Accelerating Cross-Project Knowledge Collaboration Using Collaborative Filtering and Social Networks

Masao Ohira Naoki Ohsugi Tetsuya Ohoka Ken–ichi Matsumoto
Graduate School of Information Science
Nara Institute of Science and Technology
8916-5, Takayama, Ikoma, Nara, JAPAN 630-0192
tel.+81(743)-72-5318 fax.+81(743)-72-5319
{masao, naoki-o, tetsuy-o, matumoto}@is.naist.jp

ABSTRACT

Vast numbers of free/open source software (F/OSS) development projects use hosting sites such as Java.net and SourceForge.net. These sites provide each project with a variety of software repositories (e.g. repositories for source code sharing, bug tracking, discussions, etc.) as a media for communication and collaboration. They tend to focus on supporting rich collaboration among members in each project. However, a majority of hosted projects are relatively small projects consisting of few developers and often need more resources for solving problems. In order to support cross-project knowledge collaboration in F/OSS development, we have been developing tools to collect data of projects and developers at SourceForge, and to visualize the relationship among them using the techniques of collaborative filtering and social networks. The tools help a developer identify "who should I ask?" and "what can I ask?" and so on. In this paper, we report a case study of applying the tools to F/OSS projects data collected from SourceForge and how effective the tools can be used for helping cross-project knowledge collaboration.

Categories and Subject Descriptors

H.5 [**Information Interfaces and Presentation**]: Group and Organization Interfaces—*Collaborative computing, Computer–supported cooperative work, Organization Design, Web-based interaction*

General Terms

Management, Measurement, Human Factors

Keywords

Knowledge Collaboration, Social Networks, Collaborative Filtering, Visualization Tool

1. INTRODUCTION

Vast numbers of free/open source software (F/OSS) development projects use hosting sites such as Java.net and SourceForge.net. These sites provide each project with a variety of software repositories (e.g. repositories for source code sharing, bug tracking, discussions, etc.) which can be seen as knowledge repositories for software development in the aggregate. Many researchers focus on exploiting such the repositories for supporting software development nowadays [4, 9].

While each project can accumulate its own knowledge through software development into the repositories easily, the "freely accessible" knowledge across projects is not supported sufficiently. In order to help cross-project knowledge collaboration in F/OSS development, we have been developing tools to collect data of projects and developers at SourceForge, and to visualize the relationship among them using the techniques of collaborative filtering and social networks. The tools help a developer identify "who should I ask?" and "what can I ask?" and so on.

In what follows, we first discuss the need for supporting cross-projects knowledge collaboration based on our analysis of SourceForge. Then we describe the procedure of mining software repositories at SourceForge using our tools. In the next section, we report a case study of applying Graphmania to the F/OSS projects data collected from SourceForge and illustrate how effective the tool can be used for helping cross-project knowledge collaboration.

2. NEED FOR KNOWLEDGE COLLABORATION ACROSS PROJECTS

Recent studies on F/OSS communities revealed that F/OSS communities needed further developers and people's contribution to software development. For instance, [8] reported only 4% of developers in the Apache community created 88% of new code and fixed 66% of defects. From a total of 196 developers in the Ximian project, 5 developers account for 47% of the modification requests (MRs), while 20 account for 81% of the MRs, and 55 have done 95% of them [3]. 4% of members account for 50 percent of answers on a user-to-user help site [5].

Projects with a large proportion of non-contributors have difficulty providing needed services such as bug fixes and software enhancements to all members [1]. The existence of highly motivated members would be the key success factor of a F/OSS project[2]. As an approach to motivate members

Table 1: Number of Projects with n Developers

NUMBER OF DEVELOPERS	NUMBER OF PROJECTS	(%)
0	278	0.3
1	60665	66.7
2	14151	15.6
3	5854	6.4
4	3222	3.5
Over 5	6732	7.4
TOTAL	90902	100

Table 2: Number of Developers on p Projects

NUMBER OF PARTICIPATING PROJECTS	NUMBER OF DEVELOPERS	(%)
1	100408	77.3
2	18753	14.4
3	5980	4.6
4	2350	1.8
Over 5	2406	1.8
TOTAL	129897	100

of online communities, some theories such as social capital (e.g. ExpertsExchange[1]) and social networks [6, 12] have attracted attention recently.

Relatively small projects registered at hosting sites are confronted with more difficulties than such the large projects mentioned above (e.g. the Apache project), because (1) those projects consist of few developers and contributors generally and (2) the hosting sites provide a variety of tools for rich communication and collaboration among members in each project but do not provide them with tools for exchanging or sharing problem-solving knowledge across projects directly.

To confirm the issue related to (1), we collected and analyzed the data of over 90,000 projects and about 130,000 developers[2] at SourceForge in February 2005. Table 1 shows the number of projects with n developers. 66.7% of overall projects at SourceForge had only one developer. The maximum of number of developers in one project was 272. Table 2 shows the number of developers on p projects. 77.3% of overall developers at SourceForge belonged to one project. The maximum of number of projects a developer joined was 51.

These results are very similar to the results of the social networks analysis in SourceForge in February 2002 [7]. As Madey et al. mentioned in [7], these results indicate that a small number of developers at SourceForge have rich links to others (i.e. the "rich–get–richer" effect) but a majority of developers does not have sufficient links to ask other projects' developers to help them solve problems which happened in their own projects. We believe that it is useful to give developers in small projects means to access other developers and projects that possess the information

[1]http://www.experts-exchange.com

[2]The total number of registered users at SourceForge.net are over 1,000,000. We collected the data of members who are participating in projects actually.

Table 3: Project Information (e.g. the phpMyAdmin Project at SourceForge.net)

ATTRIBUTES	EXAMPLE
project name	phpMyAdmin
description	phpMyAdmin is a tool written in PHP intended to handle the administration of MySQL ...
num. of developers	8
keywords	php, databases, ...
program lang.	PHP
operating system	OS Independent
license	GPL
status	5 – Production/Stable
registered	2001/3/18 02:07
intended audience	Developers, End Users/Desktop, System Administrators
user interface	Web-based
topics	Front-Ends, Dynamic Content, Systems Administration

Table 4: Developer Information (e.g. One of authors registered at SourceForge.jp)

ATTRIBUTES	EXAMPLE
login name	Ohsugi
public name	Naoki Ohsugi
email address	ohsugi at users.sourceforge.jp
site member since	2002/6/10 22:16
group member of	NAIST Collaborative Filtering Engines, Game-R, Bullflog
skill inventory	C/C++: Competent: 5 yr–10 yr Perl: Competent: 5 yr–10 yr Java: Want to Learn: 2 yr–5 yr

or knowledge relevant to solving problems. The next section describes the procedure of mining software repositories at SourceForge using our tools and then illustrates how the tools works.

3. GRAPHMANIA: A TOOL FOR HELPING KNOWLEDGE COLLABORATION

In order to support cross-project knowledge collaboration, we have been developing Graphmania, a tool for visualizing the relationship among developers and projects using the techniques of collaborative filtering and social networks. The tool helps a developer identify "who should I ask?" and "what can I ask?" and so on.

3.1 Data Collection

Using an autopilot tool for SourceForge.net[3] written in Ruby, we have collected two data sets; about 90,000 projects' information (table 3) and 130,000 developers' information (table 4). In what follows, for simplicity, we suppose that the data we use in this paper is **project name** and **num. of developers** from the project info. , and **login name** and **group members of** from the developer info. only (other attributes will be used in the near future).

[3]available from the third author upon your requests

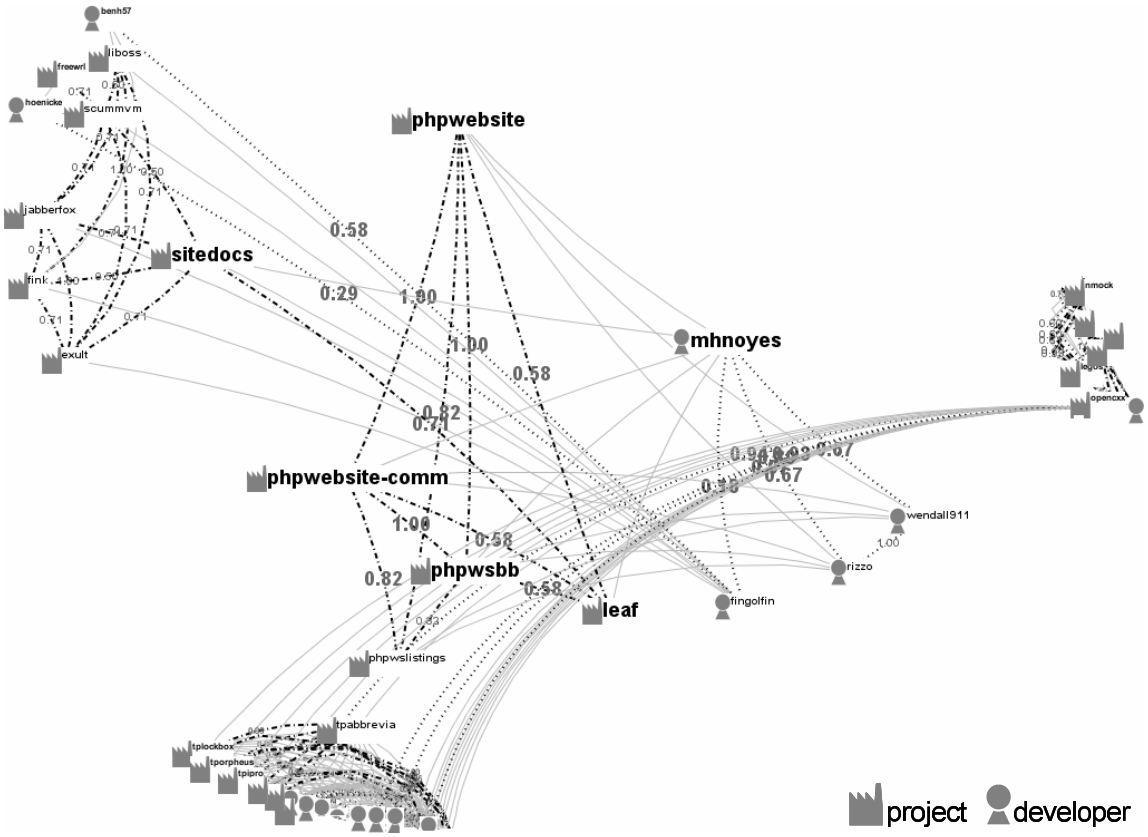

Figure 1: Developer-project networks (numbers in-between edges are the degree of similarity between nodes)

3.2 Social Network Analysis

In our approach, we can present three types of collaborative social networks using above the data; developer networks, project networks, and developer-project networks. In the developer networks, nodes correspond to developers. If two developers have participated in a same project, the two developers (nodes) are linked each other by an edge. In a similar fashion, a project node in the project networks is linked to another project node if a same developer has joined the two projects. The developer-project networks is a representation combined the two networks of developers and projects (figure 1). All the three networks are represented as undirected networks.

Most of studies on social network analysis often define degree, weight and distance of nodes to characterize the topology of networks [6, 7]. In our approach, however, the results of the similarity calculations of Graphmania using collaborative filtering are used to determine whether a node should be linked to another node or not.

3.3 Collaborative Filtering

Graphmania is a tool for calculating similarities among nodes and visualizing the results as social network graphs, incorporating NAIST Collaborative Filtering Engines (NCFE)[4] (see detailed algorithms in [10, 11]).

The similarities among developers are calculated using the frequencies of participations in same projects. If *developer i* and *developer j* join same projects many times, the similarity among the two developers is rated highly. In the same way, the similarities among projects are calculated using the frequencies of co-existence of same developers in each project. If there are many developers who are working together for both *project m* and *project n*, the similarity among the two projects is rated highly.

In general, recommender systems such as Amazon.com use similarities for predicting and recommending customers' preferred books as "Customers who bought this book also bought..." Graphmania does not recommend anything currently but defines the relationship among developers and projects using calculated similarities as threshold values. Graphmania dose not visualize relations among developers and projects with low similarities in order to avoid the complexity of visualized results.

3.4 Visualizations

Graphmania shows an undirected network graph based on the calculated similarities among F/OSS developers and projects. For developer-project networks (figure 1), the graph

[4]Graphmania and NAIST Collaborative Filtering Engines (NCFE) available from http://sourceforge.jp/projects/ncfe

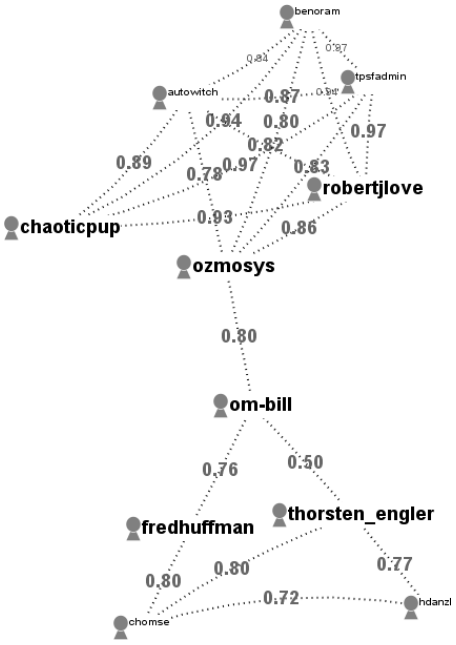

Figure 2: Developer networks (The two developers play a role of a linchpin between two social networks)

consists of two different types of nodes and three different types of edges. A dotted edge connects two developers who are working together for same projects. A dash-dotted edge connects two projects that have same developers. Each black line edge represents the relationship among projects and developers (i.e. who is working for which projects and which projects have whom).

Graphmania uses HyperGraph[5] to provide users with hyperbolic views for visualizations and with interactivity to visualized results. Hyperbolic visualizations help users understand information in detail while keeping an overview of information (such the technique is called "focus + context"). Since each node can have an URL to a website as a function of HyperGraph, users are able to access to the site (developers' information pages or projects' HP) as soon as users can find an interesting node.

4. A CASE STUDY

This section describes a case study of applying Graphmania to F/OSS projects data collected from SourceForge and how effective it can be used. As a condition of the similarity calculation, we selected nodes with maximum 5 edges. This means we used only a few percent of over 90,000 projects data for reducing the amount of the similarity calculation.

Developer networks:
Figure 2 represents a part of developers networks snipped. If you have maximum 5 edges, you can find the strength of each edge comparing with similarities because it implies shared history of participating in same projects. Even if you have only a few edges, it would be also helpful to notice important developers who play a role of a linchpin in the cluster because you have possibilities to contact others via the linchpin. For contacting others, just click the node you are interested in and then you will be able to reach a developers' information page. Developer networks are basically same ones as social networks in common online communities.

Project networks:
Figure 3 represents a part of project networks snipped. You can notice that similar name projects organize one cluster because this indicates that projects that share specific purposes or goals tend to have similar names (e.g. a project related to TurboPower have "tp" at the head of a project name). If you are a member of a project in such the cluster, you might find interesting projects related to software you create and might be able to obtain the useful information for your software development from the members of the project. Project networks are a good example of taking advantage of collaborative filtering by a common practice in which similar projects have similar project names.

Developer–project networks:
Using developer–project networks (figure 1), you can easily notice your neighborhoods who are joining similar projects with the red nodes and edges. You are easy to ask something to these neighborhoods because they are likely acquaintances and seem to have similar interests of F/OSS technologies. You can also recognize the projects' neighborhoods spend much effort from number of developers. These projects might have ideas, technologies and solutions for problems, which you need. Developer–project networks are a bit complex but useful for finding developers and projects related to your own.

5. CONCLUSION AND FUTURE WORK

In this paper, we described the issues on motivating F/OSS (online) projects and needs for supporting knowledge collaboration across projects. We introduced Graphmania, a tool for visualizing the relationship among developers and projects using the techniques of collaborative filtering and social networks.

In the near future, we have a plan to use other attributes of the collected data listed in Table 3 and Table 4 for more effective visualizations based on NCFE. We would also like to extend the tool according to the Dynamic Collaboration (DynC) framework [13] because the current tool cannot help user control the amount of communication so that "rich" developers or projects can prevent taking a lot of questions and requests from "poor" developers or projects. Then we would like to evaluate the tool through actual uses of F/OSS developers.

6. ACKNOWLEDGMENTS

This work is supported by the EASE (Empirical Approach to Software Engineering) project[6] and supported by Grant 15103 of the Open Competition for the Development of Innovative Technology program, the Comprehensive Development of e-Society Foundation Software program of the Ministry of Education, Culture, Sports, Science and Technology of Japan.

[5]http://hypergraph.sourceforge.net/

[6]http://www.empirical.jp/

Figure 3: Project networks (The large cluster in center consists of projects related to TurboPower. The isolated small cluster on the upper left consists of projects related to Linux.)

7. REFERENCES

[1] G. Beenen, K. Ling, X. Wang, K. Chang, D. Frankowski, P. Resnick, and R. E. Kraut. Using social psychology to motivate contributions to online communities. In *Proc. of the 2004 ACM conf. on Computer Supported Cooperative Work (CSCW'04)*, pages 212–221, 2004.

[2] J. Feller and B. Fitzgerald. *Understanding Open Source Software Development*. Addison-Wesley, 2002.

[3] D. German and A. Mockus. Automating the measurement of open source projects. In *Proc. of the 3rd Workshop on Open Source Software Engineering*, pages 63–67, 2003.

[4] A. E. Hassan, R. C. Holt, and A. Mockus, editors. *Proc. of 1st Intl. Workshop on Mining Software Repositories (MSR2004)*, 2004.

[5] K. R. Lakhani and E. von Hippel. How open source software works: "free" user-to-user assistance. *Research Policy*, 32(6):923–943, 2003.

[6] L. Lopez-Fernande, G. Robles, and J. M. Gonzalez-Barahona. Applying social network analysis to the information in CVS repositories. In *Proc. of 1st Intl. Workshop on Mining Software Repositories (MSR2004)*, pages 101–105, 2004.

[7] G. Madey, V. Freeh, and R. Tynan. The open source software development phenomenon: An analysis based on social network theory. In *Americas conf. on Information Systems (AMCIS2002)*, pages 1806–1813, 2002.

[8] A. Mockus, R. Fielding, and J. D. Herbsleb. Two case studies of open source software development: Apache and mozilla. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 11(3):309–346, 2002.

[9] M. Ohira, R. Yokomori, M. Sakai, K. Matsumoto, K. Inoue, and K. Torii. Empirical project monitor: A tool for mining multiple project data. In *Proc. of 1st Intl. Workshop on Mining Software Repositories (MSR2004)*, pages 42–46, 2004.

[10] N. Ohsugi. *A Framework for Software Function Recommendation Based on Collaborative Filtering*. NAIST–IS–DT0361006, Graduate School of Information Science, Nara Institute of Science and Technology, 2004.

[11] N. Ohsugi, A. Monden, and K. Matsumoto. A recommendation system for software function discovery. In *Proc. of 9th Asia-Pacific Software Engineering conf. (APSEC'02)*, pages 248–257, 2002.

[12] Y. Ye and K. Kishida. Toward an understanding of the motivation open source software developers. In *Proc. of the 25th Intl. conf. on Software Engineering (ICSE'03)*, pages 419–429, 2003.

[13] Y. Ye, Y. Yamamoto, and K. Kishida. Dynamic community: A new conceptual framework for supporting knowledge collaboration in software development. In *Proc. of 11th Asia-Pacific Software Engineering conf. (APSEC'04)*, pages 472–481, 2004.