An Exploratory Study on the Evolution of OSS Developer Communities

Kawin Ngamkajornwiwat, Dongsong Zhang, A. Güneş Koru, Lina Zhou & Robert Nolker
Department of Information Systems, University of Maryland Baltimore County
{kawin1, zhangd, gkoru, zhoul, rnolker1} @umbc.edu

Abstract

Software is developed in a dynamic context where team structure, requirements, and processes evolve together with the product. Although researchers have been studying the evolution of software systems since the early 70s, the study on the evolution of software development teams remains rare. Such evolutionary patterns and their impact on software quality are especially important in the context of OSS (Open Source Software) development, where a group of volunteer developers collaborate online for an extended period of time. By analyzing how social networks of developers evolve over time while building OSS products, we can gain knowledge and experience to improve the effectiveness and efficiency of resource management and distribution in future OSS projects. To this end, we studied the evolution of the developer communities using a suite of OSS products developed under the KOffice project. We found that in general, the social networks of OSS developer communities change over time in certain ways. Identifying such patterns can help OSS managers better understand the unique process of OSS development and improve their management and coordination of the projects.

1. Introduction

OSS (Open Source Software) development, in which programmers in Internet-based communities collaborate to voluntarily contribute programming code, has emerged as an alternative approach to software development. On SourceForge.com, the world's largest OSS development web site, there are historic and status statistics on over 100,000 projects with the project management portion of the web site recording over 1 million registered users' activities.

However, there is a cruel reality that the vast majority of OSS projects fail to take off and soon become abandoned [11]. The main reasons for the failure of OSS projects include the lack of developers on the project team, the inability of the project to bring together a critical mass of developers [11], and the lack of sufficient communication and collaboration between developers. For example, about 95% of OSS projects on SourceForge.com have five or fewer contributors.

OSS development is a prominent example of the community-based model [23]. The collaborative relationships between developers form a "complex network". Such a network has a mixture of randomness and structure, and is dynamic and evolves. Here software evolution refers to collective changes made to software systems in order to meet the change in requirements and/or surrounding environments [1]. This type of collaborative network is different from traditional social networks because links between nodes are not persistent. For example, in a developer social network, a developer can leave an OSS project at any time, causing a node and edges connected to that node to disappear from the network. Understanding the evolution of such networks and their influence on OSS quality can help OSS project managers to better manage and coordinate the development process.

Most previous research on OSS developers has been trying to understand the motivations, participation, and performance of developers, along with how different motivations affect developers' participation and performance in OSS projects (e.g., [21, 23]). To date, software product evolution has been studied more extensively than the process of development or team evolution. Particularly, very few studies have examined the evolution of developer communities over time. We aim to fill that void by examining this issue in a large, successful OSS project, namely KOffice, using social network analysis. Due to the lack of existing theories and previous empirical findings, this study is exploratory by nature. We are particularly interested in addressing the following research questions: "Does an OSS project developer community evolve over time with respect to particular social network measures such as density, and/or centrality, and if so, what are the evolutionary patterns of such communities?" The answers to these questions are essential to understanding the community development and its impact on the development process and quality of OSS products.

1530-1605/08 $25.00 © 2008 IEEE

The rest of the paper is organized as follows. We first introduce the literature on analyzing OSS communities using social networks in Section 2. Then, we present our hypotheses in Section 3, followed by an introduction of the methodology we used to analyze data extracted from the developers' mailing list of the KOffice project in Section 4. The results are presented in Section 5. After testing the hypotheses, we discuss major findings and their practical implications in Section 6. Finally, we provide some insights into future research directions in Section 7.

2. Related Work

In the evolution of a complex system of resources and communities, social organizations and tools co-evolve. In order to better understand the evolution of OSS software development, we take a look at the social network of a development community that supports such software.

Compared with traditional software projects, which are defined with relatively stable scopes and goals, OSS evolves in a more significant way because of insufficient planning and requirements collection in advance. Based on studies of socio-technical evolution, Constant [6] argued that communities of practice are the loci of technological practice. According to Constant, technology usually develops through incremental improvement within an existing community of practitioners.

Social network analysis (SNA) is a method traditionally used in social science for understanding the relationships between people, groups, organizations, and other types of social entities. A social network is modeled as a graph with nodes representing the individual actors in the network and ties representing the relationships between actors. Strong ties sustain and promote trust and collaboration in a social network. There have been studies that examined the communities of OSS developers using SNA. Madey et al. [25] modeled OSS projects as a collaborative social network and suggested that the OSS developer community could be modeled as a self-organizing social network. OSS developer communities are generally modeled into three types of social networks:

1) developer-project networks, where ties represent the association between nodes of developers and projects;
2) developer networks, where ties represent the relationship of collaboration; and
3) project networks, where ties represent the relationship of sharing developers.

Collaboration between developers is one of the major issues in an OSS developer community [15]. At a technical level, developers need to collectively determine the technical direction of an OSS product and collaborate on the system development. Collaboration and/or communication failure will result in product failure. Therefore, understanding the roles of individual contributors in a project and their communication/ collaboration patterns in the life time of a project is important to the success of OSS.

Related literature in the social network analysis of OSS can be grouped into four main streams. The first body of research focuses on categorizing the role of members in an OSS project. For example, Xu et al. [25] explored social network properties in the OSS community to identify patterns of collaboration. They measured the diameter, degree distribution, and cluster coefficients of social networks constructed with data extracted from user/developer tables at SourceForge.com. Based on the results, they classified project community members into project leaders, core developers (listed), co-developers, active users, and passive users. 35% of total members were developers. Among them, 16% were core developers. Crowston et al. [7] studied the bug reports of 116 OSS projects on SourceForge.com. They compared three approaches to identifying core developers. The simple approach, which was also used in this study, was to follow a published name list. The alternatives were based on core/periphery social network analysis and the distribution of contribution (Bradford's Law of Scatter). Since developer communities of successful OSS projects tend to be large, it is not possible to study all developers in much detail. Identifying the key figures among all developers becomes essential for further analysis and inspection.

The second body of studies analyzed the impact of social ties on open source project team formation. For example, Hahn et al. [11] randomly selected 1,030 projects from SourceForge.com that registered between Sept. 2005 and Nov. 2005. Then, they revisited selected sample projects one month after their respective registration dates to identify those developers who had subsequently joined the projects. Social network analysis revealed that the existence and the amount of prior social relations in the network increased the probability of an OSS project attracting more developers. For projects without pre-existing social ties, developers tend to join projects initiated by people with less OSS development experience. Huang and DeSanctis [14] studied online discussion forums relating to knowledge management. They employed a network-based approach to examine social capital. The results revealed that two network properties, core–periphery structure and centralization, were related to the mobilization of informational social capital in online networks.

The third stream of previous research focuses on the collaborative structures in an OSS developer community. For example, Grewal et al. [10] analyzed OSS projects to

investigate if there were sufficient differences in collaborative structures. They also measured and quantified the relationship between these structural differences and the success and failure of the associated software development projects. Results indicated that the degree and nature of network embeddedness of both projects and project managers indeed influenced project success in a complex way, moderated by the age of project. Sowe et al. [24] studied the knowledge sharing activities in Debian developer and user mailing lists. They analyzed the distribution of number of postings and replies. They found that the developer list consisted of more replies than posts on average while the reverse was true for the user list.

The fourth type of research in this area is to correlate the developer community structure with software change. For instance, after analyzing messages exchanged through the developer mailing list of the Apache project, Bird et al. [3] found that the number of messages sent by an individual correlated to the number of source changes he/she made. They also found that developers and non-developers were significantly different in measurements of in-degree, out-degree, and betweenness.

A number of prior studies have looked at the evolution aspect of OSS. For example, Gonzalez-Barahona et al. [9] applied the Girvan-Newman (GN) algorithm to the project network of Apache; Robles et al. [22] took a closer look at the growth of Debian distribution package. However, those studies did not address the corresponding developer communities, which were the major driving force behind the success of those OSS projects.

Our study closely resembles the Crowston and Howison' study [5], in which they studied the bug tracking of OSS projects on SourceForge.com. The study showed that those projects displayed a variety of social network structures in terms of centralization. However, it did not account for the different stages of development that each project was facing.

In summary, although it has been well recognized that OSS development has become a significant social phenomenon and that OSS developers and users form a complex social network via electronic communication channels on the Internet [13], few researchers have examined this phenomenon from a social network perspective. Furthermore, those above-mentioned studies only examined various aspects of collaborative networks from a static point of view. Our study attempts to explore the evolutionary patterns of OSS developer communities over time.

3. Hypotheses

First, according to Lehman's laws [17], as a software product evolves, it will become increasingly difficult to add new functionality because of its increasing complexity. We predict that this trend will be mirrored on the social network side. Therefore, we propose our first hypothesis as follows:

H1: The rate of increase in the total effort made to an OSS product will diminish over time.

Second, in an OSS project, a relatively small number of core developers ensure the architectural integrity of the project, such as adding new features and capabilities and tracking day-to-day progress on project goals and tasks [20]. In contrast, peripheral developers help enhance the code and fix the bugs. Therefore, we predict that at the early stage of an OSS project (i.e., during the first couple of releases), the major development task is to build system infrastructure and add major system functions. In most commercial OSS projects, the core developers are paid employees, while secondary developers or peripherals are volunteers. Therefore, core developers will be involved in the communication much more than peripheral developers. While in the later stage (i.e., after several major releases), the major development task gradually becomes bug fixes and other relatively insignificant code changes. Therefore, peripheral developers will play an increased role and the difference in the communication participation between core developers and peripheral developers will be significantly reduced. Thus, we state our second hypothesis as follows:

H2: The difference in communication participation between core developers and peripheral developers will decrease over time.

In addition, there are other exploratory research questions we are interested in addressing, such as: Does an OSS project developer community evolve over time with respect to some social network measures such as density and centrality? If so, what are the evolutionary patterns of such communities? The answers to these questions are essential to understanding the community development and its impact on the development process and quality of OSS products. Therefore, we propose our third hypothesis:

H3: The social network of an OSS project developer community will change over time in observable patterns.

4. Methodology

In the empirical analysis of the evolution of software products and processes, it is important to examine longer time periods so that the analysis results are less likely to be affected by the data fluctuations that may be experienced in shorter time intervals. KOffice is a project that has generated a successfully evolving product over a long period (i.e., a decade), which makes it a suitable candidate for our study.

4.1. Data Collection and Preparation

Mailing lists have been commonly used for analyzing interactions between developers involved in an OSS project (e.g., [16]). Much of the OSS development is done by a small percentage of individuals despite the fact that there could be tens of thousands of developers and users available.

Social network analysis applied to gain an understanding of the OSS development community requires the construction of a sufficiently rich social network based on the available data [25]. While social networks derived from basic membership data such as association of users with projects allow global properties of the community to be calculated, often times they suffer from the overload of unrelated discussions, FAQs, support requests, etc. Such networks are not sufficient to gain a detailed understanding of the development processes occurring within and/or across projects. Therefore, in this study, we constructed social networks from the posts extracted from the developers' mailing list.

We collected data from the official online archive of the KDE (K Desktop Environment) mailing list (http://lists.kde.org/). KOffice is an integrated office application suite developed for KDE. It is an ideal project for our study because it consists of multiple applications (or products) with varying degrees of complexity and features. More importantly, KOffice maintains most of its development activities in one central place. The KDE mailing list is the only publicly announced way to participate in KOffice development. There are two KOffice related mailing lists hosted by KDE: koffice-devel and koffice. We chose the former one because it involves most development-related discussion of KOffice projects, while the latter is not moderated and more user-focused. We did not include messages from Kexi-devel and KPlato lists so that other products without dedicated mailing lists were treated equally. Messages posted after the latest release date at the time of data collection (Release 1.6.1, November 2006) were excluded from this study. This resulted in 15,651 messages spanning from May 18, 2000 to November 29, 2006.

We preprocessed the data before analysis. One challenging issue we had to address was identity resolution. Many developers use multiple alias and/or email addresses. Because our data were collected from a long period of time (more than five years), there are chances for individual developers to change their alias and/or email addresses. Misspellings further exacerbated the problem. By adopting a similar procedure from [3], we resolved identify representations based on the following procedure:

First, we obtained the list of official developers and maintainers from the project website. At the time of this study, the KOffice project site (September 2006 update) reported that there were 74 official personnel consisting of 71 developers and three project maintainers.

Next, we extracted names and email addresses from the mailing list archive, grouped them based on both name and email address, and sorted them alphabetically. Then, including names from the official name list, we manually consolidated similar records based on the following two heuristics:

- If the first and last names of two records match, they refer to the same individual;
- If two email addresses of two different names are identical, those two names were considered to refer to the same individual.

Additionally, two email addresses would be considered a match if they had the same user names and came from the same domain but with different host names (e.g., kevin@umbc.edu matches kevin@is.umbc.edu) Because we were looking at a developer-focused mailing list, we believed that the false positive matches based on the heuristics would be negligible (i.e., less chance of two "John Smith" developers). As a result, we found that 148 out of 531 posters used multiple aliases. We classified individuals whose names were included in the official project developer list as core developers, and the rest as secondary developers, which theoretically include peripheral developers and active users.

Because we are interested in interactions among developers, those messages that did not generate a reply were excluded from further analysis. Overall, we extracted poster name, email address, time of post, message id, thread id, and the message body from 14,511 messages within 2,577 threads. 1,140 non-thread messages were discarded.

Since there was no solid structure between email messages and their responses, threads were conceptually recreated (by KDE archive) by grouping messages with similar subject lines. This resulted in some large threads of actually unrelated messages. Those messages were very difficult to distinguish automatically. They would likely require manual inspection. Perer and Shneiderman

[19] proposed a visualization tool to address this issue, yet it still requires intensive human involvement. Using message content analysis and observing the quoting practice [2] when authors refer to prior messages could be useful in constructing interaction threads with higher accuracy.

KOffice involves a suite of OSS products. To extract threads and messages related to the development of each product from the mailing list, we performed a keyword-based search for twelve KOffice product names as well as for "koffice". The search scope included the subject line and message body of each message. This approach allowed us to classify 93 percent of the total threads. The remaining 175 threads were manually inspected to determine their product associations. Multiple assignments of product names were allowed if a thread involved multiple products (i.e., having multiple keywords). Among them, 603 threads (23%) were assigned to more than one product.

Between major releases of software products, developers may issue several bug-fix releases. In this paper, we call them sub-releases to distinguish from major releases. We identified these sub-releases by an additional digit in version number (i.e., 1.3 versus 1.3.1). Finally, we selected both sub-releases and major releases as the units of analyses for network evolution. Compared with major releases, sub-releases provide finer pictures for network evolution. Accordingly, we used the release dates to create partitions of activities for each release. For example, all mailing list activities occurred between the release of version 1.2.1 and 1.3 were considered relating to 1.3. The release dates of KOffice were obtained from the project site, as well as from the KDE news archive at KDE.NEWS (http://news.kde.org).

We also analyzed the accumulation of mailing list activities from the very beginning till the day of each release. This approach allows us to get a complete picture of activities relating to product evolution.

We created an affiliation matrix between developers and threads. A developer was associated with a thread into which his mail message(s) fall. The number of messages posted by a developer in each thread was recorded to reflect the strength of the tie. The matrix was later transformed into an adjacency matrix among developers. We set the tie strength between developers A and B as the cumulative number of email messages that A writes to threads in which B also writes a message.

One of our underlying assumptions is that a relationship between two developers occurs when they post messages in the same thread. Because of the asynchronous nature of mailing list activities, some may argue that initial posters may or may not come back to visit the thread at later stages. Even if we consider directional relations where each poster only relates to the posters before them, it can be argued that one may not read or respond to every post but only a few prior posts. Since our study does not consider the direction of the relations but rather the overall structure of the group, we can therefore recreate the social network of developers in a simplified way.

4.2. Measures

We counted the number of developers and their corresponding messages. In addition, we also adopted four other network metrics to examine the evolution of OSS social networks. The definitions of the metrics and their measurements are described as the following:

- **Number of Developers** is one of the simplest measures of the social network. Each developer is assigned to the core or secondary group.
- **Number of Messages** is used as a surrogate measure of total effort for each developer. It is also categorized into core or secondary.
- **Density** is the ratio of unweighted relations present versus total possible relations in the network of N developers. Its value ranges from 0 to 1.
- **Clustering Coefficient (CC)** is simply the average of the densities of the neighborhoods of all actors [12]. Its value also ranges from 0 to 1.
- **Group Degree Centrality (GDC)** is the ratio of the graph centralization versus those of a perfect star network of the same size. A perfect star is a network with one central node, which every other node connects to exclusively. We adopted Freeman's definition of GDC [8], as shown in Formula (1).

$$C_D = \frac{\sum_{i=1}^{n}\left[C_D(p^*) - C_D(p_i)\right]}{(n-1)(n-2)} \qquad (1)$$

where p* is the most central point where the degree measurement C_D is the maximum; n is number of nodes in the network.

- **Group Betweenness Centrality (GBC)** is another measure for graph centralization. It is derived from the node-level betweenness, as shown in Formula (2).

$$C_B = \frac{\sum_{i=1}^{n}\left[C_B(p^*) - C_B(p_i)\right]}{(n-1)^2(n-2)} \qquad (2)$$

where p* is the most central point where the betweenness measurement C_B is maximum, n is number of nodes in the network.

The first two metrics can be derived with simple queries. We used Ucinet [4] to compute the third and fourth metrics (i.e., density and clustering coefficient) and developed a software program to compute the last two metrics (i.e., GDC and GBC) automatically. Ucinet's NetDraw allowed us to visualize the affiliation (2-mode) and adjacency (1-mode) matrices of each product-release. Social network matrices were created for each product and each release. At the time of the study, there were 19 KOffice releases including twelve products plus one group of shared central components.

5. Results

The descriptive statistics of the KOffice development community at the end of our data collection period is shown in Table 1. The total number of developers for individual products ranged from 36 to 248. Specifically, the number of core developers ranged from 23 to 55. Compared with the difference in the number of core developers, there was a much bigger difference in the number of secondary developers between different individual products. Additionally, the majority of core to secondary developer's ratios fell into the range of 0.57 to 1 with a couple of exceptions on each end. The ratios for KWord and for KSpread were below 0.3 and 0.4, respectively. On the other hand, the ratios of KPlato and Kugar were above 1. Given that the former two products are the largest and the latter two are the smallest in terms of software size, it seems that the size of products may be negatively associated with the core-to-secondary developer ratio. We will investigate this potential relationship in the future study.

Table 1. The Number of Core and Secondary Developers of KOffice Products

Product	Core	Sec	Total	Core/Sec Ratio
Karbon	42	70	112	0.600000
KChart	33	34	67	0.970588
Kexi	30	42	72	0.714286
KFormula	27	33	60	0.818182
Kivio	38	47	85	0.808511
Kontour	24	24	48	1.000000
KPlato	23	13	36	1.769231
KPresenter	47	82	129	0.573171
Krita	38	50	88	0.760000
KSpread	52	131	183	0.396947
Kugar	27	18	45	1.500000
KWord	55	193	248	0.284974
Shared	61	353	414	0.172805

A similar analysis of the number of message posts is presented in Table 2. As shown in Table 2, the total number of message posts ranged from 176 to 3,925 (excluding shared components). Compared with the average contribution of message posts from core developers (see column 2), the contribution from secondary developers (see column 3) was much less. Nonetheless, the ratio of core-to-secondary message posts was not very big, partly due to the large sized groups of secondary developers. These results show that core developers are key contributors to the mailing list discussions. This observation is confirmed in a two-mode plot of the KWord social network, as shown in Figure 1, and the plot of primary components, as shown in Figure 2. The message threads were denoted by squares while developers are denoted by circles. The lines represent the relation where a developer posts to a message thread.

Besides the shared components, KWord is the most active product in KOffice (in terms of the number of developers and the number of posts), and KWord 1.2 was the most active among all of its product-releases. In Figure 1, core developers, secondary developers, and email threads are denoted in red, blue, and green, respectively. In Figure 2, three core developers (Faure, Zander, and Goutte) clearly stood out from others in this network. Additionally, it is also noted from Figure 1 that some secondary developers at times participated heavily in thread discussion.

Table 2. Descriptive statistics of KOffice message posts

Product	Mean Core	Mean Sec	Total	Core/Sec Ratio
Karbon	6.875000	0.394654	1272	1.533865
KChart	3.492537	0.254777	314	2.925000
Kexi	5.097222	0.200436	459	3.989130
KFormula	4.116667	0.249240	329	3.012195
Kivio	4.400000	0.257937	504	2.876923
Kontour	2.895833	0.210227	176	3.756757
KPlato	5.055556	0.125000	208	7.000000
KPresenter	9.131783	0.206734	1485	3.837134
Krita	9.193182	0.196624	1007	4.085859
KSpread	11.923500	0.216517	2785	3.618574
Kugar	3.044444	0.259459	185	2.854167
KWord	11.875000	0.249682	3925	3.005102
Shared	13.99275	0.266337	7896	2.754636

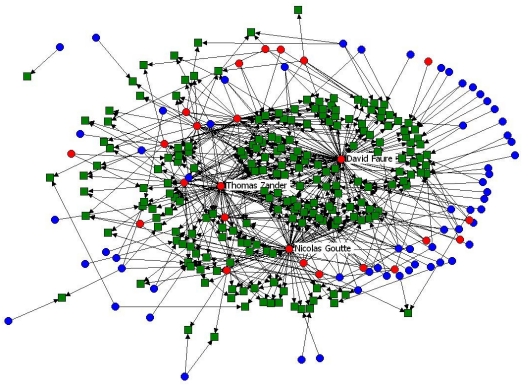

Figure 1. The two-mode network of KWord 1.2

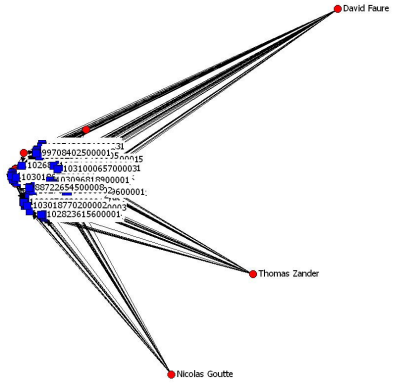

Figure 2. Primary Components of KWord 1.2

Many activities, either directly or indirectly, affect the outcome of a software project. Since we are working with mailing list posts in this study, we used the number of discussion posts as an indicator of the total effort as noted in Section 3.2.

For the evolution patterns, we selected KWord, KSpread, and Kpresenter, the top-three products in KOffice based on the number of developers. Figure 3 shows that the increase in total effort did not follow a straight line, but rather a logarithmic function. Therefore, hypothesis 1 is supported.

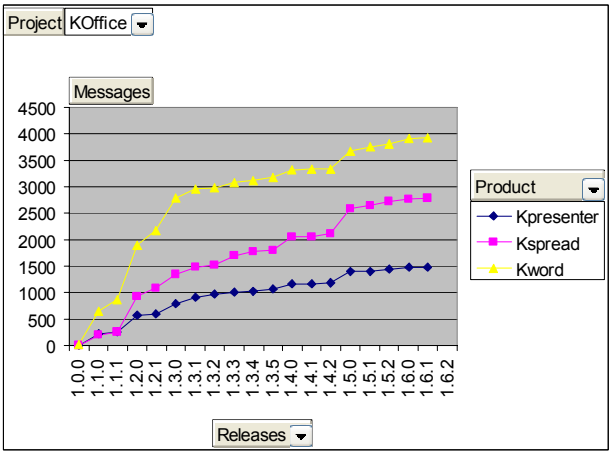

Figure 3. The total effort associated with product releases

Next, we examined the evolutionary patterns of differences in the number of messages exchanged between core and secondary developers, as shown in Figure 4. Overall, the trend lines are decreasing with exceptions at the major releases. Thus, they indicate that the difference in communication participation between core developers and peripheral developers will decrease over time. So, hypothesis 2 is supported.

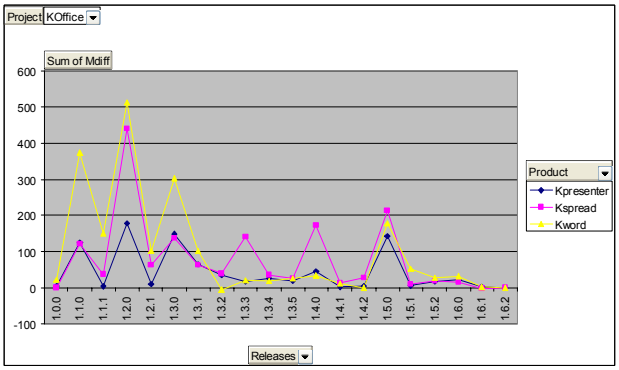

Figure 4. The difference in number of messages posted by core and secondary developers over time

Figures 5, 6, 7, and 8 display the evolution patterns based on social network measures for our selected top-three products. Based on these results, it is clear that there are some general patterns about how social networks of developers evolve over time. Therefore, hypothesis 3 is supported.

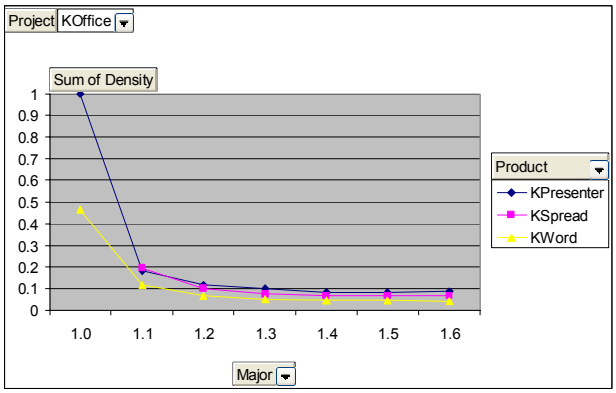

Figure 5. The evolution pattern of density

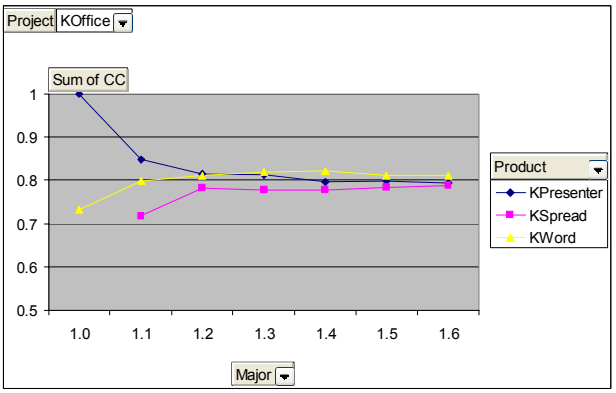

Figure 8. The evolution pattern of CC

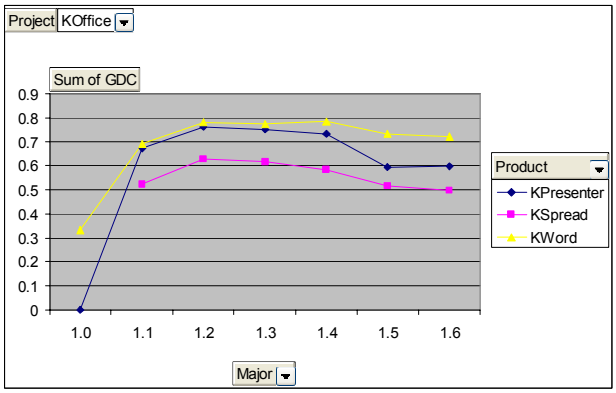

Figure 6. The evolution pattern of GDC

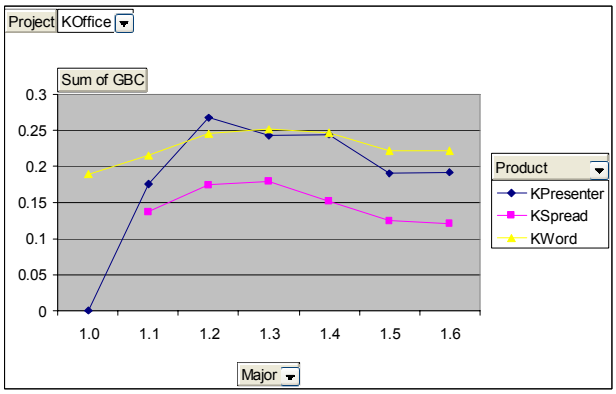

Figure 7. The evolution pattern of GBC

Based on Figure 5, we observe that for all three selected products, developers' communication network density experienced a rapid decrease at the beginning of the project, and then stabilized. In early releases of KOffice products, many developers started to join the group when the communication links between different developers had not yet been established. Therefore, density experienced a rapid decrease. As the developers continue to join and collaborate with one another on resolving design issues and implementing system functions, the communication among developers became more intense and the number of developers increased at a much slower pace, hence network density stabilizes.

It is shown in Figure 6 that the group degree centrality experienced a rapid increase at the beginning, then slowly decreased, and finally stabilized. In general, the graph shows that the developer networks are neither star-like nor fully connected but fall somewhere in between. The rapid increase at an early period indicates that a group of core developers quickly emerge as the center of the networks. The central role of a small group of core members tends to decrease as the network size increases. The slow decreasing trend is discontinued at a later stage as a small number of long-term committed developers begin to stand out.

The evolution trend of GBC, as shown in Figure 7, is similar to that of GDC but with a much larger variation over time. The findings suggest that GBC is more sensitive to the change in individuals' positions in a developer network.

Clustering coefficient was relatively stable, which increased very slowly over time, as shown in Figure 8. The metrics indicate the tendency to form local developer sub-communities over time. These sub-groups are likely to be core developers who have made a long-term commitment to the community.

6. Discussion

We found that the rate of increase in the total effort diminished over time. The shape of the total effort curves we obtained did not look linear, but rather logarithmic. This result has important implications for the management of OSS projects. It implies that a constant effort may not be required over the life-time of an OSS product (Otherwise we would observe a linear increase in communication). The logarithmic increase in effort occurred not because KOffice met a large portion of its requirements or reached a high quality level early in its life. Indeed, there are always new requirements for any software with considerable functionality and usage. OSS communities constantly discuss and propose new features. The non-linearity in effort can be better explained by the inherent difficulty to add more functionality due to the increasing complexity of software as stated by Lehman [17].

We found decreasing patterns of difference in communication participation between core and secondary developers in all three selected products. Upon closer inspection of the data, we observed that both groups had low communication participation. When the core group has higher input in early releases, the resulting reduction of the difference becomes more significant. We did not find an increase in secondary developer participation as a result of bug fixes as we speculated. This may be due to the fact that bug fixing activities in KOffice are recorded in a separate bug tracking system. It may therefore be useful to combine additional data from that system in follow-up studies.

Finally, we identified interesting evolutionary patterns that all three selected products had in common. Detailed studies to understand and predict these patterns could be useful for OSS project planning and management. OSS managers and developers should consider these evolutionary patterns in order to manage and utilize their resources efficiently.

7. Conclusion and Future Research

In this study, we applied a social network approach to examining the evolutionary changes and patterns in the OSS project communities. To our best knowledge, it is the first attempt of this kind. We analyzed data collected from KOffice, an OSS project with a relatively long history. The results confirmed our predictions that developer community networks change over time in certain patterns. Understanding those evolutionary patterns can greatly help OSS managers better manage the community and project.

One limitation of this study lies in the inability of mailing list archives to record response relationships between messages. As a result, we assumed a message post from a participant in the Koffice mailing list is related to all other participants in the thread in which the post appears. In another study that used bug reports [7], a message was considered replying to a prior post. We felt our strategy might be more reflective of the nature of mailing list archives. There is still a need to develop a better mechanism for uncovering the relationships between members of a mailing list. In addition, we used simple keyword-based matching to assign threads to the products. The grouping of threads from mailing list archives may not be completely accurate due to the possibility of involving unrelated subjects or keywords in messages. A Content analysis approach could be employed to improve the grouping of threads by evaluating the accumulative probability of the reference to products.

There are several future research directions. First, we plan to develop and adopt new metrics to evaluate the evolution of the roles of developers in an OSS community [18]. The results could help a community identify potential project leaders or core developers from a dynamic perspective. Second, we can assess the validity of network evolution patterns discovered from this study using other OSS communities. In this study, we only analyzed one OSS project, KOffice. It would be interesting to see if the findings of this study can be validated by other long-term OSS projects. Third, it would be very beneficial to investigate the relationship between the evolution of social networks and the success of an OSS community, which can provide insights into how to attract and retain developers within an OSS project.

References

1. Aoyama, M. Metrics and analysis of software architecture evolution with discontinuity *Proceedings of the International Workshop on Principles of Software Evolution*, ACM Press, Orlando, Florida, 2002.

2. Barcellini, F., Detienne, F., Burkhardt, J.-M. and Sack, W. Thematic coherence and quotation practices in OSS design-oriented online discussions *Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work*, ACM Press, Sanibel Island, Florida, USA, 2005.

3. Bird, C., Gourley, A., Devanbu, P., Gertz, M. and Swaminathan, A. Mining email social networks *Proceedings of the 2006 international workshop on Mining software repositories*, ACM Press, Shanghai, China, 2006.

4. Borgatti, S.P., Everett, M.G. and Freeman, L.C. Ucinet for Windows: Software for Social Network Analysis, Analytic Technologies, Harvard, MA, 2002.

5. Conklin, M., Howison, J. and Crowston, K. Collaboration using OSSmole: a repository of FLOSS data and analyses *Proceedings of the 2005 international workshop on Mining software repositories*, ACM Press, St. Louis, Missouri, 2005.

6. Constant, E.W. The Social locus of technological practice: community, system, or organization? in Bijker, W.E., Hughes, T.P. and Pinch, T.J. eds. *The Social Construction of Technological Systems: New Directions in the Sociology and History of Technology*, MIT Press, Cambridge, MA, 1987, 223-242.
7. Crowston, K., Wei, K., Li, Q. and Howison, J., Core and periphery in Free/Libre and Open Source software team communications. in *Hawaii International Conference on System Sciences*, (Hawaii, 2006).
8. Freeman, L.C. Centrality in Social Networks: Conceptual Clarification. *Social Networks*, *1*. 215-239.
9. Gonzalez-Barahona, J.M., Lopez, L. and Robles, G. Community structure of modules in the Apache project. *TBD*.
10. Grewal, R., Lilien, G.L. and Mallapragada, G. Location, Location, Location: How Network Embeddedness Affects Project Success in Open Source Systems. *Management Science*, *52*. 1043–1056.
11. Hahn, J., Moon, J.Y. and Zhang, C., Impact of Social Ties on Open Source Project Team Formation. in *The Second International Conference on Open Source Systems - OSS 2006*, (Como, Italy, 2006).
12. Hanneman, R.A. and Riddle, M. Introduction to social network methods, University of California, Riverside, 2005.
13. Hippel, E.V. and Krogh, G.G.V. Open Source Software and the "Private-Collective" Innovation Model: Issues for Organization Science. *Organization Science*, *14*. 209-223.
14. Huang, S. and DeSanctis, G., Mobilizing Informational Social Capital in Cyber Space: Online Social Network Structural Properties and Knowledge Sharing. in *Twenty-Sixth International Conference on Information Systems*, (Las Vegas, 2005), 207-219.
15. Jensen, C. and Scacchi, W. Collaboration, Leadership, Control, and Conflict Negotiation in the Netbeans.org Open Source Software Development Community *38th Annual Hawaii International Conference on System Sciences (HICSS)*, Big Island, HI, 2005.
16. Kuk, G. Strategic Interaction and Knowledge Sharing in the KDE Developer Mailing List. *Management Science*, *52*. 1031-1042.
17. Lehman, M.M. and Belady, L.A. *Program Evolution: Processes of Software Change* Academic Press, London, 1985.
18. Nolker, R.D. and Zhou, L., Social Computing and Weighting to Identify Member Roles in Online Communities. in *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, (2005), 87-93.
19. Perer, A. and Shneiderman, B. Beyond Threads: Identifying Discussions in Email Archives *IEEE Information Visualization (InfoVis)*, IEEE, Minneapolis, MN, 2005.
20. Porter, A., Yilmaz, C., Memon, A.M., Krishna, A.S., Schmidt, D.C. and Gokhale, A. Techniques and Processes for Improving the Quality and Performance of Open-Source Software. *Software Process Improvement and Practice*, *11*. 163-176.
21. Roberts, J.A., Hann, I.-H. and Slaughter, S.A. Understanding the motivations, participation, and performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects. *Management Science*, *52*. 984-999.
22. Robles, G., Gonzalez-Barahona, J.M., Michlmayr, M. and Amor, J.J. Mining large software compilations over time: another perspective of software evolution *Proceedings of the 2006 international workshop on Mining software repositories*, ACM Press, Shanghai, China, 2006.
23. Shah, S.K. Motivation, governance, and the viability of hybrid forms in open source software development. *Management Science*, *52*. 1000-1014.
24. Sowe, S.K., Stamelos, L. and Angelis, L. Understanding knowledge sharing activities in free/open source software projects: An empirical study. *The Journal of Systems and Software* (In Press).
25. Xu, J., Gao, Y., Christley, S. and Madey, G., A Topological Analysis of the Open Souce Software Development Community. in *the 38th Hawaii International Conference on System Sciences*, (Hawaii, USA, 2005), 198a-198a.