A STUDY ON COMMUNICATION PATTERNS IN OSS COMMUNITIES

Mehdi R. Kanso, Jackson State University

Abstract

Recently, it is noted that Open Source Software (OSS) is invading the field of software development. “OSS is developed in public in a collaborative manner that gives the user the ability to access, modify, improve, and to redistribute the software freely”[7]. OSS is a new phenomenon that is challenging the conventional commercial way of developing software projects. The fact that such types of projects gather wide range of skills from different contributors makes the project, if successfully managed and completed, competitive software that can some times dominate the market (ex. Linux, Java, MySQL, PHP). This way of development requires a lot of management, organization, and cooperation to assure that the project will result in a complete software. Various communication methods are used to connect different individuals or groups of the same project. An important facet in OSS is to control communication methods that allow successful management and productive cooperation. Various communication methods are used by the contributor to get support from other contributors, keep track of latest changes, be involved in the management process, and keep connections to facilitate monitoring, testing, and debugging.

1) INTRODUCTION

Open Source Software (or OSS) is a computer software that is developed in a collaborative and incremental manner by users that can redistribute, modify, and recompile the software. OSS provides the source code of the software to the general public with flexible or non-existent intellectual property restrictions. Eric S. Raymond demonstrates in his famous paper, “The Cathedral and the Bazaar”, the difference between the commercial way of developing the software project (the Cathedral model) and the Bazaar model OSS follows. The cathedral model is the conventional way in which the development takes place in a centralized way with predefined tasks for each developer. On the other hand, OSS is implemented in the bazaar model in which the code is developed over the internet in view of the public. OSS implementation requires an online open system that acquires different contributions from different developers. This way of collaborating in the software makes it very hard to control the development process and to direct it into a complete project that makes full use of the contributor skills, experience, motivation, and time. Both “methods of communication” and “communication patterns” directly affect the track of the software project into completion. One of the challenges in the OSS field is the ability to control communication relations between hundreds, thousands, or even millions of contributors that participate in various ways. Various developers from different backgrounds enter the project in different periods, and that is what makes the OSS project unstable and inefficient if left without high level of communication and coordination. The key to success when using the OSS is to manage and coordinate such informal communications and make use of them. Melvin E. Conway in his paper “How Do Committees Invent?” declares the famous **Conway’s Law** that states that the structure of any software product is an image of the communication structures that governed participants enrolled in developing the software project. This law points out the extreme importance of

controlling communication between developers in order to maintain a high quality of the software product. One problem that we face in developing the OSS is that these communications are informal; therefore, we should highly consider controlling the way they originate. The study of the communication structure is very critical and thus we can project Conway's Law to OSS without limiting the developer from contributing appropriately to the project. Our study on communication patterns in this paper will help understand the OSS development process more; which will directly affect the quality of the OSS project and even direct it into accomplishment or failure.

2) COMMUNICATION METHODS IN OSS

OSS is invading the field of software, thus OSS projects can contain members ranging from one developer to thousands or even millions of contributors. During the development process, the contributor develops various ways of communication to get support from other contributors, keeps track of latest changes, gets involved in the management process, and keeps connections to facilitate monitoring, testing, and debugging. Thus, the OSS community turns to be a very wide and virtual software network that encloses various ways of communication. The following tools are highly used by contributors (developers, users, administrators, Bug fixers and reporters…) to communicate with each other:

2.1) Email: It is the number one tool used in OSS projects. An email can be sent in no time to one contributor or to a group of contributors in case any communication channel is required. Some large OSS projects have built-in email systems to help communication between users and all contributors of the project.

2.2) Instant messaging: Another popular way of communication among OSS contributors is the instant messaging system and especially the IRC. Messages can be sent instantly through individual or group chat rooms to other contributors who are online.

2.3) Concurrent Versioning System (CVS): It is an open source version control system that allows contributors to keep track of changes occurring to project files. Several developers can use this system to work on the same file together at the same time. Changes to the project will be accepted on the latest version of the file only. In addition, CVS maintains different branches of a project; it can maintain developing, debugging, and testing a project through different braches that contributors can access. Log is also preserved in case any problem emerges or a review is required. An important communication technique detected in CVS is that contributors chat through comments in the body of files to demonstrate a point, demand help, or to show an error tendency.

2.4) Blog/web forum/newsgroup/discussion groups: All these communication methods are used to enhance individual or group communication. Postings in such methods are usually displayed in chronological order and permit contributors to communicate and follow up with the latest information they need for contribution.

3) COMMUNICATION PATTERNS IN OSS

Analyzing communication methods mentioned above points out patterns that contributors use in the development process of the OSS. Contributors vary in knowledge and experience; they also differ in the time they can afford to the project. Due to these differences, contributors to the OSS usually form groups that work together and help each other inside the community. Communication patterns directly affect relations between

groups and contributors inside each group. Contributors in small projects use simple patterns whereas contributors in large projects should use specified patterns to control the development process and to coordinate huge amount of contributions. Patterns analyzed in the OSS are:

3.1) Hierarchal Pattern: This pattern is usually used for management tasks that should follow a hierarchy for the information to be exchanged between different classifications of the community. The community council then passes this information to committees/teams that communicate with groups of contributors working on the OSS. For example, in case the community council starts a new project, it will pass information to the marketing team to start finding volunteers for the project. The marketing team will then contact different groups in the OSS community and search for interested, motivated, and skilled individuals.

3.2) Non-Hierarchal Pattern: This pattern is the most widely used in OSS. Different contributors and groups will develop non-hierarchal communication channels among themselves to get/give support, exchange latest news and achievements, and strengthen the process of monitoring, testing, and debugging.

3.3) Centralized Pattern: This pattern is another popular pattern in OSS. One member with high knowledge and experience can get questions from many contributors concerning the project. Usually, this person has a blog or a forum that accepts posts from different contributors. This pattern imposes on him high responsibilities since a delay or not following up with the requests will hinder the project progress.

3.4) One-to-One pattern: This pattern is mostly used in small projects where there is a limited number of contributors and no clear differentiation between members and administrators. It is processed mainly through email or instant messaging.

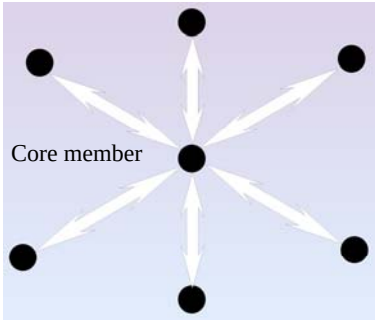

Figure 1: Centralized Pattern

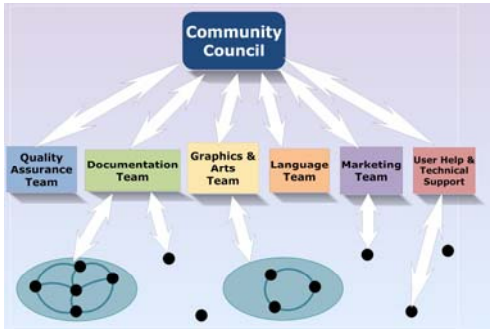

Figure 2: Hierarchal Pattern

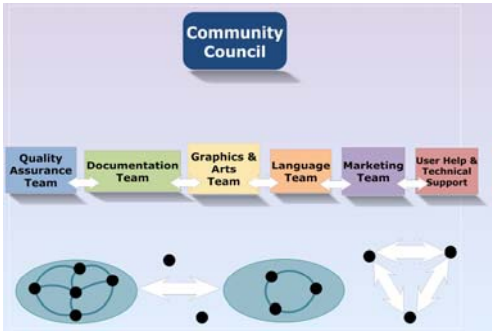

Figure 3: Non-Hierarchal Pattern

4) PROBLEMS IN OSS COMMUNICATION

Open source software projects can have a small community with less than ten contributors. On the other hand, large OSS projects tend to have thousands of contributors (ex. mySQL). Usually small community projects do not face severe communication problems except for simple disorganization that can be solved. In large virtual communities, communication is critical and, throughout the development process, contributors could face many problems in communicating with others. Problems can happen because of some gaps in the communication methods. Email, the most widely used way of communication, can lead to delay in the project progress since the email

sender should wait for the other person to be online to reply; and, in most of the times, to solve a problem more than one email and reply is needed. This gap is critical in centralized pattern communication since the central person receives many emails and any delay will affect large part of the project. In instant messaging, if a contributor is not online, he will miss information that can be mentioned in his absence. CVS, in some cases, will need manual intervention if a *conflict* arises and thus working concurrently may proliferate some problems. Contributors in the OSS are heterogeneous players and this heterogeneity can also be a problem in communication. People contributing vary in their knowledge in the software domain. A contributor having high knowledge in a certain project if communicated with another contributor with much less knowledge will cause delay to the project and will affect the project's efficiency. It was analyzed that contributors with close levels of knowledge in a certain domain will tend to form a group in which they have easy and efficient communication. Experience in the OSS domain is another difference between contributors that may affect OSS stability. Timing is critical as well and will directly affect the project's efficiency. People will vary in the time they can afford to give for contributing to the project. For example, those who give the project a big slot of their time will face delays if they build a communication channel with people who can give the OSS an hour a week. Problems mentioned can be faced during the development process of any OSS project. Such problems can have a critical effect on the released product as referred by Conway's Law.

5) COMMUNICATION NETWORK EVALUATION AND PROPOSED SOLUTION

Communication, as mentioned, is a major problem especially in large OSS projects. Such problems will directly affect motivation, productivity, and interest of the contributor. The major problem explored is the contributors' heterogeneity, the difference in knowledge, experience, and the time afforded to the project. This problem causes many delays and makes the communication channel inefficient. Large OSS projects usually follow up with contributors' activity (number of entrances to the website, number of contributions, number of errors corrected or reported, time spent online…). For example, Openoffice and openBSD keep track of every single contributor activity in the project and publish statistics online. Such statistics can be used to evaluate, solve, and minimize communication problems. Two major variables observed are knowledge/experience and time afforded to the project. Efficient, stable, and productive communication channels will require minimizing such differences.

Knowledge/ Experience (%)	Time afforded (Hours/week)	Knowledge/ Experience (%)	Time afforded (Hours/week)
15	1	45	24
10	4	35	25
20	4	25	33
20	15	20	40
35	2	10	38
85	1	50	38
75	3	65	36
90	6	70	34
80	8	70	39
60	14	80	36
35	22		

Table 1: Simulated data of an OSS project

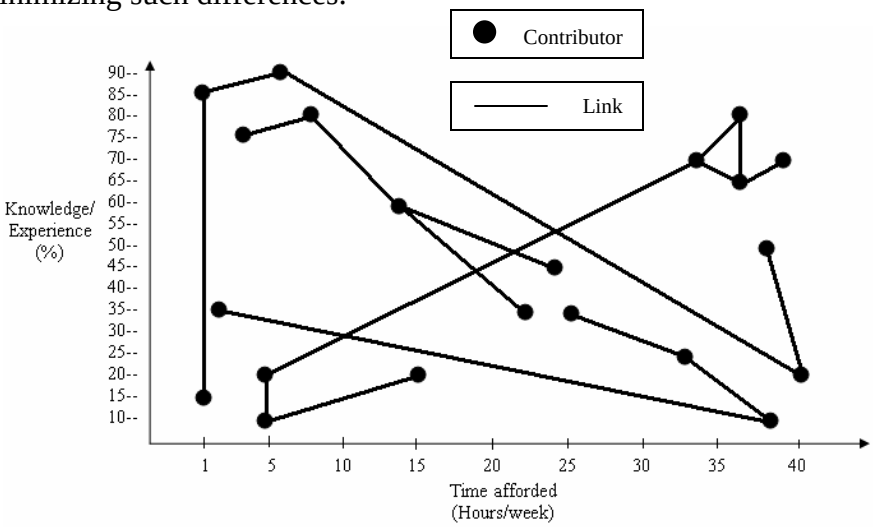

Figure 4: Graph representing communication in OSS community

Using social network and graph theory to analyze the OSS virtual community, we can construct a system to locate the position of each contributor with respect to the two variables mentioned.

The two variables “knowledge/experience” and “time afforded” can be estimated depending on the statistics or the log that the project keeps track of. For “knowledge/experience”, we will need a scaling system to estimate the contributor’s knowledge and experience depending on elements in statistics and on some surveys that describe the contributor and his history in such projects. This scaling system can be developed by a special team formed to keep track of communication channels. “Time afforded” can be estimated by using some elements in the statistics (time spent online, number of contributions per day...) and some surveys that can ask contributors to estimate the time they spent working on the project. The link between two nodes (contributors) is a communication channel that links these two contributors. To represent the virtual community, we should observe links that connect contributors while they work on the project. An example is illustrated in the figure above of heterogenic contributors that communicate through channels represented by links on the graph. The long link represents two very different contributors having communication. Such a link is costly to the project and there is high probability that it will lead to a delay. On the other hand, the short link represents an efficient and stable communication channel between two contributors of close level of knowledge, experience, and time they can afford to the project. The weight of the graph used will be the simple distance $\omega = d = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$ to represent the difference between “knowledge/experience” and “time afforded” for contributors. This weight is the cost of the communication channel between two contributors and our goal is to minimize such weight to make the communication efficient, stable, and as productive as possible. After we locate the position of each contributor with respect to the two variables specified, we need to get the combination of communication channels of which the total weight is minimized and that include all contributors. We project “Minimum Spanning tree” solution on the complete graph of contributors to get the best network in terms of efficient and stable connections. Let the graph of contributors on the two axes be denoted as G. V is the set of vertices in the system (contributors). E is the edge between two vertices and it represents a communication channel between two contributors. Prim’s algorithm is a greedy algorithm that operates much like Dijkstra’s algorithm for finding shortest paths in the graph. The total time for Prim’s algorithm will be $O(E \lg V)$. We run Prim’s algorithm on the complete graph that connects all the contributors and the result will be a tree that connects all contributors with the minimum communication cost and the most efficient.

For an existing network study, to evaluate the cost of communications, we simply sum up all the weights of the links in the network. Such sum is represented by $w(G) = \sum_{(u,v)\in G} w(u,v)$. This sum will then reflect the difference between contributors that are communicating and collaborating. It represents how much heavy communications the network includes. Compare the obtained sum with the sum of weights of the tree with minimized weights (obtained after Prim’s algorithm) to find out the position of the project in terms of efficient and stable communication.

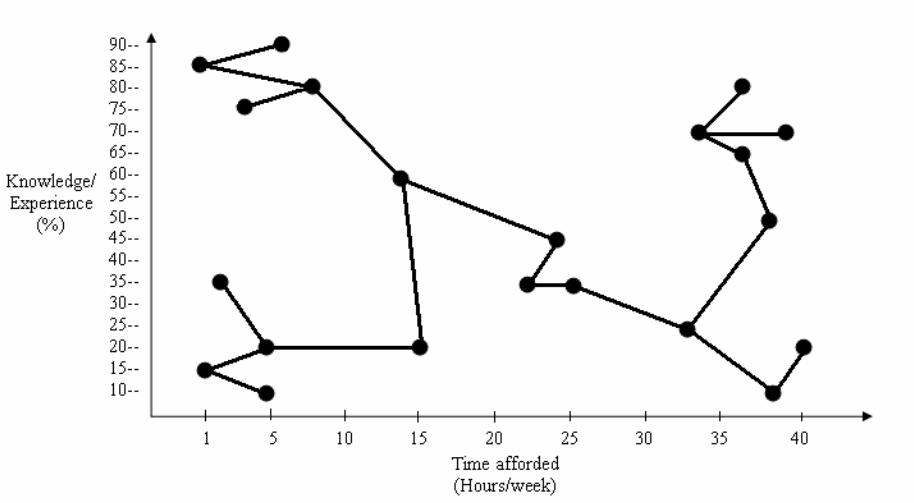

Figure 5: Graph after applying Prim's algorithm

Weight of the graph : $w(G) = \sum_{(u,v)\in G} w(u,v)$	
For the project in real life	457.44
After applying Prim's	252.31

Table 2: Prim's algorithm effect on the weight

An important point is that the community council cannot force contributors to go into the most efficient communication channel, but they can give advice and motivate them since any contribution in the OSS will be a volunteering work. It was observed that some contributors contact the community council to get advice about the suitable person or group to work with. The answer is found in the resultant tree after running Prim's algorithm.

6) CONCLUSION

OSS is growing rapidly and everyday hundreds or even more OSS projects are being initiated. Communication is becoming more critical as the number of contributors increase and as more projects start. Controlling communication in the software developing process will directly affect the structure of the software and thus its quality. Studying communication patterns and methods and finding solutions to problems occurring is required to promote OSS projects and to direct them into completion and success. Following well-defined patterns and various methods of communication made some large OSS projects (Linux, mySQL, openOffice…) invade the software field and market. OSS projects start with a small number of contributors and thus communication organization and control will be easier. As the number of contributors increase, there should be monitoring and organization for the communication patterns and methods being involved in the project. Statistics will help in following up with all the contributors and in directing them into efficient, stable, and productive communication channels. More research and study is required in the domain of OSS communication because of the growing role of OSS and its involvement in the software domain.

REFERENCES

1. Singh, P., Tan, Y., Dey, D. , ***Stability and Efficiency of Communication Networks in Open Source Software Projects,*** **University of Washington, 1-5, 2006.**

2. Wagstrom, P., Carley, K., ***Social Networks From Free/Open Source Developers Weblogs,*** **Carnegie Mellon University, 1-21, 2005.**

3. Crowston, K., Scozzi, B., ***Open Source Software Projects as Virtual Organizations: Competency Rallying for Software Development,*** **IEE proceedings Software, 41, 2002.**

4.Nakakoji, K., Yamamoto, Y., Nishinaka ,Y.,Kishida, Y.,Ye , Y., ***Evolution Patterns of Open-Source Software Systems and Communities,*** **Japan society for the promotion of science,4-10, 2002.**

5. Ahuja, M., Carley, K., ***Network structure in virtual organizations,*** **1998, http://jcmc.indiana.edu/vol3/issue4/ahuja.html.**

6. OpenOffice, ***OpenOffice.org statistics ,*** **2007, http://stats.openoffice.org/spreadsheet/index.html.**

7. Wikipedia, ***Open-Source Software,*** **2007, http://en.wikipedia.org/wiki/Open_source_software.**