE Registry. Accessed 2011. http://health.oregonstate.edu/healthy-aging/life-registry.

[12] Lindsay, S., D. Jackson, G. Schofield, and P. Olivier. 2012 Engaging older people using participatory design, *In Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, New York: ACM, pp. 1199-1208.

[13] Lobert, B. M., D. G. Dologite. 1994. Measuring creativity of information system ideas: an exploratory investigation, *Proceedings of the Twenty-Seventh Hawaii International Conference on System Sciences*, vol.4, pp. 392-402.

[14] Lorenz, A., D. Mielke, R. Oppermann, L. Zahl. 2007. Personalized mobile health monitoring for elderly. In Proceedings of the 9th international conference on Human computer interaction with mobile devices and services (MobileHCI '07). ACM, New York, NY, USA, 297-304.

[15] Lubben, J, M. Gironda. 2003. Centrality of social ties to the health and well-being older adults. Book chapter in Social work and health care in an aging society: education, policy, practice and research. Springer Publishing Company, Inc. New York, NY. 319-325.

[16] Madden, M. Pew Internet & American Life Project: "Older Adults and Social Media". Available online: : http://pewinternet.org/Reports/2010/Older-Adults-and-Social-Media.aspx. August 2010.

[17] Massimi, M., R. M. Baecker, and M. Wu. 2007. Using participatory activities with seniors to critique, build, and evaluate mobile phones. In *Proceedings of the 9th international ACM SIGACCESS conference on Computers and accessibility* (Assets '07). ACM, New York, NY, USA, 155-162.

[18] Oasis Program, "Discover Life After 50," http://www.oasisnet.org/, accessed June 2012.

[19] Park, D. C., G. N. Bischof. 2011. Neuroplasticity, Aging, and Cognitive Function, Handbook of Psychology of Aging, pp. 109-119, Elsevier, Inc.

[20] Uzor, S., L. Baillie, D. Skelton. 2012. Senior designers: empowering seniors to design enjoyable falls rehabilitation tools, *In Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, New York: ACM, pp. 1179-1188.

[21] Vines, J, M. Blythe, P. Dunphy, V. Vlachokyriakos, I. Teece, A. Monk, and P. Olivier. 2012. Cheque mates: participatory design of digital payments with eighty somethings, *In Proceedings of the 2012 ACM annual conference on Human Factors in Computing System*, New York: ACM, pp. 1189-1198.

[22] Vines, J., M. Blythe, S. Lindsay, P. Dunphy, A. Monk, and P. Olivier. 2012. Questionable concepts: critique as resource for designing with eighty somethings. *In Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems (CHI '12).* ACM, New York, NY, USA, 1169-1178.

[23] Ye, Yunwen. Kishida, K. 2003. Toward an Understanding of the Motivation of Open Source Software Developers. *Proceedings of the 25th International Conference on Software Engineering (ICSE'03)*, IEEE, 419-429.

A Research Proposal on Software Standards and their Implementation in OSS

Erik Bergström
University of Skövde, Informatics Research Centre, P.O. Box 408,
SE-541 28 Skövde, Sweden erik.bergstrom@his.se

Abstract. The work presented in this paper targets software standards, their implementation in Open Source Software (OSS) and how this affects small and medium enterprises. Much research exists in the areas of standardisation and OSS, but I argue that there are gaps in the existing body of knowledge regarding the relation between standards and their implementation in OSS, and what inhibitors are there for SMEs regarding adoption and implementation of software standards in OSS. In this paper, I present a roadmap for tackling these questions, I outline expected results and discuss implications for stakeholders affected.

Keywords: Software standards, standard implementation, Open Source Software

1 Introduction

In our daily lives, we are subjected to standards in different ways more or less constantly, and probably for most of the time even without reflecting over it. In the evolution of the information and communication technology (ICT) area standards have played an important role [1]. Standards are important for many reasons. From an economic viewpoint, for example, lowered transactional costs [2], the economy of compatibility and interoperability [3, 4], and the economics of scale [5] are mentioned as some of the economic enablers for using standards. From a more technological viewpoint, standards are important since standards, especially with a level of openness, create diffusion of technology [6].

The specification of a standard is realised through an implementation. Over time, standards and their specifications evolve which implies that the specification will lose the tie with associated implementations in software systems. Further, depending on the need of the implementer or the need in the software, project or program where the standard is to be incorporated, either a complete, extended or partial implementation of a specific version of the standard are made.

There are many reasons why there might be a deviation between the specification and the implementation. One reason or strategy is the embrace and extend strategy, where the implementer extends the standard, adding extra (sometimes proprietary)

extensions to the standard [7, 8]. This strategy also exists with a more sinister last stage, extinguish, where the goal is to take control of the standard and eliminate competition [8]. Much harm can also be caused if not the whole standard is implemented, as in the embrace and omit strategy [9].

When implementing any given specification, there are different properties that vary. The implementation can e.g. be done in different programming languages, have different levels of documentation and released under different licenses. There are two broad categories of licenses, proprietary implementations and Open Source Software[1] (OSS) implementations. OSS is becoming increasingly important, and there are many reasons why companies try to increase the amount of OSS in their organisations, including potential cost [10-13], and avoidance of vendor lock-in [13-15]. In recent years, governments and confederations such as e.g. EU have favoured software and standards with a high level of openness when procuring [16].

Standards themselves are an enabler for interoperability, permitting the possibility for several actors to e.g. read, send or store data created in a standardised way. Different standards are provided under different licensing conditions. It has been claimed that for standards which are provided under royalty free conditions, such standards become easier to adopt and implement [17]. This may be particularly important for small or medium enterprises (SMEs), and can in turn lead to increased competition and lowered procurement costs. The problem today is that even when legislated, few public sector organizations utilise the possibilities inherited by the more open standards [16], leading to among others lock-in and interoperability problems.

This paper describes a plan for an intended doctoral thesis. The paper motivates why it is important to further investigate the relation between standards and their implementation, and what inhibitors there are for implementation and adoption of standards in a SME context, and how these can be ameliorated.

This paper is organised as follows: In the next section, a background to the area is given. The problem area is introduced and the key components are problematized in Section 3. In Section 4, tentative research questions posed, and in Section 5 an outline over the proposed methods, and the stakeholders involved in the project are defined. Finally, the paper concludes with a discussion in Section 6 targeting some of the expected outcome of the research.

2 On standards, standardisation and implementation

According to Jakobs [18], the term standard is tricky, especially when it comes to information technology. The scientific standards and standardization community has not managed to achieve well-defined and agreed-upon concepts [19], and a number

[1] In this work Open Source Software is software with a licence approved by the Open Source Initiative. A complete list of the OSI licenses can be found at: http://www.opensource.org/licenses/

of definitions for the term standard exist. Different stakeholders use different definitions, including standardization bodies such as ISO/IEC [20] and confederations such as EU [21]. The scientific community has also theorised the term and a large number of definitions and discussions on what a standard is exist, including those provided in [5, 19, 22, 23].

The term standard is not the only term in the standardization field that requires discussion. For something to become a standard, the process of standardization must take place, and there must be an actor behind this, the Standard Developing Organisation (SDO), Standard Setting Organisation (SSO) or consortia. There is a separate discussion on who is a SDO, SSO and what levels of recognition and accreditation they possess. In this work, I use the term SDO inclusive, and for more on this topic, see e.g. Best [24]. The SDO should according to some standard definitions be a "recognized body", which by some only include the International Organization for Standardization (ISO). I also include standard consortia such as e.g. World Wide Web Consortium (W3C), and Organization for the Advancement of Structured Information Standards (OASIS) and organizations such as the Internet Engineering Task Force (IETF) and Institute of Electrical and Electronics Engineers (IEEE).

Standards could be divided into de facto standards and de jure standards. De facto standards are standards that have a public acceptance or normally a dominant market position. They might not formally be a developed by a SDO, as with for example Microsoft's DOC format. De facto standards can become de jure standards (developed by a SDO), as with for example PDF that became an ISO standard in 2005. The de jure standards can be of regulatory power, but normally they are voluntary. For more on classifications of standards, see for example Vries [19].

Each standard has a life cycle with states that define the status in which the standard is at the moment. The terminology differs between different SDOs, and in W3C for example, the process from idea of a standard goes via different states of drafts until the standard is a recommendation (the W3C equivalent of a standard). The standard will continue to be a standard until it is obsoleted with a new recommendation.

If the specification of a standard is hard to interpret or contain ambiguities implementation issues can arise, causing interoperability or compatibility problems. One way of possibly avoiding some of these problems is to develop a reference implementation of the specification or the parts of the specification where doubts lie [25]. Reference implementation can be defined as an *"*[i]*mplementation whose attributes and behavior are sufficiently defined by standard(s), tested by certifiable test method(s), and traceable to standard(s). The implementation may be used for assessment of a measurement method or the assignment of test method values"* [26]. There are different reasons why reference implementations are implemented, including showing the feasibility of the official specification [27], demonstrate optional features [28], capture proper behaviour in testing [29], verify the architecture [30], and increase the quality of the standard [31, 32]. Even before the OSS movement, and the focus on Open Standards, the wish and need of an available

and free reference implementation at the time of standard were communicated, e.g. through [33].

In the context of standards and their implementation there are many stakeholders. The stakeholders in standardization, and the relation between the stakeholders has been identified by e.g. [18]. Standards and standardization can be viewed from several perspectives and with different focus. Krechmer [34] defines three stakeholder perspectives for Open Standards, that should apply for standards in general (although possibly with a limited variety of implementations to choose among for the user); the SDO, or standards creator, the implementer, e.g. a programmer needing to implement a functionality, and the user. [35] suggests an Open Standards stakeholder pyramid with the roles: provider, user, developer and legislator.

This leads us to the role of Open Standards in the standards landscape. Like in the case of the term standard, there is not a universally prevailing definition, and the subject is extensively discussed, for example by [36-38].

The literature reports that when Open Standards are used, there are a number of desirable effects on the marketplace such as reduced lock-in [39-42] and interoperability [34, 43]. A key motivation for using Open Standards is that they are fundamental for achieving interoperability between products or systems. An Open Standard also ensures that the data can be interpreted independently of the tool which generated it [16]. For example, the Open Document Format (ODF), which is an Open Standard,[2] could serve as an example. If ODF is used as a file format, the organisation or user using the format has the possibility to use any software that has implemented the standard. In other words, the user could easily migrate from one ODF processor, e.g. OpenOffice and open it in LibreOffice or any other[3] interoperable word processor. It is also important to point out that there are levels of openness in Open Standards, and for example Krechmer [34] have proposed ten rights that enable Open Standards, and West [44] argues not one single model for Open Standards exist, and that there are economic reasons why levels of openness exist.

3 Problem area

According to Jakobs [22], research in the standards field could be divided into two categories, “research about standards”, and “research for standards”. This research is focused on “research about standards”, but even if delimited to ICT standards, the field spans over several areas outside ICT including economics, social sciences, law, and political sciences. The research about standards includes research on how and

[2] ODF is standardised through both ISO/IEC 26300:2006 Open Document Format for Office Applications (OpenDocument) v1.0, and OASIS.

[3] Wikipedia maintains an impressive list of ODF software at:
http://en.wikipedia.org/wiki/OpenDocument_software

why standards emerge, how they are used, and what shapes them. Lyytinen and King [1] state that *"*[d]*espite the importance of standardization, the IS field has not pursued research on it vigorously"*, and that *"IS standardization is a growing and significant research topic that deserves attention and active theoretical work in years to come"*. Also Hanseth et al. [45] acknowledge this fact, but point out the need to mitigate the increasing complexity of IS standardisation. Some areas of the field of standardisation have been thoroughly researched. For example, vast amounts of research about standards that investigates the standardisation process, and standards from different perspectives, such as economical, that according to Jakobs [18] is the most researched aspect of standardisation.

Jakobs [22] identifies 10 standards research topics, each with a number of identified research issues needing more investigation, and some of these are related to open standards, and their implementation in OSS, such as: *"*[w]*hich characteristics constitute an 'open' standard (to settle this question for once and all?"*", *"*[h]*ow to establish a common ground to bring together the OSS community and the world of standards setting?"*, and *"*[c]*ould, and should, OS implementations be incorporated into the development process for open standards?"*. Furthermore, Jakobs [22] touches the areas of how standards should be constructed, and how regulations affect the market with the questions: *"*[w]*hich impact do an SSB's* [Standard Setting Bodies] *characteristics (e.g. IPR policy, process, level of consensus, etc) have on its success in the market?"*, and *"*[i]*s this 'free market' actually desirable, or are regulatory interventions preferable, to better enable interoperability through standardisation?"*.

From this, I draw that there are broad issues concerning standardisation because of how SDOs characterise the standards, but also how it in turn affects the implementation of the standard. There are a number of studies giving recommendations for how to better develop standards including [9, 46-49], but they generally take the perspective of the SDO rather than the perspective of the implementers or users of the standard. This might be because many SDOs neglect standard implementation issues, even if for example Gans [8] argues that standard development and implementation are intertwined.

From the different stakeholder perspectives, there are still many questions related to the relations between the specification and the implementation, and between the implementation and the software/program or project that use the implemented standard, that needs further investigation. Egyedi [9] mentions a number of causes for incompatibility, including some relating to the implementation process, such as missing details, too many options and parameters, overload of standards, interference between standards, and recommendations for reducing these causes. For reaching the results, [9] used an institutional analysis, case studies and a debate among experts. I propose to take this even further, and with more stakeholders involved to learn more about the relation between standard, specification and implementation.

To focus, and approach a research objective, emphasis will be put on some more specific aspects than standardisation, specification of a standard and the implementation of standards in general. Firstly, a focus on the standard scope to

software standards (when implemented into software systems) will be made. Secondly, a focus on the implementer and provider roles to SMEs and the user role to public sector will be made.

The focus on software standards have been selected as opposed to ICT standards in general. The ICT field can be split into Information and Communication Infrastructure (ICI), and Information Technology (IT), where IT has been defined[4] as *"the hardware and software of information collection, storage, processing, and presentation"*. In my research, I focus on software standards, which I refer to as standards covering information collection, storage, processing, and presentation. There are several reasons why a focus on software standards has been made. Egyedi and Heijnen [50] classifies software and hardware standards as a dichotomy with different characteristics. Hardware standards has less room for "on the fly" improvements, and are normally standardized after the technology development and testing. Software standards, on the other hand, are more likely to be standardized before or in parallel to the technology development [50]. Interoperability amongst software is crucial to governments, industry and the third sector, and when focusing on software standards, the number of actors increases [51]. Normally, the hardware SDOs (e.g. International Telecommunication Union (ITU) and IEEE) differ from the software SDOs (e.g. IETF and W3C), and according to Vincent and Camp [52] the hardware standards have created policies that does not take advantage of the openness in software standards. To exemplify the hardware-software dichotomy a smart-phone could serve as an example. The hardware is excluded but software standards such as e.g. TCP/IP that run on the phone are included.

In EU, SMEs account for 99 per cent of all enterprises [53], making this sector undoubtedly very important for the future economy of Europe [54]. Therefore, the work focuses on SMEs active in the software sector. In this work I use the European Commission's SME definition that refers to companies with fewer than 250 employees [55], a definition also adopted in Sweden by Statistics Sweden [56]. From a SME perspective, software standards in general, but particularly software standards with a high level of openness are described as an enabler for interoperability, but also key for entering new markets.

4 Towards Research Questions

An overarching goal for this research is to establish *how SMEs use software standards aimed for implementation in software systems.*

To address this goal from the SME perspective, investigations will consider software standards which are provided under different licensing conditions. The extent to which standards are genuinely open is, by many, considered as a key enabler for

[4] For definitions including ICT, ICI and IT, see the World Bank's ICT Glossary Guide at: http://go.worldbank.org/GCEYAAZD70

market entry. This is due to removal of cost of acquiring the specification, and that no royalties are claimed. This could lead to increased competition which in turn can lead to lowered costs [41]. This belief is further reinforced by the European Union through the European Interoperability Framework (EIF v. 1.0), where Open Standards are explicitly named as a strategy for interoperability. In the updated version (EIF v. 2.0), the term Open Standard is not used, but instead the criteria that reflect the openness of the specification is discussed [57]. In other governmental settings, such as e.g. Massachusetts [41], and Sweden [16], Open Standards are favoured for public procurement. Furthermore, public administrations should offer consistency in procurements (even if the opposite has been showed by e.g. [16]), but also a transparency in procurement documentation through e.g. Tenders Electronic Daily[5] (TED). From an IPR perspective, reasonable and non-discriminatory (RAND), and royalty free (FR) are the most common standards-related royalty approaches [44]. It is considered that RAND licenses discriminate OSS developers [17, 58], because if there are royalties it is not possible for developers to distribute their software as OSS. This is limiting the development of possible solutions, e.g. for SMEs developing for the public sector, thus making it relevant to focus on OSS solutions.

Based on the overarching research objective, three interrelated RQs have been specified as follows:

RQ1: *What characterizes the relation between software standards and their implementations in OSS, and how can SMEs leverage from OSS in which software standards are implemented?*

In addition, two related sub-questions have also been established:

RQ2: *What inhibitors are there for SMEs regarding implementation of software standards in OSS, and how can these be ameliorated?*

RQ3: *What inhibitors are there for SMEs regarding adoption of software standards in OSS, and how can these be ameliorated?*

5 Research Approach

The work leading to a doctoral thesis will be conducted in context of the research project ORIOS (Open source software Reference Implementations for Open Standards) that run until 2015, providing access to data necessary for addressing the RQs.

[5] TED is a procurement portal dedicated to European public procurement.

A set of research methods will be employed as part of this work. A recently started interpretive case study as described by Walsham [59, 60] supplemented by a literature review and a field study is being carried out. The research initially targets RQ1 to gain understanding of this broad area, and aims at including all the stakeholder roles.

The collection of field data in the case study is carried out in several ways. Field notes, workshops and interviews form the foundation. As pointed out earlier, the public sector with its procurement regulations is an extensive user of software standards and OSS. Therefore, this work specifically focuses on the public sector and the case studies will consequently be conducted in various contexts in this sector. My sampling include organisations that have established contacts with the Swedish Legal, Financial and Administrative Services Agency (swe. Kammarkollegiet), providing procurement documentation. Procurement documentation will also be harvested from the TED database which is publically available online. Furthermore, key informants on different organizational levels from both the public and private sector will be interviewed. These informants are part of the network within the ORIOS project, and provide us with access to for example standard implementers, procurers, SDOs and the users in the public sector.

One source of data is procurement documentation, but I will also investigate systems already implemented or under implementation. One initial system has been selected for studying, a "legal information system" (swe. Rättsinformationssystemet[6]). The system is being developed by the Swedish government in corporation with SMEs, and contains information with sources of law from the Swedish government, higher courts and other public authorities.

Furthermore, specific standards, their associated documentations and specifications will be selected and included in the case study.

Lastly, one of the participating companies in the ORIOS project is a member of SDOs, and participates in standardization, giving insight and access to this stakeholder role.

As the research progresses, RQ2 and RQ3 will be addressed in a more collaborative way as action case research. As the action case study iterates, I expect to move from interpretation towards intervention as described by Braa and Vidgen [61]. It is expected that a longitudinal literature review is performed in each iteration of the action case study.

6 Discussion

The results from this work aim at contributing to the stakeholder groups in several ways. On a general level, I expect to contribute to an increased knowledge on how the specification of standards and their level of openness are perceived by the

[6] More information on the system is available (mainly in Swedish) at: http://lagrummet.se/Om-lagrummetse/om_lagrummet_se/rattsinformationsprojektet/

stakeholders. For SDOs, I expect this to contribute to how specifications are perceived by the other stakeholders, and I will suggest recommendations that increase the quality of the specifications. Even if some SDOs do not consider the implementation a part of the standard, it is important that qualitative specifications are developed, since it otherwise can lead to incompatibility and/or standards that are not used. On a legislative level, I expect to contribute to Action 21-25 in the Digital Agenda for Europe[7], from a SME perspective. I expect to relay understanding on how standards are implemented in OSS and adopted by SMEs when included in public procurement. In the long run, this potentially leads to many benefits for the European public sector such as increased interoperability, lowered costs and avoided lock-in. From a SME perspective it leads to increased competitiveness, and potential to break in to areas traditionally dominated by large companies. The Digital Agenda for Europe is very important since it affects the whole ICT sector of Europe.

I will also contribute to the understanding of how reference implementations are perceived and used by SMEs, especially regarding reference implementations with different levels of openness. This is expected to have implications for SDOs, but also for developers. Preliminary results from the case study indicate there are different opinions on what could be considered a software standard, and where in the lifecycle of a standard the standard resides, both in procurements and in implemented systems.

I will also contribute not only to the knowledge on how stakeholders choose between different implementations, and different standards, but also on how and why different implementations are implemented in the way they are.

I also aim at deepening the understanding on why certain standards are selected for implementation and others not, and what inhibits standards to be implemented as OSS. I aim to explain under what conditions different stakeholder groups perceive and use standards when implementing software systems. This question is multifaceted and will be viewed from the different stakeholder perspectives for gaining a better understanding of the usage of software standards, and give results for the respective stakeholders.

Acknowledgements

The author would like to thank the interviewees, the companies and organizations submitting data to the initial analysis, and colleagues in the ORIOS project for their encouragement and support. The ORIOS project is financially supported by the Knowledge Foundation ("KK-stiftelsen").

[7] Digital Agenda for Europe has a list of actions regarding interoperability and standards at: http://ec.europa.eu/information_society/newsroom/cf/pillar.cfm?pillar_id=44&pillar=Interoperability%20and%20Standards

References

1. Lyytinen, K. and J.L. King, *Standard making: a critical research frontier for information systems research.* Mis Quarterly, 2006. 30(1): p. 405-411.
2. Kindleberger, C.P., *Standards as Public, Collective and Private Goods.* Kyklos, 1983. 36(3): p. 377-396.
3. David, P. and S. Greenstein, *The Economics Of Compatibility Standards: An Introduction To Recent Research.* Economics of Innovation and New Technology, 1990. 1(1): p. 3-41.
4. Economides, N., *The economics of networks.* International Journal of Industrial Organization, 1996. 14(6): p. 673-699.
5. Tassey, G., *Standardization in technology-based markets.* Research Policy, 2000. 29(4–5): p. 587-602.
6. Funk, J.L. and D.T. Methe, *Market- and committee-based mechanisms in the creation and diffusion of global industry standards: the case of mobile communication.* Research Policy, 2001. 30(4): p. 589-610.
7. Jakobs, K. *Installation of an IEEE 802.11 WLAN in a large university setting*. in *The 4th Conference on Standardization and Innovation in Information Technology*. 2005.
8. Gans, J.S., *Remedies for tying in computer applications.* International Journal of Industrial Organization, 2011. 29(5): p. 505-512.
9. Egyedi, T.M., *Standard-compliant, but incompatible?!* Computer Standards & Interfaces, 2007. 29(6): p. 605-613.
10. Fitzgerald, B. and T. Kenny. *Open Source Software the Trenches: Lessons from a Large-Scale OSS Implementation*. in *Proceedings of the International Conference on Information Systems*. 2003. Seattle, WA., USA: Association for Information Systems.
11. Varian, H.R. and C. Shapiro. *Linux Adoption in the Public Sector: An Economic Analysis*. in *Working Paper*. 2003. University of California, Berkley.
12. Fitzgerald, B. and P.J. Ågerfalk. *The Mysteries of Open Source Software: Black and White and Red All Over?* in *Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05)*. 2005. Big Island, Hawaii, USA.
13. Morgan, L. and P. Finnegan, *Deciding on Open Innovation: An Exploration of How Firms Create and Capture Value with Open Source Software*, in *Open IT-Based Innovation: Moving Towards Cooperative IT Transfer and Knowledge Diffusion* 2008, Springer Boston. p. 229-246.
14. Brink, D., L. Roos, J. Weller, and J.-P. Van Belle. *Critical Success Factors for Migrating to OSS-on-the-Desktop: Common Themes across Three South African Case Studies*. in *Proceedings of the 2nd International Conference on Open Source Systems*. 2006.
15. Dedrick, J. and J. West, *Movement ideology vs. user pragmatism in the organizational adoption of open source software*. Computerization Movements and Technology Diffusion: From Mainframes to Ubiquitous Computing, ed. M. S.Elliott and K.L. Kraemer2008, Medford, NJ., USA: Information Today, Inc.
16. Lundell, B., *e-Governance in public sector ICT-procurement: what is shaping practice in Sweden?* European Journal of ePractice, 2011. 12(6): p. 66-78.
17. Rosen, L., *Defining Open Standards*, in *The Standards Edge: Unifier or Divider?* 2009, Sheridan Books: (to appear).
18. Jakobs, K. *Information Technology Standards, Standards Setting and Standards Research*. in *Presented at: Stanhope Center's Roundtable on systematic barriers to the inclusion of a public interest voice in the design of Information and Communications Technologies*. 2003. Cotswolds.

19. Vries, H.J.D., *IT Standards Typology*, in *Advanced Topics in Information Technology Standards and Standardization Research, Volume 1* 2006, IGI Global. p. 1-26.
20. International Organization for Standardization (ISO) & International Electrotechnical Commission (IEC), *ISO/IEC guide 2: Standardization and related activities-General vocabulary*. 8 ed2004, Geneva, Switzerland.
21. European Community, *Directive 98/34/EC of the European Parliament and of the Council of 22 June 1998 laying down a procedure for the provision of information in the field of technical standards and regulations.* Official Journal of the European Communities, 1998. 41: p. 37-48.
22. Jakobs, K., *ICT Standards Research - Quo Vadis?* Homo Oeconomicus, 2006. 23(1): p. 79-107.
23. Egyedi, T.M. and K. Blind, *The Dynamics Of Standards* 2008, Cheltenham, UK: Edward Elgar Publishing.
24. Best, K.F., *Is Accreditation Important in Standards Work?*, 2009, IGI Global. p. 35-47.
25. Egyedi, T.M. and A. Dahanayake. *Difficulties implementing standards*. in *The 3rd Conference on Standardization and Innovation in Information Technology*. 2003.
26. Carnahan, L., G. Carver, M. Gray, M. Hogan, T. Hopp, J. Horlick, G. Lyon, and E. Messina, *Metrology for information technology.* StandardView, 1997. 5(3): p. 103-109.
27. Lee, H. and N. Baek, *AlexVG: An OpenVG implementation with SVG-Tiny support.* Computer Standards & Interfaces, 2009. 31(4): p. 661-668.
28. Blinov, M. and A. Patel, *An application of the Reference Model for Open Distributed Processing to electronic brokerage.* Computer Standards & Interfaces, 2003. 25(4): p. 411-425.
29. Rosenthal, L., M. Skall, M. Brady, and C. Montanez-Rivera, *Web-based conformance testing for VRML.* StandardView, 1997. 5(3): p. 110-114.
30. Patel, A., M. Blinov, and M. Bessonov, *Reference model and functional architecture for information availability.* Computer Standards & Interfaces, 1999. 21(3): p. 273-285.
31. Kelly, G., R. Barker, G. Parkin, and M. Stevens, *SCT: Building testing into standards.* Computer Standards & Interfaces, 1998. 19(2): p. 129-141.
32. Hempel, R. and D.W. Walker, *The emergence of the MPI message passing standard for parallel computing.* Computer Standards & Interfaces, 1999. 21(1): p. 51-62.
33. Wood, D.C., *European standardization policy.* StandardView, 1995. 3(3): p. 112-116.
34. Krechmer, K. *The Meaning of Open Standards*. in *HICSS '05. Proceedings of the 38th Annual Hawaii International Conference on System Sciences*. 2005.
35. Lundell, B. *Why do we need Open Standards?* in *Proceedings 17th EURAS Annual Standardisation Conference 'Standards and Innovation'*. 2012. Aachen.
36. Krechmer, K., *Open Standards Requirements*, in *Advanced Topics in Information Technology Standards and Standardization Research, Volume 1* 2006, IGI Global.
37. Tiemann, M., *An objective definition of open standards.* Computer Standards & Interfaces, 2006. 28(5): p. 495-507.
38. Cerri, D. and A. Fuggetta, *Open standards, open formats, and open source.* Journal of Systems and Software, 2007. 80(11): p. 1930-1937.
39. West, J. *What are Open Standards? Implications for Adoption, Competition and Policy*. in *Standards and Public Policy*. 2004. Chicago, IL.
40. Guijarro, L., *Interoperability frameworks and enterprise architectures in e-government initiatives in Europe and the United States.* Government Information Quarterly, 2007. 24(1): p. 89-101.

41. Shah, R., J. Kesan, and A. Kennis. *Lessons for open standard policies: a case study of the Massachusetts experience*. in *Proceedings of the 1st international conference on Theory and practice of electronic governance*. 2007. Macao, China: ACM.
42. Galán, F., A. Sampaio, L. Rodero-Merino, I. Loy, V. Gil, and L.M. Vaquero. *Service specification in cloud environments based on extensions to open standards*. in *Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE*. 2009. Dublin, Ireland: ACM.
43. Fomin, V.V., M. Kühn Pedersen, and H.J. de Vries, *Open Standards and Government Policy: Results of a Delphi Survey.* Communications of the Association for Information Systems, 2008. 22(5): p. 459-484.
44. West, J., *The Economic Realities of Open Standards: Black, White and Many Shades of Gray*, in *Standards and Public Policy*, S. Greenstein and V. Stango, Editors. 2006, Cambridge University Press.
45. Hanseth, O., E. Jacucci, M. Grisot, and M. Aanestad, *Reflexive Standardization: Side Effects and Complexity in Standard Making.* Mis Quarterly, 2006. 30(Special Issue on Standard Making): p. 563-581.
46. Sinnott, R.O. and K.J. Turner, *Applying formal methods to standard development: The open distributed processing experience.* Computer Standards & Interfaces, 1995. 17(5–6): p. 615-630.
47. Marc van, W., *Standardization and Competing Consortia: The Trade-Off Between Speed and Compatibility.* International Journal of IT Standards & Standardization Research, 2004. 2(2): p. 18-33.
48. Sherif, M.H., T.M. Egyedi, and K. Jakobs. *Standards of quality and quality of standards for telecommunications and information technologies*. in *The 4th Conference on Standardization and Innovation in Information Technology*. 2005.
49. Backhouse, J., C.W. Hsu, and L. Silva, *Circuits of Power in Creating de jure Standards: Shaping an International Information Systems Security Standard.* Mis Quarterly, 2006. 30(Special Issue on Standard Making): p. 413-438.
50. Egyedi, T.M. and P. Heijnen. *Scale of standards dynamics in JTC1*. in *The 4th Conference on Standardization and Innovation in Information Technology*. 2005.
51. Undheim, T.A. and J. Friedrich, *The Momentum of Open Standards - a Pragmatic Approach to Software Interoperability.* European Journal of ePractice, 2008. 5(1): p. 1-13.
52. Vincent, C. and J. Camp, *Looking to the Internet for models of governance.* Ethics and Information Technology, 2004. 6(3): p. 161-173.
53. European Community, *Opportunity and Responsibility – How to help more small businesses to integrate social and environmental issues into what they do.* Directorate General for Enterprise and Industry, 2007.
54. Kroes, N., *State of the Digital Union*, in *SPEECH/11/443* 2011, Vice-President of the European Commission responsible for the Digital Agenda: Brussels.
55. European Commission, *Commission Recommendation 2003/361/EC of 6 May 2003 concerning the definition of micro, small and medium-sized enterprises.* OJ L 124 of 20.5.2003, 2003.
56. Statistiska centralbyrån, *De små och medelstora företagens ekonomi.* NV19 Näringslivets struktur, 2010. NV19SM1004.
57. Mutkoski, S. *Defining Open Standards: A comparison of policy and practice*. in *7th International Conference on Standardization and Innovation in Information Technology (SIIT)*. 2011.

58. Välimäki, M. and V. Oksanen, *Patents on Compatibility Standards and Open Source – Do Patent Law Exceptions and Royalty-Free Requirements Make Sense?* SCRIPTed, 2005. 2(3): p. 397-406.
59. Walsham, G., *Doing interpretive research.* European Journal of Information Systems, 2006. 15(3): p. 320-330.
60. Walsham, G., *Interpreting Information Systems in Organizations* 1993: John Wiley & Sons, Inc.
61. Braa, K. and R. Vidgen, *Interpretation, intervention, and reduction in the organizational laboratory: a framework for in-context information system research.* Accounting, Management and Information Technologies, 1999. 9(1): p. 25-47.

Adoption of Open Source Software in Corporate Sector

Lakshmanan Ramanathan
(Supervisor: Dr Sundaresan Krishnan Iyer)
Birla Institute of Technology and Science (BITS), Pilani, India
sanlakit@gmail.com

Abstract. Open source software is increasingly being considered/ evaluated/used across major corporates. While OSS adoption is relatively high in few organizations, it is abysmally low in many sectors. This research seeks to identify factors influencing OSS adoption in IT Outsourcing organizations and the impact IT service vendors have on those organizations.

Keywords: open source software, adoption, corporate sector, outsource

1 Introduction

Noncommercial software development in 80's and early 90's was predominantly community driven with prime focus of developing high quality free software to satisfy the immediate needs. Free software was renamed as 'Open Source Software' (OSS) in 1998 in a strategy session in California to gain business acceptance. Open Source refers to software, the license for which requires that its software code be open, extensible, and freely distributable. Some of the star OS products (Linux, Apache etc) gave a run for money for its competitors. The development of open source software happens through collaborative, informal networks of professional or amateur programmers, and the networked distribution making it available to developers and end-users free of charge. Until recently, the OSS movement has been championed by individuals, mostly software engineers and developers, with little firm participation [1]. Given the power of networking amidst communities and scale of reach, a few firms entered into the open source fray and this started the formalization of how OSS business is done. Fitzgerald goes to the extent of terming this transformation phenomenon as OSS 2.0 [2].

OSS is becoming a buzzword in many a software procurement discussions. The variations in OSS adoption can be explained by characteristics of the organizational hierarchy, geographical presence, developers and architects - including their level of awareness and experience in the OSS, business verticals, external customers, and regulatory requirements. This research aims at investigating the factors that influence

the adoption of OSS in the corporate sector and come up with solutions to address the challenges.

2 Background

Over the past decade, the OSS phenomenon has had a global impact on the way organizations and individuals create, distribute, acquire and use software and software-based services. OSS has challenged the conventional wisdom of the software engineering and software business communities. After multiple annual international conferences on open source systems, a number of special research areas emerged ranging from Software engineering perspectives on OSS development [4], Studies of OSS deployment [5], Social science perspectives on OSS development [6], and External perspectives and influences on OSS.

2.1 Need for OSS adoption:

Proprietary research output from commercial players on with limited scale in closed source development methodology will hamper the growth of the community (world) in long term [7]. OSS usage will increase the innovation process and prevents re-invention of wheel [8].

2.2 Adoption of OSS

The study by Agerfalk et al. [3] revealed an ongoing shift from OSS as a community of individual developers to OSS as a community of commercial organizations, primarily small to medium-sized enterprises. Richard's study [9] reconfirms that open source programs have moved beyond the desktops of code hackers and are now in production in a growing number of corporate IS departments.

Onetti [5] discusses about the growing number of companies adopting OSS products, however the study focuses usage of very few turnkey OSS products like Linux, Apache and claim that the adoption rate is growing. The use of OSS needs to be more than just adopting Linux as the standard for operating systems [10]. A detailed study is required to understand the adoption of various other OSS products.

Gilberto [10] mentions, corporate need to look OSS as an alternative for every possible segment of software requirement and ensure that philosophy of open source is not the software, but it is the process by which software is created. Everyone wants a tried and tested OSS product which is stable. However it is the increase in the adoption level and contribution back to the OS community that drives the maturity of the OSS product. Due to collaborative nature of work and distributed development, OSS contributions are significantly tilted towards horizontal infrastructure

components where requirements are part of conventional wisdom. Fitzgerald [2] points out that collaboration has to happen more in vertical domains, where business requirements are more complex and demand more-specialized knowledge. In the same article, Fitzgerald also emphasizes the case study of an Ireland hospital which benefitted through OSS adoption open sourced a number of healthcare applications for the benefit of community. This is a bold step that might require corporates and industry leaders to come forward to socialize and collaborate in a larger way. Little or no credible research is done to address the lack of willingness and benefits of collaboration (OSS process adoption) of corporate to contribute to OSS community.

Oliver [11] advocates that a firm adhering to the private model of innovation make the transition to a more open one by choosing the respective business model and strategically placing the launch date of such open source initiative to gain investors confidence. But the survey participants of this research were mainly from one geographical area with rather homogeneous culture which may have influenced participants' level of intrinsic and extrinsic motivations. While Spinellis et al. [17] analysed organizational adoption of OSS, this is predominantly based on secondary data from companies web browsing and serving activities and may not be a true reflection of OSS adoption.

Hauge et al. [12] performed a systematic literature review and states that the research literature has been unclear about what it actually mean to adopt OSS. They have proposed a classification framework with six ways of adopting OSS for software-intensive organizations based on organizations that a) deploy OSS products, and b) use OSS in software development. The six ways include: deploying OSS products, using OSS CASE tools, integrating OSS components, participating in the development of OSS products, providing OSS products, and using OSS development practices. Claudia et al. [21] highlights possible hidden costs as well as technical and managerial issues related to each of these six adoption approaches.

3 Motivation for the Research

Role of Global IT vendor in OSS adoption:

Worldwide technology products and related services spend is estimated to be USD 1.6 trillion [13] and is growing. Technology focused IT vendors who have a huge pie in this due to IT outsourcing play an important role in influencing the adoption of OSS in the companies. Outsourcing, is about governance - when an activity is outsourced, it is performed by another organization, as opposed to in-house by the organization itself. An activity can be performed either offshore or onshore and can be performed in-house or be outsourced. Agerfalk et al. [3] coined a new term 'opensourcing', which is the use of the OSS development model as a global sourcing strategy for an organization's software development process. The study further iterates that much OSS research has focused inward on the OSS phenomenon

itself and far less has been done on the organizational implications of OSS, and particularly on company-led OSS projects in an opensourcing strategy.

In order to motivate the IT vendors to increase OSS adoption (within itself and turn with its customers), there is a need to come up with a new framework/business model that would incentivize the IT vendors. Several business models have been proposed since the acceptance of OSS as a revenue stream [8, 14, 15]. Two have been most significant – namely, value-added service-enabling pioneered by RedHat [14] and market-creating model adopted by IBM with respect to its Eclipse platform. IBM open sourced Eclipse IDE, which was worth 40 million USD [14] potential revenue generator to the OS community. This was a strategic dual licensing [14, 15] to promote the downstream products of Rational suite of applications, which were built on top of Eclipse framework. All of these business models were focused on OSS product companies. There is no study of business models on OSS practicing for IT vendors.

There is a need for an innovative business model that motivates the clients to source-in the components from OSS and source-out developed/pre-packaged components to OSS community through the IT vendors. The research focuses on improving the penetration of Open Source Software (OSS) at various levels in the corporate sector. The main objectives of the proposed work to be carried out are:

- Role of Open Source Software (OSS) approach in software outsourcing context (Globally distributed software development process)
- Establish a comprehensive framework for open source business model for corporates to improve adoption of Open Source Software (OSS)

4 Research Questions

This section compiles some of the most important studies into perspective to form the basis of the theoretical framework of the study. The research questions and hypothesis are adapted based on the theoretical framework. There have been different studies which has identified different levels at which OSS is adopted in an organization. While Agerfalk et al. [3] talks about opensourcing as a global sourcing strategy, there is no correlation made or clear about how would this improve the adoption of OSS. This research aims to fill the gap by coming up with a framework/business model that would incentivize IT vendors to embed OSS practices in their process model, which in turn would increase the adoption of OSS in corporate sector. Gaps in literature survey lead us to following research questions

RQ1: How and to what extent are IT Outsourcing organizations currently adopting OSS?

RQ2: How and in what ways does IT service vendors impact OSS adoption of its customers?

Adoption of OSS will be assessed based on the classification framework proposed by Hauge et al. [12]. The six ways include: deploying OSS products, using OSS CASE tools, integrating OSS components, participating in the development of OSS products, providing OSS products, and using OSS development practices. Factors influencing the adoption will be analyzed by extending the framework developed by Glynn et al. [18]. In addition to four macro-factors defined in the framework (external environment, organizational context, technological context and individual factors), economic and legal factors, and user perception factors are added.

H1. Organizational Factors

- H1a. The level of the organization in which the OSS software is applied is related to the degree of adoption of OSS by the organization.
- H1b. The size of the organization in which the OSS software is applied is related to the degree of adoption of OSS by the organization.
- H1c. The relevance of OSS application to the type of business is related to the degree of adoption of OSS by the organization.
- H1d. The level of intensity of IT adoption is related to the degree of adoption of OSS by the organization.
- H1e. The organizational stability is an important factor found to influence the degree of adoption of OSS by the organization.
- H1f. The outsourcing strategy of the organization is an important factor found to influence the degree of adoption of OSS by the organization.

H2. Economic and Legal Factors

- H2a. The cost savings obtained by the company as a result of OSS adoption is an important factor found to influence the degree of adoption of OSS by the organization.
- H2b. The economics of OSS operation in terms of promoting a given business model is found to influence the degree of adoption of OSS by the organization.
- H2c. The ability to reduce dependency on other firms for software provision is a factor found to influence the level of OSS adoption by an organization.
- H2d. Licensing and other legal issues are factors found to influence the level of OSS adoption by an organization.

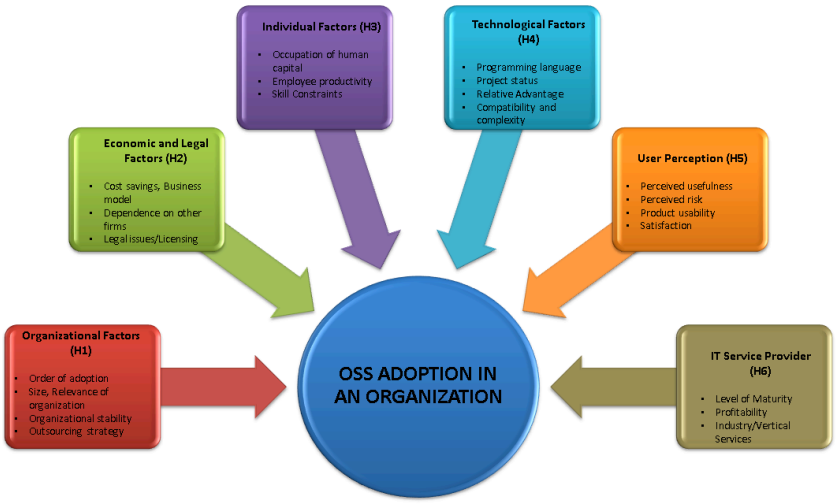

Figure 1 Factors impacting OSS Adoption in an organization

H3. Individual factors are found to be associated with the OSS adoption by an organization

H3a. The level of human capital involvement with respect to OSS usage is found to influence the level of OSS adoption by an organization.

H3b. The employee productivity level with respect to OSS usage is found to influence the level of OSS adoption by an organization.

H3c. The obstacles of skill development in terms of OSS use of the employee are found to influence the level of OSS adoption by an organization.

H3d. The uncertainty and doubt of the employees in terms of OSS use of the employee is found to influence the level of OSS adoption by an organization.

H4. Technological factors are found to be associated with the OSS adoption by an organization

H4a. The type of programming language used is found to influence the success of OSS adoption by an organization.

H4b. The status level of the OSS project adopted is found to influence the success of OSS adoption by an organization.

H4c. The trialability of the OSS used is found to influence the success of OSS adoption by an organization.

H4d. The relative advantage of the OSS used is found to influence the success of OSS adoption by an organization.

H4e. The compatibility and complexity of the OSS used is found to influence the success of OSS adoption by an organization.

H5. The user perception of the OSS has an impact on the organization adoption of open source software.

H5a. The perceived usefulness of the OSS by the user has an impact on the organization adoption of open source software.

H5b. The perceived ease of use of the OSS by the user has an impact on the organization adoption of open source software.

H5c. The user perception of the OSS compatibility an impact on the organization adoption of open source software.

H5d. The user perception of the OSS risks has an impact on the organization adoption of open source software.

H5e. The user satisfaction with the OSS has an impact on the organization adoption of open source software.

H5f. The user perception of OSS accessibility has an impact on the organization adoption of open source software.

H5g. The user perception of OSS product usability has an impact on the organization adoption of open source software.

While the first six hypotheses (H1-H5) focus on influencers with-in the organization, H6 focuses on influence of outsourced IT service providers

H6. IT Service providers are found to be associated with the OSS adoption by an organization

H6a. The level of maturity of OSS adoption in IT services provider has an impact on its clients' organization adoption of open source software.

H6b. Profitability of existing services lines of IT services provider has an impact on its ability to provide OSS services for its clients.

H6c. Industry/Vertical serviced by IT services provider has an impact on its ability to provide OSS services for its clients.

5 Research Design

Research purpose adopted in this research is explanatory. This method of research involves the clarification and determination of the existing relationship between variables proposed.

Why Explanatory is preferred for this research?

- Exploratory research type plays a role in uncovering new knowledge. The aim of this research is to find out open source software (OSS) adoption in corporate sector.
- The research questions which are often answered through descriptive research are descriptive in nature and are usually answered by using a review of literature. This research while using review of literature to identify gaps in literature and form the conceptual framework looks for empirical proof of research (*we not only want to know if OSS adoption has occurred, but also what to identify at what level*).
- The explanatory research purpose adopted helps in arriving at a solution to the proposed problem by trying to obtain an explanation for the complex web of interrelated variables which are identified and followed directly from a central research question or a hypothesis (the various facets which have an impact on the type of research and development.

This research adopts the "onion" research process promoted by Saunders et al. [19].

Table 1 Research Onion model (Adopted from Saunders[19])

Layer	**Approaches**	**Adopted for this Research**
Research philosophy	Positivism, Interpretivism, Realism, Pragmatism	Realism/Interpretivism
Research approaches	Deductive, Inductive	Deductive
Research strategies	Experiment, Survey, Case study, Grounded theory, Ethnography, Action research	Exploratory phase: Case study Testing phase: Survey, Ethnography, Action research
Time horizons	Cross Sectional, Longitudinal	Cross Sectional
Data collection methods	Sampling, Secondary data, Observation, Interviews, Questionnaires	Mixed (Quantitative and Qualitative)

- Research philosophy - This research philosophy is focused from a realistic/interpretive point of view as the research is conducted based on inputs from social actors who may be subjective. The research is value bound [19] and the researcher is part of what is being researched and is biased by world views, cultural experiences and upbringing.
- Research approaches - The research approach used here involves a top down approach (Deductive) as it is beyond the academic scope to arrive at conclusive theories and it is easier to collect empirical data once a set of hypotheses are formulated.

- Research strategies – Case study mode is adopted during exploratory phase as the research question can be analyzed in a natural setting. In the testing phase the main objective is to ensure that the methodology proposed helps in arriving at adoption of OSS. The research strategy which is adopted in the testing phase is a large scale survey method in ethnographic research as well as action research.
- Time horizons - Cross sectional study is adopted as it helps obtain data from a large number of participants and it answers a number of questions including who, what, when, where and why. It helps in generation of hypotheses for future research
- Data collection methods - This research uses a mixed measure (Quantitative and Qualitative) of research strategy. Objectivity, generalization as well as reliability of the chosen research strategy can be arrived at by using quantitative tools of data analysis [20]. However when looking at qualitative tools of analysis are based on analysing the available content presented in a non-numerical format, it is important that the research methodology adopted covers both the numerical and non-numerical data available. The advantage of using a mixed method helps in covering the research questions in a comprehensive manner.

Extensive literature survey was conducted on various forms of OSS in the market, their business modes, licensing implication and usage patterns. The key focus areas were

- Understanding organizational and psychological issues in OSS development
- Licensing, intellectual property and other legal issues in OSS
- Case studies of OSS deployment, migration models, success and failure
- OSS business models and strategies
- Open sourcing vs. offshoring of development
- Dynamics of OSS project communities, building and sustaining

The investigations in this research will be mainly been conducted through three different studies

S1. Case Study

- OSS Adoption in an IT outsourcing organization
- OSS Adoption in an IT service provider organization

S2. Survey - OSS Adoption amongst IT employees

S3. In-depth Interviews – Industry practioners, OSS developers

S1. Case Study - OSS Adoption in an IT outsourcing organization/IT service provider organization

The Case Study will be based on a large multinational corporation which is outsourcing its IT and a large IT service provider. Adoption will be analyzed based

on factors identified in framework defined earlier such as Organizational factors (Size, level of IT, Maturity level etc). This will be based on several interviews, conversations with project members, workshops, and notes from project meetings and company visits. Being an industry practitioner, the bias will be minimized/eliminated by using ethnographic research methods.

S2. Survey - OSS Adoption amongst IT employees

The data collection method will be simple rigid close ended questionnaires to the employees. Questionnaires allows in the collection of a large volume of information on a limited budget and in a very short time [22]. This research method adopts the use of surveys both normal as well as online. A formal protocol would be established for research process.

The survey will be aimed for employees at various levels within organization [Top Management Level (CXO, Director, Head of Department/Business Unit, AVP, VP etc), Managerial Level (Project Manager, Business Manager, System Manager, Delivery Manager etc), and Non-Managerial Level (Developer, Tester, Analyst, Lead etc.)]. It will focus on gathering following information. Link for web-based survey https://www.surveymonkey.com/s/OSSAdoption

- Quantitative
    - Client Profile(Demographics)
    - OSS adoption penetration across service lines
    - Degree of Adoption
    - Individual factors
    - User perception
    - Impact on Business Value
    - Dependence on software vendors
- Qualitative
    - Experience with OSS in their organization (Name of product, Specific instance, Adoption Methodology, Issues faced, Success stories etc.)
    - Expectation on Value proposition and Business model from IT services vendors for providing "Open Source Support services"

The data analysis will be carried out involves a qualitative and quantitative research analysis

- Quantitative analysis: The data would be analyzed using descriptive statistical measures. Descriptive statistics includes the use of data ranges and means. This also helps in the provision of simple summaries of the immediate data available.

S3. Case Study - In-depth Interviews – Industry practioners, OSS developers

Semi-structured interviews would be conducted to selected participants across OSS spectrum (Managers, Developers, Top Management, OSS legal experts etc.)

and the information obtained from the different respondents would be coded into text format and then analyzed to identify the commonality between the views

Post data analysis, a new business model is expected to be proposed that would motivate the organizations to source-in lot more components/productions/solutions from OSS and source-out developed/pre-packaged components to OSS community there by improving the overall adoptions at various levels.

References

1. Andrea F., Marco S. G., Alessandra L., The Penguin Has Entered the Building: The Commercialization of Open Source Software Products. *Organization Science*, 2008, No.2. 19:292-305,
2. Fitzgerald B., The Transformation of Open Source Software, *MIS Quarterly*. 2006, No.3. 30: 587-598.
3. Agerfalk P. J., Fitzgerald B., Outsourcing to an Unknown Workforce: Exploring Opensourcing as a Global Sourcing Strategy. *MIS Quarterly*, 2008. Issue 2, 32:385-409.
4. Østerlie T., Jaccheri L., A Critical Review of Software Engineering Research on Open Source Software Development, *Proceedings of the 2nd AIS SIGSAND European Symposium on Systems Analysis and Design*, Gdansk, Poland, 2007 Jun 5, 2-8.
5. Onetti A., Capobianco F., Open Source and Business Model Innovation. The Funambol case, *First International Conference on Open Source Systems Genova*. 2005 Jul 11-15.
6. Lundestad C. V., Idealism and Commercialism – Developing free/libre and open source software in private businesses*, First International Conference on Open Source Systems, Genova*, 2005 Jul 11-15.
7. Raymond, E., *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary, Revised Edition.* O'Reilly, California, USA. 2001, 33-50.
8. Linden F., Lundell B., and Marttiin P.., Commodification of Industrial Software: A Case for Open Source. *IEEE Software (IEEE Computer Society Press)*. 2009, 26: 77-83.
9. Richard T. W., Marie-Claude B., Paul T. Y., Martina E. G., Donald W. Jr., The Business of Open Source. *Communications of the ACM.* 2008, No. 4, 51:41-46.
10. Gilberto C., Frederico F., Information policies and open source software in developing countries. *Journal of the American Society for Information Science & Technology*, 2007, 58:121-132.
11. Oliver A., *How Firms Can Profit From Being Open, Free Revealing*, Ph.D. Dissertation, University of Technology Munich, Germany, 2008.
12. Øyvind H, Claudia P. A, and Reidar C. Adoption of open source software in software-intensive organizations - A systematic literature review, Information and Software Technology, v.52 n.11, p.1133-1154, November, 2010
13. National Association of Software Companies (NASSCOMM). Nasscom Strategic Review, 2009. [cited 2010 Mar 15] Available from: http://www.nasscom.org/upload/60452/Executive_summary.pdf
14. Koenig, J., *Seven open source business strategies for competitive advantage*, IT Manager's Journal, 2004 May. [cited 2010 Mar 10]. Available from: http://management.itmanagersjournal.com/article.pl?sid=04/05/l0/2052216&tid=85&tid=4

15. Krishnamurthy, S., *An Analysis of Open Source Business Models in Perspectives on Free and Open Source Software,* J. Feller, B. Fitzgerald, S. Hissam, and K. Lakhani (Ed.), MIT Press, Cambridge, MA, USA. 2005, 279-296.
16. Berdou E.., *Managing the Bazaar: Commercialization and peripheral participation in mature, community-led Free/Open source software projects* [PhD. Thesis], London School of Economics and Political Science, UK, 2007
17. Spinellis D., Giannikas V., Organizational adoption of open source software. *Journal of Systems and Software*, Mar 2012, 85(3):666–682
18. Glynn E., Fitzgerald B., Exton C., Commercial adoption of open source software: An empirical study. *International Symposium on Empirical Software Engineering*, 2005, 225-234
19. Saunders M., Lewis P., Thornhill., *Research methods for business students*. Prentice Hall, Paris, France. 2007
20. Creswell J., *Research design: Qualitative, quantitative and mixed methods approaches* (2nd ed.). SAGE Publications, Thousand Oaks, CA, USA. 2003
21. Claudia P. A., Daniela C., Øyvind H., Reidar C., Five Facts on the Adoption of Open Source Software, IEEE Software, vol. 28, no. 2, pp. 95-99, March-April 2011
22. Creswell, J.W., *Research Design: Qualitative, Quantitative and Mixed Method Approaches*, Sage Publication , California, USA. 2003.

Open Source Software, Proprietary Software and Factors Favoring the Adoption (Research in progress)

Leila Ennajeh[1], Mokhtar Amami[2],
[1] Leila Ennajeh, Doctoral student, Ecole Supérieur du commerce, LIGUE labortaory, Mannouba University ; ennajehleila@yahoo.fr,
[2] Mokhtar Amami,
Professor MIS and eSupply Management, Business Administration, RMC, Kingston, Ontario Visiting Professor University of Mannouba, amami-m@rmc.ca

Abstract. This paper is a research in progress that aims to identify factors that favor open source software adoption comparing to proprietary ones for individuals users in an organizational context. A conceptual model of OSS adoption was derived from previous IT adoption theories and was extended to integrate organizational and external factors. To validate this model, a triangulation of research methodologies was chosen. First, an exploratory study was conducted to adapt the theoretical findings with particularities of the Tunisian users and organizations (object of this paper). The next stage will be the generalization results at a large scale by a survey method and a structural equation modeling approach.

Key words: Open Source Software OSS, proprietary software, technical factors, organizational factors, adoption model, exploratory study, Tunisian context.

1 Introduction

Open source software OSS gains growing interest from the community of software and information systems developers and users in Tunisia and over the world. Today, free software movement is more dynamic than ever. Statistics provided by Netcraft[1] in October 2012 prove the world wide diffusion of OSS. In

[1] http://news.netcraft.com/archives/2012/

fact, Apache has dominated the market of web server since 1996. Furthermore, it has the largest market share (58%) against Microsoft which has only 16.28%.

Our target throughout the current research is to provide reasons that could explain this considerable growth of the market share of OSS solutions. The first apparent one is the user's decision to adopt an OSS (Miralles et al., 2006). Thus, user's choice between free software and proprietary one is an important issue and a critical determinant of the spread of such software in the world. This study focuses on the explanation of the adoption behavior of OSS solutions by identifying relevant factors that can be involved in the adoption process.

Many Models of adoption was founded in the literature on the acceptance of Information Technologies (IT) (Venkatesh et al., 2003). However, those models focused on individual factors while neglected contextual and organizational ones. In fact, studies on technology acceptance are considered as potential users' subjective analyses that do not integrate external factors (Snook, 2005). Our research is focusing on this limit and considers that individuals are not isolated from social interactions that occur in organizational context that have the potential to influence their choice. In other words, we are trying to answer the following question:

What are the factors that promote the adoption of OSS by individuals in an organizational context?

To address this issue, a literature review about open source software was presented to identify previous studies tendencies on this area. Then, we tried to develop a theoretical model that explains OSS adoption derived from previous IT adoption model and integrates organizational and environmental factors that could overcome in the adoption process of an OSS. Empirical validation of the proposed model in the Tunisian context involves two stages. The first one is the exploratory study (qualitative approach) that aims to adapt the theoretical model to the Tunisian organizations and users specificities. The second stage, is a survey (quantitative approach) by which we want the generalization of the obtained results and the subsequent model[2] that can explain users behavior against OSS Adoption.

2 Literature Review

OSS originality comparing to commercial software is due to many issues mainly availability of source code, development mode, developer's goals, support, users…

Topics found in the available literature are focusing on Business model of firms operating in open source sectors, participant's motivations to OSS projects, OSS communities and OSS development (Fugetta, 2004; Hippel and Krogh, 2003; Brydon and Vining, 2008; Tiwari, 2010; Barahona et al., 2006; Scacchi, 2004; Scacchi et al., 2006 ; Edmund Koh, 2009; Crowston et al., 2010; Snow et al., 2011).

The most important topic related to our study is OSS adoption. Some researchers identify favorable factors (relative advantage, code source availability, cost…) and unfavorable factors (lack of support, compatibility problems, switching cost…)

[2] Because of the progress of our research, analyses are limited here to the exploratory study.

(Chau and Tam, 1997; West and Dedrick, 2005; Miralles et al., 2006; Ven et al., 2008; Edmund Koh, 2009; Gallego et al., 2008; Amberg and Moller, 2006; Brydon and Vining, 2008).

Furthermore, in some other researches, adoption models were created to identify factors influencing OSS adoption particularly by organization. We find for example, the Dynamic Acceptance Model for the Reevaluation of Technologies DART model of Amberg and Moller (2006), the dynamic model of OSS adoption and disruption of Brydon and Vining (2008); and others ones developed by Kwan and West (2006) and Gallego et al., 2008). Another recent OSS adoption model was suggested by Fitzgerald et al (2012). It provides lessons on OSS adoption and assumes that managers are facing problems related to strategic, social and organizational issues. The study identifies factors that determine OSS adoption in public institutions.

In spite of the relevance of these studies, OSS adoption phenomenon needs deeper understanding and analysis that take into account managerial, technical, ideological and socio-political issues. We want, through the current research, to extend the existing literature and to create a model that can explain reasons that determine the choice between two equivalent solutions available in the market (open source and proprietary software).

3 Theoretical foundations

One of the most important difficulties encountered in our study is to reach a dispersed and extensive literature. In fact, OSS is in the core of philosophical, ideological, societal, technological and managerial debates. Nevertheless, we tried to reassemble theories that could limit the core issues related to OSS:

- **Transaction cost theory/Peer to Peer market:** it gives lessons on economic issues of software community and activity organization. Peer to peer architecture seems more appropriate for those projects[3] (Willamson, 1975; Benkler, 2002).
- **Software engineering theory**: this theory discusses the software engineering technics in the case of OSS comparing to proprietary software (Brooks, 1995; Raymond, 1998).
- **Network externalities theory:** it underlines network externalities effect in the choice of any technology especially when the value of the product increases with the network of its users (Shapiro and Varian, 1999).
- **Innovation theory**: this theory extracts problems facing managers in an innovation adoption in an organizational context. Those factors are: human problems of managing attention, a process problem in managing new ideas into good currency, a structural problem of managing the part-whole relationship, and a strategic problem of institutional leadership (Van de Ven, 1986).

[3] For more details you can refer to authors paper "The Innovative Nature of the OSS development Model", SIIE proceedings, 2012, Djerba, Tunisia.

- **IT adoption theory:** this is one of the most explored areas in information systems literature. Many models (and factors) that explain IT adoption process and use by individuals are established (Venkatesh et al., 2003).
- **Intellectual Property theory:** this theory discusses issues related to licenses that govern the use of the software and their innovativeness comparing to intellectual property principles (Silverberg, 2006; Trust, 2006).
- **Business oriented theory:** it gives lessons about software costs. Total Cost of Ownership concept refers to all direct and indirect costs incurred during the live cycle of an asset and accordingly software (Murrain et al., 2004).
- **Institutional Theory:** it discusses the government influence on organization behavior. In our study we will examine the Tunisian government policies in OSS adoption (Carpenter et Feroz, 2001).

4 Conceptual model

Our research aims to create a model that can explain OSS adoption behavior by individuals in an organizational context. In the most IT theories, adoption behavior is the explained variable. It is defined as the observable act of use in the adoption process (Azjen and Fishbein, 1980). Previous studies in IS field showed also that intention is the direct predictor of adoption behavior (Venkatesh et al., 2003). In our model, we keep this relationship and we assume that all adoption factors influence the individual's intention which plays a mediator role in the model. We assume that the adoption process began at an individual level then organizational and environmental ones.

4.1 The individual level:

At the individual level, the adoption's intention is determined by the factors that are directly related to technical features of the software, especially because potential users face two competitive solutions in the market (proprietary and OSS). The choice will be, normally, a rational one (Boudon, 2002). This idea agrees with Miralles et al. (2006) who assumed that decision makers rely on technological attributes to evaluate OSS solutions compared with proprietary ones. Given this reasoning, contextual and social factors will not be selected in this stage of the adoption process.

Second, the backbone of our model is essentially inspired from Unified Theory of Acceptance and Usefulness of Technology UTAUT (Venkatesh et al., 2003). The strength of this model is shown in many studies because it synthesizes 20 years of researches and tests of eight basic models in this IT adoption area.

Thus, giving this reasoning, we finished by choose the two following factors: *performance expectancy* [4] and *effort expectancy*[5]: two concepts that synthesize many

[4] "*The degree to which an individual believes that using the system will help him or her to attain gains in job performance*" (Venkatesh et al., 2003, p.447)

[5] "*The degree of ease associated with the use of the system*" (Venkatesh et al., 2003, p.450)

previous constructs and assumed to be strong predictors of adoption's intention.

Furthermore, we find it more reasonable to add the TCO construct[6]. This factor is significant in this particular case of technology (OSS) that seems 'free' for many users.

4.2 Organizational and environmental levels

In our model, we attempt to integrate organizational and external factors giving that individuals cannot be separated from their organizational context and environmental one. The role of those factors is assumed to be a moderator that impacts the transformation of intention to an adoption behavior.

To identify the relevant organizational factors, we referred to Van De Ven (1986) innovation's theory that studied the innovation management in an organizational context and underlined the following key factors: leadership, group pressure, organizational structure and physiological limitations. This theory is not exploited in the literature; its embedded concepts were not measured. Nevertheless, we attempt to reproach them with other measured constructs in the existing literature.

For the environmental factors, external factors taken into account in the OSS adoption process are: *Network effects* (Shapiro and Varian, 1999), *government policies* or regulations and their impact on OSS adoption (Boyer and Robert, 2005; Gaudeul, 2004) and finally *transaction costs* (Soares, 2004) with the provider which leads to switching costs.

4.3 The conceptual model

The OSS adoption model we want to create presents a continuum between an individual evaluation of the software solution (based on technical features) and organizational and external moderators. An illustration of the proposed model is shown in the figure 4.3.1.

[6] Total cost of ownership TCO: originally initiated by Gartner Group in 1987, the TCO includes all direct and indirect costs incurred during the life cycle of an asset (Heilala, 2007; Murrain et al., 2004).

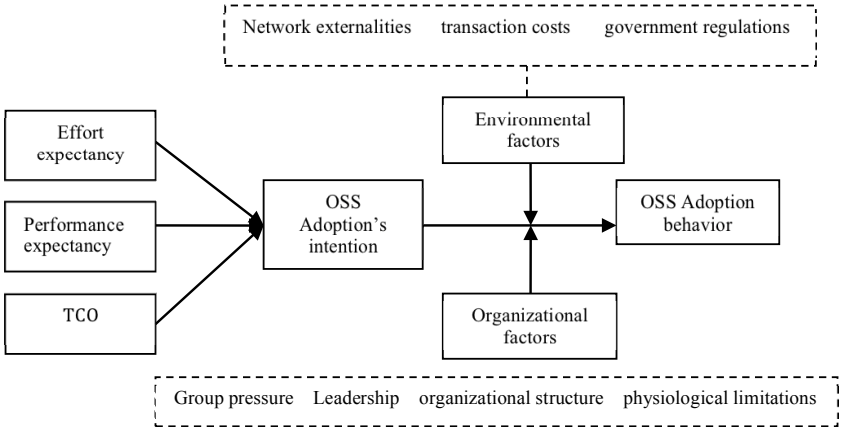

Figure 4.3.1 Conceptual Model of OSS Adoption

5 Research Methodology and Research Design

This empirical validation of this research is conducted under a triangulation of research methodologies (Johnson et Onwuegbuzie, 2004; Bryman, 1984; Jick, 1979…). In fact, we aim to validate the proposed model through two fundamental stages: qualitative research (case studies) and quantitative research (survey).

We have achieved interviews with a number of executives (24)[7] in 14 Tunisian organizations (3 public institutions, 4 public and semipublic enterprises, and 7 private ones) where activities are directly or indirectly related to information's technologies.

Collected data was analyzed manually using content analysis as suggested by a number of researchers (Wanlin, 2007; Boutigny, 2005; Burnard et al., 2008 and Schneider, 2007). Case studies were not analyzed separately, results was gathered together to obtain more generalized findings.

6 Results of the exploratory study

According to the qualitative study results, the preliminary models resulting from the theoretical reasoning has been modified and adapted to the Tunisian context. In fact, the two initially construct (*performance expectancy*, *effort expectancy)* kept their role as direct predictors of adoption's intention. However, there was an emergence of other factors: *technical compatibility* (software and hardware levels as

[7] 22 men and 2 women; the average age is 34 years, at least 5 years of experience in his/her job.

defined by Bradford and Florin, 2003); *software quality* and *system capability* (Gallego et al, 2008). Those constructs seem relevant because they are frequently mentioned by interviewees. Furthermore, the TCO is not a very important criterion to select the suitable software solution. The cost ranks second comparing to software efficiency, it gains an organizational preoccupation. As a consequence, we choose to move TCO to organizational factors.

Then, physiological limitations are also validated because all interviewees mentioned the constraints of the 'habit' (with the proprietary environment) and the 'anxiety' toward OSS use (as new tools). However, physiological limitations seem attached to individual's factors, not to the organizational ones. Thus, we decide to move them to individual determinants of OSS adoption's intention.

Furthermore, it is important to underline the emergence of a key factor in OSS adoption: individual's skills toward computer and especially toward OSS solutions. Going back to IT literature we found the '*computer self-efficacy'* concept (initially invented by Bandura (1986) and adapted to IS field by Compeau et al., (1999)). It is defined as an individual's beliefs about his or her capabilities to use Computers. This construct is shown as an important determinant not only of OSS adoption but also in the perception of OSS technical quality and the physiological limitations.

At the organizational level, many factors affect 'directly' the adoption's intention. Many interviewees said that the organizational context is very important and affects intention at the beginning of the adoption process. Thus, organizational factors do not play any role of moderator as we proposed above; their role is a direct predictor of intention. Identification of organizational factors was a complicated task because they are interdependent and similar in some cases. We kept the previous factors (*group pressure*, *organizational structure*, *leadership*[8]) because interviews showed their importance. Another important organizational factor seems relevant to include, it is the *social influence* adapted with respect to Yang (2009) perspective which includes three dimensions: *image, voluntariness* and *visibility*.

Results demonstrate also that external factors have no influence on individual intention to adopt an OSS solution. Their impact is important when we consider organizational adoption. Given that the current study is focusing on individual adoption, we choose to omit external variables from the model and limit their analysis at the exploratory level.

7 Conclusion

The model of open source software adoption as discussed above contains technical factors, organizational factors, physiological limitations and computer self-efficacy as direct predictors of intention. Adoption behavior, the endogenous

[8] There is many leadership styles identified in the leadership theory (Van Seters and Field, 1990), interviews demonstrates that the democratic style (consideration) (Stogdill, 1963) is most appropriate to OSS adoption in the Tunisian context.

variable in the model, is directly predicted by intention. Thus, intention is a central mediator between all exogenous variables and adoption behavior. This model is obtained from theoretical development and then modified according to exploratory study results achieved with Tunisian executives and IT users.

The next stage in our research will be the validation of results at a large scale with professionals in Tunisian organizations. The goal is to generalize the model and subsequently explain the OSS adoption phenomenon in Tunisia by identifying the most influencing factors involved in the adoption process. Thus, we can offer a support to the decision maker to promote the use of those software and benefit from their advantages. At the theoretical level, we want to enrich existing IT adoption models by integrating organizational factors and demonstrating their relevance in the understanding of the individual adoption behavior.

References

Ajzen, I., Fishbein, M., (1980)"Understanding attitudes and predicting social behavior", édition Prentice-Hall

Amberg, M. et Moller, S. (2006), "Adoption of open source on the enterprise desktop", OSS 2006 tOSSad workshop proceedings Como, Italy, June 10, 2006.

Barahona, J.M.G. ; Robles, G. et Herráiz, I. (2006), "Coordination Action for Libre Software Engineering for Open Development Platforms for Software and Services", Calibre project D1.3 Version 1.0.

Benkler, Y., (2002), "Coase's Penguin, or Linux and The Nature of the Firm", The Yale Law Journal, Vol. 112, 2002. Available at: http://www.ted.com/talks/yochai_benkler_on_the_new_open_source_economics.html

Boudon, R. (2002) " Théorie du choix rationnel ou individualisme méthodologique ?", Sociologie et sociétés, vol. 34, n° 1, 2002, p. 9-34. http://www.erudit.org/revue/socsoc/2002/v34/n1/009743ar.pdf

Boutigny, E., (2005), « vers un renouvellement de la démarche qualitative en sciences de gestion ? », management prospective ed. management & amp; avenir 2005/2 - n° 4, p.p 59-69.

Boyer, M. & Robert, J. (2005), 'L'économie du logiciel libre et ouvert Recommandations en vue d'une politique gouvernementale à l'égard du logiciel libre (open source software),' CIRANO ; pp. 1-44. http://www.cirano.qc.ca/pdf/publication/2005RP-05.pdf

Bradford, M., et Florin, J., (2003), "Examining the role of innovation diffusion factors on the implementation success of enterprise resource planning systems", International Journal of Accounting Information Systems 4 (2003) 205–225.

Brooks, F., (1995), "The Mythical Man Month", Essays on Software Engineering. Addison-Wesley publishing company, USA, Anniversary edition.

Brown, J.H., (2008), « developing and using computer self-efficacy scale for adults », 24th Annual Conference on Distance Teaching and Learning.

Brydon, M., et Vining, A., (2008), "Adoption, improvement, and disruption: predicting the impact of open source applications in enterprise software markets", Journal of database management, Vol.19, N. 2, Issue 22, pp. 73.

Bryman,A. (1984), "The debate about Quantitative and Qualitative Research: A Question of Method or Epistemology?", The British Journal of Sociology, Vol. 35, No. 1 (Mar., 1984), pp. 75-92

Burnard, P. ; Gill, P. ; Stewart,K. ; Treasure, E. et Chadwick, B., (2008), « Analysing and presenting qualitative data », british dental journal, V. 204, N. 8, APR 26 2008, pp. 429-432.

Carpenter, V. L., Feroz, E.H, (2001), « Institutional theory and accounting role choice: an analysis of four US states government's decision to adopt generally accepted accounting principles », Accounting, Organization and society, 26 (2001), 565-596.

Chau, P.K., Tam, K.Y. (1997), "Factors affecting the adoption of open systems: An exploratory study", MIS Quarterly, Vol. 21, N.1, pp. 1-24.

Compeau, D; Higgins, C. A.; Huff, S. (1999), "Social Cognitive Theory and Individual Reactions to Computing Technology: A Longitudinal Study", MIS Quarterly, Vol. 23, No. 2., pp. 145-158.

Crowston, K., Wei, K., Howison,J., Wiggins, A., (2010), "Free/libre open source software development: what we know and what we do not know", ACM Comput. Surv, 2010.Available at: http://flossplanet.info/system/files/CrowstonFLOSSReviewPaperPreprint.pdf

Davis, F. D. (1989), "Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology", MIS Quarterly, Vol. 13, No. 3, pp. 319-340.

Edmud Koh, K.Y, (2009), "The adoption of open source software by Singaporean campanies", a dissertation submitted for the award of masters of information technology, Queensland University of Technology, Janvier 2009.

Ennajeh L. and Amami, M., (2012), « The Innovative Nature of The OSS Development Model », Proceedings of 5th International Conference-SIIE2012-Djerba (Tunisia)-February 16th-18th, 2012, pp. 107-116.

Fitzgerald, B.; Kesan J. P., Russo, B. Shaikh, M. et. Succi, G., (2012) , "Lessons Learned from the Adoption of Open Source Software", The European Financial Review June – July 2012.

Fuggetta, A. (2004), " Open source and free software: a new model for the software development process", The European Journal for the Informatics professional, Vol. V, N. 5, pp. 22-26.

Gallego, M.D., Luna, P. et Bueno, S., (2008), "User acceptance model of open source software", Computers in Human Behavior, Vol. 24 , pp. 2199–2216.

Gaudeul, A. (2004), "Competition between open source and proprietary software: the LATEX case study", Universities of Toulouse and of Southampton, May 5, pp. 1-38.

Gonzalez-Barahona, J.M. (2000), "Impact of open source in the total cost of ownership", http://eu.conecta.it/paper/Impact_open_source_in.html

Heilala, J., Montonen, J. et Helin, K., (2007), « Selecting the right system - assembly system comparison with total cost of ownership methodology », Assembly Automation V.27, N.1, 2007, pp. 44–54.

Hippel, E.V. et Krogh, G.V. (2003), "Open source software and the private-collective innovation model: issues for organization science", Organization Science, Vol. 14, N.2, pp. 209-223. http://www2.chass.ncsu.edu/garson/pa765/path.htm

Jick, T.D, (1979), "Mixing Qualitative and Quantitative Methods: Triangulation in Action", Administrative Science Quarterly, Vol. 24, No. 4, Qualitative Methodology (Dec., 1979), pp. 602-611.

Johnson, B. et Onwuegbuzie, A.J. (2004), "Mixed Methods Research: A Research Paradigm Whose Time Has Come", Educational Researcher, Vol. 33, No. 7 (Oct., 2004), pp. 14-26

Kwan, S.K., West, J., (2006), "A Conceptual Model for Enterprise Adoption of Open Source Software" Silicon Valley Open Source Research Project, San José State University.

Limayem, M., and Hirt, S. G., (2003), "Force of Habit and Information systems usage: Theory and Initial Validation,"Journal of the Association for Information Systems (4), 2003, pp. 65-97

Miller, M.D., Kelly Rainer, R. Jr., Harper, J., (1997), "The unidimensionality, validity, and reliability of Moore and Benbasat's relative advantage and compatibility scale", The Journal of Computer Information Systems; Fall 1997; 38, 1; ABI/INFORM Global, p. 38.

Miralles, F.; Sieber, S. et Valor, S. (2006), "An exploratory framework for assessing OSS adoption", Systèmes d'Information et Management, Vol. 11, N. 1, pp. 85-112.

Murrain, M; Cowan, R; Silvers, R; Schneiderman, A.; Hickman, A et McClelland, (2004), "Shoosing and using open source software: a primer for Nonprofits", Nonprofit Open Source Initiative NOSI, pp.1-24.

Raymond, E. (1998), "The Cathedral and the Bazaar", http://www.tuxedo.org/~esr/writings/cathedral-bazaar/cathedral-bazar.html

Scacchi, W. (2004), "Understanding open source software evolution", In Software Evolution and Feedback, Madhavji NH, Lehman MM, Ramil JF, Perry D(eds). John Wiley and Sons: New York

Sccachi, W.; Feller, J.; Fitzgerald, B.; Hissam, S. et Lakhani, K. (2006), "Understanding Free/Open source software development process", Software Process Improvement Practices, Vol. 11, pp.95–105.

Schneider, D.K., (2007), « Méthodes qualitatives en sciences sociales : Petite introduction aux méthodes qualitatives » http://tecfa.unige.ch/guides/methodo/quali/

Shapiro, C. et Varian, H.R. (1999), « économie de l'information: guide stratégique de l'économie des réseaux », traduction de la première édition américaine par Fabrice Mazerolle, De Boeck université s.a,

Silverberg, B. (2006), "Intellectual property in the era of open source", http://news.com.com/2001-1_3-0.html?tag=hd_ts

Snook, J.S., (2005), « Socionormative Influence in Software Adoption and Usage », Dissertation submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of DOCTOR OF PHILOSOPHY in Computer Science & Applications.

Snow, C.C., Fieldstad, O.D., Lettl, C., et Milles, R.E., (2011), "Organizing Continuous Product Development and Commercialization: The Collaborative Community of Firms Model", Journal of Product Innovation Management, Vol. 28, Issue. 1, 2011, pp. 3–16.

Soares, M. (2004), "Reducing transaction cost in information infrastructure using FLOSS", First Monday, Vol. 9, No. 11.

Stogdill, R.M. (1963), "Manual for the Leader Behavior Description Questionnaire – Form XII". An Experimental Revision, Bureau of Business Research, Columbus, OH, USA: the Ohio State University.

Taylor S., Todd P., (1995) "Understanding information technology usage: A Test of Competing models", Information System Research, (6:4), pp. 144-176.

Thompson, R.L. Higgins C.A., and Honwell, J.M. (1991) "Personal Computing: Toward A Conceptual Model of Utilization", MIS Quarterly (15:1), 1991, pp. 124-143.

Tiwari, V., (2010), "Some observations on open source software development on software engineering perspectives", International Journal of Computer Science & Information Technology (IJCSIT), Vol 2, No 6, 2010.

Triandis, H. C. (1979), "Values, Attitudes, and Interpersonal Behavior," in Beliefs, Attitudes, and Values, University of Nebraska Press, pp. 195-259.

Trust, K.V. (2006) « Legal world and childhood dreams Open source software and intellectual property rights », http://kvtrust.blogspot.com/2006/12/open-source-software-and-intellectual.html

Van De Ven, A.H. (1986), "central problem in the management of innovation", Management science, vol. 32, No.5, pp590-607.

Van Seters, D.A. and Field, R.H.G., (1990), "The evolution of leadership theory", JOCM, 3,3, pp.29-45.

Ven, K., De Haes, S. Grembergen, W.V., et Verelst, J., (2008), "Using COBIT 4.1 to Guide the Adoption and Implementation of Open Source Software", Information Systems Control Journal, Vol. 3, pp. 1-5.

Venkatesh, V., Morris M.G., Davis G.B., Davis F.D., (2003) "User Acceptance Of Information Technology: Toward A Unified View", MIS Quarterly Vol. 27 No. 3, pp. 425-478/September 2003.

Wanlin, p., (2007), « l'analyse de contenu comme méthode d'analyse qualitative d'entretiens : une comparaison entre les traitements manuels et l'utilisation de logiciels », recherches qualitatives – hors série – numéro 3, actes du colloque bilan et prospectives de la recherche qualitative, © 2007 association pour la recherche qualitative, pp. 243-272.

West, J. et Dedrick, J. (2005) "The effect of computerization Movements Upon Organizational Adoption of Open source", Social Informatics Workshop: Extending the Contributions of Professor Rob Kling to the Analysis of Computerization Movements, March 11-12, pp. 1-30.

West, J. et Dedrick, J. (2006), "Scope and timing of development: Moderators of Organizational adoption of the Linux server platform", Journal of IT Standards & Standardization Research, Vol.4, N.2, pp.1-23.

Williamson, O. (1987), "The economic institutions of capitalism: firms, markets, relational contracting", Vol.93, No.3, pp.701-708. A Division of Macmillan, Inc. New York, Collier Macmillan Publishers, London.

Yang, H.D., Moon, Y.J., Rowley, C., (2009), « Social influence on knowledge worker's adoption of innovative information technology », Journal of Computer Information Systems, pp. 25-36.

When is a Fork a Fork? or: How I learned to Start Worrying and Loathe Semantics

Linus Nyman
Hanken School of Economics
Arkadiankatu 22, FI-00100 Helsinki, Finland
linus.nyman@hanken.fi

Abstract. The right to fork code is one of the central rights of open source software; however, to what extent does the OS community agree on what a fork is? This paper describes the struggles of a PhD student to understand his topic – code forking in open source software – and introduces work still to be completed, which seeks to answer the questions: 1) *how does the OS community define code forking?*, and 2) *how is forking perceived: when is it considered acceptable to fork a program and when is it not?*

1 Introduction

On February 18, 1930, American astronomer Clyde Tombaugh discovered a mass of rock and ice which came to be called 134340 Pluto[1]. For three quarters of a century Pluto was classified as a planet; however, around the turn of the millennium more objects of a similar size were found in the outer Solar System. This raised the need for a formal definition – when is a planet a planet? In 2006 the International Astronomical Union (IAU) presented their definition[2], and Pluto was a planet no more[3]. Posing the question "What do you think about the planets in our Solar System?" to an astronomer would, then, potentially have received a different response in early versus late 2006 – in the first instance, Pluto would have been considered part of the question; in the second, Pluto would not.

This somewhat distant tale is to give an example of a challenge I have come across in my research into code forking in open source software. Raymond, Fogel and others have provided us with means of categorization and classification of both

[1] Or just "Pluto" among friends.
[2] IAU resolution B5, "Definition of a Planet in the Solar System", available at: http://www.iau.org/static/resolutions/Resolution_GA26-5-6.pdf, last checked 21 August 2012.
[3] In IAU resolution B6 Pluto was reclassified as a "dwarf planet".

code forking as well as variations and subcategories of forking[4]. With these categories in mind I set out to research code forking, blissfully unaware of what lay ahead.

2 The plot thickens… or: Why researchers of code forking don't sleep well

Due mainly to the scarcity of previous study on code forking I began my research using an exploratory approach (see, for instance, [12]). I scourged SourceForge to see what could be learned about code forking through analyzing cases of code forking stored on the forge, hoping that what insights I gained could guide me to a clearer research problem and focus. The good news was that this approach resulted in two papers, one of which was an analysis of motivations developers stated for their forks [13]. The bad news was that that paper resulted in a slight sense of consternation and unease: from their descriptions of why they forked the programs, I got the sense that not all hackers appeared to follow the existing norms for defining a fork. Code fragmentation, code reuse, and pseudo-forking seemed commonplace, but precious few cases seemed to fit what I considered to be the traditional definition of a fork – and yet the entire data set was chosen based on the programmers themselves having stated that they had forked the code. Over time, the small sense of unease began to grow.

I had hoped that my explorative study would lead to a clear research question. Instead, the clarity it offered was of a problem with my research: how does one research a phenomenon for which there might not be a universally accepted definition? Returning to the story of Pluto, that would be like researching planets but where different researchers include and exclude different planetary bodies based on their own interpretations of what should be defined as a planet. I found myself chewing on two questions: *why are there different views on how to define a fork* and *what should I do about it?*

In pondering these questions I begin a light trawl of the field of literary theory, including its headache inducing terms semasiology[5] and onomasiology[6]. One of the potentially relevant concepts for answering the first question is that of semantic change: the phenomenon of the meaning of a word changing over time[7]. These changes, however, traditionally seem to happen over decades, if not centuries. Is the world of open source, with high-speed cable-free Wi-Fi so fast-paced that such

---

[4] A summary of the topic can be found in Appendix 1: *Code reuse, code fragmentation, and code forking in open source software.*

[5] "What does word X mean?"

[6] "How do you express X?"

[7] For instance the word "silly" was originally Old English (*selig*) for happy or fortuitous, and had by the 15th century changed to the sense of deserving of pity; then it changed to ignorant or feeble-minded, and changed again to mean foolish.

change occurs at a vastly sped up rate? Semantic change is not the only option; other possible hypotheses are that the meaning is different for different groups (academia, industry, different nationalities, different age groups, etc.), or simply that the programmers included in my data were not aware of the preexisting definitions.

While I would consider an understanding, or explanation, of why different definitions have come to exist interesting, the second question – what to do about these different definitions – was clearly the more pressing of the two. I had hoped through my studies to come to a greater understanding of code forking; however, I could find no alternative but to first try to understand what different people mean by the term. Any interviews done without first understanding an interviewee's views on this question seemed to include too great a risk of misunderstanding.

As if I wasn't losing enough sleep over definitions, my discussions with practitioners about code forking additionally made it clear to me that views on forking vary quite a lot, ranging from "avoid at all costs" to "sure, why not". However, to what extent is that dependent upon one's definition of a fork? I assume one's definition of a fork will in fact have much to do with one's feelings towards the concept; this, however, will need to be answered through interviews rather than through a doctoral student's *ex cathedra* theorizing. And so these two points – when is a fork a fork, and how forks are viewed – became my focus.

3 Testing the waters, or: I haven't been there – can I get there from here?

For those of you just joining us, two research questions have thus far been identified:

1) *How does the OS community define code forking?*
2) *How is forking perceived?* When is it considered acceptable to fork a program and when is it not? (Is the answer to this question linked to the answer to the first? How?)

In an attempt to begin unraveling this issue and finding some answers to my research questions I conducted a small batch of test interviews[8]. The interviews showed that, indeed, the existing textbook definitions of forking are not as prevalent as I had initially assumed they would be. However, interestingly, while the interviewees' definitions of forking were different in many aspects from the textbooks, their definitions were quite similar amongst the group – this even though the interviews were conducted individually, thus eliminating any potential group consensus or group pressure. While my test interview group had been exceedingly small, there was a feeling of both piqued interest as well as a glimmer of hope for my growing despair – perhaps the situation was not as hopeless as I had at first feared. It

[8] The interviews were conducted with faculty and staff of a Finnish University of Technology. While they shall remain anonymous, my sincere gratitude to them will from now on be a matter of public record.

could be that while hackers' definitions of what constitutes a fork may differ from the concise list I had gathered from academic texts, they could still be clear, concise and largely uniform. My mission, should I choose to accept it, is to find out.

4 What now? or: Coming soon(ish) to a conference near you

I will seek to answer my research questions (stated in the previous section) through a series of semi-structured interviews. While I could include more people through a survey, allowing for a clearer picture of the whole, this would first require enough knowledge of in which aspects people differ in their views of forking to be able to include these grey areas in the questionnaire. Thus, my approach will be interviews first and then, time permitting, a large(er)-scale survey.

The goal of my next paper will be to identify the various existing views of forking, both its definition as well as its acceptance. In the event that I manage to categorize interview subjects, I will also be able to offer some description of which views are more commonly held among the different groups, which could be used as a basis for future study[9].

Things that make me go "hmmm…", or: Any and all feedback welcome

Interview questions One of the challenges is how to make sure we are talking about the same thing. As peoples' opinions of forking will depend on their definition, I must make sure to get enough information about their definition that I can feel certain I understand it, and can be able to group responses correctly. Before I head out to start interviewing more people: are my questions sufficient to capture all the necessary information? I want to have good questions before I conduct my interviews, but the most effective way of improving the questions is through interviewing… Anyone interested in helping improve my interview questions, kindly see appendix 2: *Questions about code forking* for their current (incomplete) state.

Grouping The test interviews were chosen based on ease of access. It is possible that the responses of a group working and/or studying in University will have a different view on the matter than non-academics. For my main batch of interviews I would like to focus on hackers, possibly mainly those active in the OS community, while attempting to separate answers from clearly definable groups to be able to see if there are clusters of opinion. A challenge here is: what groups? Academia, industry, developers, users? What about people active in several groups? Can one make the distinctions clear enough that they are useable? Even given that I choose to focus only on programmers, what about subcategories within that group? (And which are those subcategories? Those who program primarily OS versus primarily

[9] Do industry views on forking differ from those of hackers? Does OS involvement influence views on forking?, etc.

proprietary software? Hobbyist programmers versus fulltime programmers? Those who use only OS programs vs. those who use propriety as well as OS software?)

If more is more, what is enough? How many interviews (and what selection criteria) would be necessary to be able to confidently state that one knows what group X thinks about forking? Unless one chooses a small enough group (e.g. "paid programmers at company X", or "the 30 most active contributors to project Y"), how much can one rely on the ability to extrapolate from my results?

Is this the way to go about it? Is this the time to do it? Is this a good thing to spend one's time on? Should we find out how people define forking, or would that time be better spent trying to market already existing categorizations in order to achieve one standard definition, rather than achieving a greater understanding of existing different definitions? Furthermore, if people, when left to their own devices, reinvent the definition of "fork", how long until research into its definition(s) is nothing more than a snapshot of a time gone by? Additionally, with the GitHubs of the world adapting quite different takes on the term[10], it seems that this is a particularly turbulent time for the term fork. Is it too soon to examine it – has the dust settled yet[11]?

References

1. Naur, P. and Randell, B. (1969) Software Engineering, Report on a conference sponsored by the NATO Science Committee, Garmisch, Germany, 7th to 11th October 1968, Scientific Affairs Division, NATO, Brussels, 138-155. Available at: http://www.cs.dartmouth.edu/~doug/components.txt, accessed 4 March, 2011.
2. Krueger (1992). Software Reuse. *ACM Computing Surveys*, Vol. 24, No. 2, pp. 131-183.
3. Lynex and Layzell (1998). Organizational considerations for software reuse. *Annals of Software Engineering*. 5, pp. 105-124.
4. Haefliger, von Krogh, and Spaeth (2008). *Code Reuse in Open Source Software.* Management Science, Vol. 54, No. 1, pp. 180-193.
5. Fogel, K. (2006) Producing Open Source Software. O'Reilly, Sebastopol, CA.
6. Raymond (2001) The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary. O'Reilly, Sebastopol, CA. Available at: http://catb.org/~esr/writings/homesteading/, last checked 21 November, 2011.
7. Moody (2011) The Deeper Significance of LibreOffice 3.3. ComputerWorld UK, 28 January.
8. Weber (2004) The Success of Open Source. Harvard University Press, Cambridge, Massachusetts and London, England.

[10] See, for instance, https://github.com/features/community, last checked August 24, 2012.

[11] Not to mention the even more disturbing question: Is this research destined to be outdated and/or irrelevant before the ink has even dried?

9. Lerner and Tirole (2002) Some Simple Economics of Open Source. The Journal of Industrial Economics, Vol. 50, No. 2, pp. 197-234.

10. Moody (2009). Who owns commercial open source and can forks work? *Linux Journal*, Apr 23. Available at: http://www.linuxjournal.com/content/who-owns-commercial-open-source-%E2%80%93-and-can-forks-work, accessed 24 February, 2010.

11. Meeker (2008) The Open Source Alternative: Understanding Risks and Leveraging Opportunities. Wiley, Hoboken, N.Y.

12. Hair, Money, Samouel, and Page (2007) Research methods for business. John Wiley & Sons, Chichester, West Sussex, England.

13. Nyman and Mikkonen (2011) To Fork or Not to Fork: Fork: Fork Motivations in SourceForge Projects. Proceedings of the 7th International Conference on Open Source Systems (OSS 2011), pp. 259-268, Springer.

Appendix 1. Code reuse, code fragmentation, and code forking in open source software

Code reuse, using existing software in the construction of a new software program (often in the form of reusing software components), has long been a source of both academic and practitioner interest. During the late 1960s both the use of software components and code reuse were proposed as a solution to the problem of building large, reliable software systems in a controlled, cost-effective way [1, 2]. While early data has shown that capitalizing on the promise of code reuse has proven more challenging than perhaps originally thought [3], code reuse nevertheless remains a common practice in open source programming [4].

Code forking is a somewhat more recent concept[12] and one tied even more strongly to open source software. Fogel [5] identifies two different types of forks: one group due to amicable but irreconcilable disagreements and interpersonal conflicts about the direction of the project, the second – and, as the author notes, perhaps more common group – due to both technical disagreements and interpersonal conflicts; however, it is not always possible to tell the difference between the two types. The most obvious form of forking takes place when a program splits into two versions, due to a disagreement among developers, with the original code serving as the basis for the new version of the program.

More common than code forking is *code fragmentation*, where different versions or distros (distributions) of a program emerge. Raymond [6] sees the actions of the developer community as well as the compatibility of new code to be central issues in differentiating code forking from code fragmentation. He calls different distributions of a program 'pseudo-forks', noting that they look like forks but are not perceived as such since they 'can benefit from each other's development efforts completely enough that they are neither technically nor sociologically a waste'. Moody [7] mirrors Raymond's sentiments, pointing out that code fragmentation does not traditionally lead to a split in the community and is thus considered less of a concern than a fork of the same program. These sentiments both echo a distinction made by Fogel [5]: it is not the existence of a fork which hurts a project, but rather the loss of developers and users.

Code forking has seen little discussion in the academic literature, perhaps due to negative connotations associated with the word. Forking is often perceived to be a threat, the outcome of a community failure. These kinds of connotations may lead to reluctance to use the term. However, code forking – or at the very least the option to fork – is a vital part of open source, and something which ensures its survival. A deeper analysis and understanding of code forking is important because of the central role that code forking – as well as the possibility of forking – plays in open source software. Fogel [5] considers the potential to fork a program to be 'the indispensable ingredient that binds developers together', noting that since a fork is bad for

[12] Arguably the first big cases of open source software code forking were the variants of AT&Ts UNIX in the 1970s.

everyone, the more serious the threat of a fork the more willing people are to compromise in order to avoid it. The potential to fork is also a strong element in the governance of open source programs, as no one has what Fogel calls a 'magical hold' over any project since anyone can fork any project at any time.

Both Weber [8] and Fogel [5] discuss the concept of forks as being healthy for the ecosystem in a 'survival of the fittest' sense: the best code will survive. However, they also note that while a fork may benefit the ecosystem, it is likely to be harmful for the individual project.

Another source of concern is the "hijacking" of code, which means that a commercial vendor attempts to privatize the source code [9]. However, and perhaps somewhat paradoxically, the potential to fork any open source code also ensures the possibility of survival for any project. As Moody [10] points out, open source companies and the open source community differ substantially in that companies can be bought and sold but the community cannot. If the community disapproves of the actions of an open source company, whether due to attempts to privatize the source code or other reasons related to an open source program, the open source community can simply fork the software from the last open version and continue working in whichever direction they choose.

A point worth mentioning when discussing the consequences of forking is that, even in the case of an actual split of the developer and user base, a fork can potentially offer some benefit to the program. In a situation in which a programmer would be interested in working on a program, but be reluctant to work with a specific person or team working on the same project, a fork would solve such a problem. Also, given that a fork is kept under an open source license, anything the forked version of a program develops, the original program has the right to reuse – i.e. incorporate into the original version of the program. Sometimes the forked versions either merge back together, or the fork becomes so popular among both developers and users that it becomes the new de facto main version[13].

In addition to concerns regarding the dilution of the workforce, another concern regarding program forks is their compatibility with other programs, i.e. the ability of existing programs to "port" into the new program. As Meeker [11] notes:

> In the open source world, everyone has the unfettered right to change and fork a code base, but people tend not to. Although they possess the right, they forgo it voluntarily. Lack of standardization has obvious practical problems. If there are 500 [incompatible] versions of Linux, no one will write applications for it. So there is at the same time a legal freedom to fork and a social pressure to avoid forking.

Meeker's views are mirrored by Raymond [6], who notes that while open source licenses arguably encourage both forking and pseudo-forking, it is only pseudo-forking (i.e. different distributions of a program) which is common; due to the strong social pressure against forking projects it is rarely done.

---

[13] As was the case with EGCC which forked from GCC.

Appendix 2. Questions about forking

This is a work in progress. I modified the questions during and after my test interviews, but further improvements are welcome and needed. Ideas, anyone?

Background

1. What is your OS background?
   - (Hacker, user, developer, industry links, etc.)
2. How did you become a computer programmer?
   - Since childhood? In school/Uni?

Forking - definition

3. When, in your opinion, is it appropriate to use the term "fork"?
4. How would you define a fork? (When is a fork a fork? – Code forking vs. reuse: reusing a few lines of code, reusing the entire code. Branch vs. code.)
5. Is it a fork if a functionality has been reused as part of the new program, buried/invisible inside it, or at least not prominent?
   - What if the functionality is the entire forked program vs. only part of it?
6. Does it matter (to the definition) whether I plan to merge the forked program back with the original?
   - If it matters, when is it appropriate to call it a fork? (Until merged back? Only if it never merges back? Other answer?)
7. Does the term require community involvement?
   a) Is it a fork if I take code for a personal program which I never share with anyone else?
   b) Is it a fork if one or more people take code for a project which they *do* share with others?
      - Does it matter to the definition if the program competes with the original (for users/developers)?

Forking - perceptions

8. What are the benefits of the existence of forking as a practice?
9. What are the drawbacks of the existence of forking as a practice?

10. Have you ever forked a program, or been involved in a forked program?
11. When, or under which circumstances, is it OK to fork a program?
    a) Under what kind of circumstances would you yourself consider forking a program in which you were participating?
    b) Can you think of any instances of forking which you felt it was good that a fork had happened?
12. When is it *not* OK to fork a program?
    a) Under what kind of circumstances would you consider it a bad thing that a fork happened?
    b) Can you think of any instances of forking which you felt it was a bad thing that a fork had happened?
13. When someone forks a program, are there any moral obligations towards anyone?
    - Towards the original developers? The community? Other parties?

Forking – how to go about it (in a gentlemanly manner)

14. What is the proper code of conduct or proper procedure when one wants to fork a program?

Possible grouping/control information (check: answered in 1 & 2?):

- Developer, user, or industry? (Check for combinations of the groups. Only "users" can (most likely) be in only one group.)
- How long have you been a hacker? (Age – relevant? Acceptable to ask?)
- Self-taught, University background, other?
- Paid/hobby hacker?
- Primarily/exclusively proprietary vs. OS programming?
- Opinion about proprietary software? (Uses, doesn't use? If not, why?)

Tampereen teknillinen yliopisto
PL 527
33101 Tampere

Tampere University of Technology
P.O.B. 527
FI-33101 Tampere, Finland