Mining CVS Repositories to Understand Open-Source Project Developer Roles

Liguo Yu
Computer Science and Informatics
Indiana University South Bend
South Bend, IN, USA
ligyu@iusb.edu

Srini Ramaswamy
Computer Science Department
University of Arkansas at Little Rock
Little Rock, AR, USA
srini@acm.org

Abstract

This paper presents a model to represent the interactions of distributed open-source software developers and utilizes data mining techniques to derive developer roles. The model is then applied on case studies of two open-source projects, ORAC-DR and Mediawiki with encouraging results.

1. Introduction

The increasing use of open-source software coupled with the expanding of open-source community reflects the importance of open-source development and is no longer being ignored by software engineering researchers. Considerable studies have been conducted in this field. For example, Reis and Fortes [1] analyzed the development of the Mozilla web browser to model the software process. Jensen and Scacchi [2] studied the negotiation of conflicts, collaborative efforts, and leadership in the Netbeans community. Espinosa et al. [3] investigated how shared mental models, work familiarity and geographic dispersion affect coordination in software teams. Elliott and Scacchi [4] analyzed the virtual organization of GNU project. Scacchi [5] outlined the socio-technical activities of open-source projects in different communities. Madey et al. studied open-source community as a social network [6].

In this paper we present our results of mining CVS repositories to understand developer roles on software projects.

2. Open-source software developer model

For two developers *i* and *j*, we define *interaction frequency* as the degree of interactions between *i* and *j* based on one or more measures between them. The measurement of interaction frequency is a context based concept, which means different measures may result in different interaction frequency. In distributed open-source development, candidate measures for interaction frequency are the frequency of email correspondence, the frequency of co-editing, the frequency of task sharing, and so on.

For a project that contains n developers, the degree of interactions between these n developers is represented as an n×n matrix, in which item at position (i, j) is the interaction frequency between developer i and developer j.

2.1 Clustering and role identification

Complete-linkage hierarchical clustering method [7] is used to group developers according to the interaction frequencies between them. Given a set of n developers to be clustered, and an n×n interaction matrix, the basic procedure is described below.

1. Start by assigning each developer to a cluster. Let the interaction frequency between the clusters be the same as the interaction frequency between the developers they contain.
2. Find the pair of clusters that have the largest interaction frequency (say, IF) and merge them into a single cluster. The new cluster is said to have the interaction frequency of C_{IF}=IF.
3. Compute interaction frequency between the new cluster and each of the other (old) clusters. The interaction frequency between them is considered to be equal to the smallest interaction frequency from any member of one cluster to any member of the other cluster.

Repeat steps 2 and 3 until all developers are clustered into a single cluster of size n. The complete-linkage hierarchical clustering is used to ensure that the interactions between every two members in a cluster have at least the interaction frequency equal to the frequency of the cluster C_{IF}.

For a small-size and medium-size project, the developer roles can be basically divided into two types, core members and associate members. After clustering, the developers that are in a cluster with interaction frequency (C_{IF}) greater or equal to a specified threshold are called core members. The rest of the developers are called associate members.

It can be seen from the specification that the interaction frequencies between any pair of core members are greater or equal to the threshold. It worth noting that (1) the definition of core member and associate member are context based, which means, the determination of the specified threshold of interaction frequency threshold is based on the analysis of the specific project; (2) the two-level role (core member and associate member) categorization scheme is best for small-sized and medium-sized (less than 100 developers) projects. For large-size (say, over 200 developers) projects, it might be appropriate to have more levels (categories) of developer roles.

2.2 Rule extraction and prediction

In practice, it is important to analyze the external attributes of different developer roles and determine the rules that can predict the developer roles according to the external attributes. In this study, data classification technique [8] is used to build predictive rules. The procedure is described below.

1. Determine the attributes of a developer that can be used for the rule establishment. The data is then arranged in a two dimension table in which each column represents one attribute and each row contains a record of a developer.
2. Rules are constructed using the training data set.
3. The predictive accuracy, percentage coverage, and attribute significance of each rule are analyzed. The predictive accuracy is the percentage of test set samples that are correctly classified by the rule. The percentage coverage is the percentage of a developer role that can be predicted by the rule. The attribute significance is derived by subtracting the smallest class mean from the largest mean value. In this study, the rules that have predictive accuracy less than 0.9, or percentage coverage less than 0.7, or attribute significance less than 0.25 are considered insignificant and have to be eliminated.

3. Case studies

In this research, one small-size open-source project and one medium-size open-source project are investigated. They are ORAC-DR [9], which contains 14 developers, and Mediawiki [10], which contains 56 developers. The interaction frequency is represented with the measure of the number of common source code modules two developers share. The value of interaction frequency threshold is specified to be 10. Our selection of interaction frequency threshold is tentative and needs to be further validated.

3.1 Clustering and role identification

ORAC-DR contains 14 developers. The clustering results show that 5 developers are the core members. The rest are associate members. Figure 1 shows the organization of the ORAC-DR development team. Strong interactions (interaction frequency is greater than 10) exist among core members, weak interactions (interaction frequency is less than 10) are found between associate member and core member. Few lower-degree interactions (interaction frequency is less than or equals to 2) are also found among associate members.

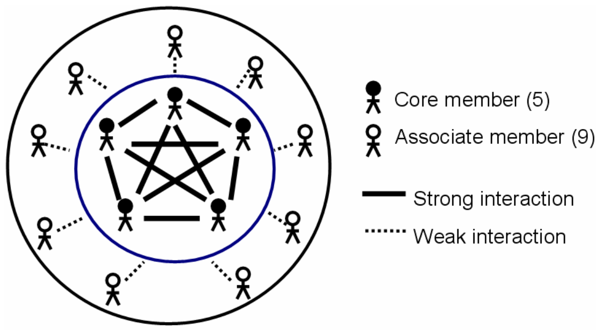

Figure 1. The organization of ORAC-DR development team

Mediawiki contains 56 developers. After clustering, the development team is divided into two groups. Each group contains core members and associate members. Figure 2 shows the developer organization of the Mediawiki development team. Strong interactions (interaction frequency is greater than or equal to 10) exist between core members; weak interactions (interaction frequency is less than 10) exist between associate members and core members in the same group; medium interaction (interaction frequency is between 5 to 10) exist between core members of different groups; few lower-degree interactions (interaction frequency is less than or equals to 2) exist between associate members.

Although in theory, strong interactions could exist between an associate member and a particular core member, in the studies of ORAC-DR project and Mediawiki project, no such interactions were found.

3.2 Rule extraction and cross-validation

For each developer, the following attributes are used as inputs of the rules: percentage of revisions, percentage of modified lines of code, and major workday. The major workday is categorized as everyday, every weekday, part weekday, and weekend. The role category (core member and associate member) is used as output of rules. Table 1 and Table 2 show the rules that satisfy the thresholds specified in Section 2.

Two rules are generated for ORAC-DR project, one is for core member and one is for associate member. Four rules are generated for Mediawiki project, two are for core members, two are for associate members. The results also show that workday does not contribute to the classification of developer roles.

ORAC-DR and Mediawiki are two open-source projects that are of different sizes and under different domains. To understand whether the rules generated in one project are applicable to another project, the rule cross-validation is performed and shown in Table 3.

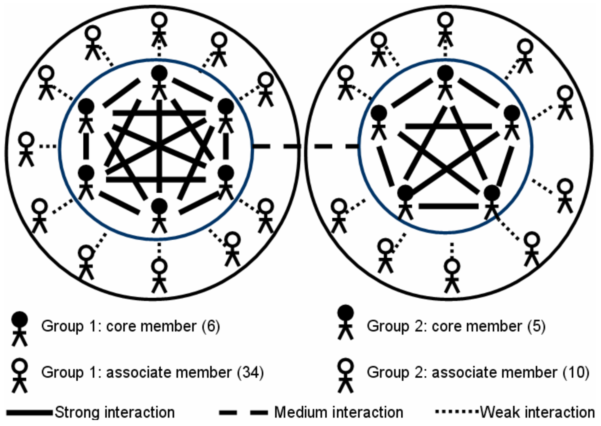

Figure 2. The organization of Mediawiki development team

Table 3 shows that the rules are cross project valid. Despite ORAC-DR and Mediawiki are two different projects, they share some similarities in charactering core members and associate members. This also indicates that the selection of 10 as interaction frequency threshold is valid for the two projects.

Table 1. The developer role prediction rules of ORAC-DR project

Rule number	Role	Rule	Accuracy	Coverage
1	Core	Percentage of modified lines ≥ 4.4%	100%	80%
2	Associate	Percentage of revisions <= 1.4%	100%	89%

Table 2. The developer role classification rules of Mediawiki project

Rule number	Role	Rule	Accuracy	Coverage
3	Core	Percentage of revisions ≥ 2.1%	100%	92%
4	Core	Percentage of modified lines ≥ 3.6%	100%	82%
5	Associate	Percentage of revisions <= 3.3%	94%	100%
6	Associate	Percentage of modified lines <= 2.9%	96%	100%

Table 3. Cross-validation of the prediction rules

Rule number	Role	Generated from	Applied on	Accuracy	Coverage
1	Core	ORAC-DR	Mediawiki	100%	73%
2	Associate	ORAC-DR	Mediawiki	100%	98%
3	Core	Mediawiki	ORAC-DR	83%	100%
4	Core	Mediawiki	ORAC-DR	67%	80%
5	Associate	Mediawiki	ORAC-DR	100%	89%
6	Associate	Mediawiki	ORAC-DR	100%	78%

3.3 Development effort distribution

Table 4 and Table 5 show the development effort with respect to the number of lines of code modified and the number of revisions in ORAC-DR project and Mediawiki project respectively.

Table 4. The project effort of ORAC-DR

Members	Number of lines of code modified (KLOC)	Thousand number of revisions
Core (5)	233.068	10.722
Associate (9)	32.982	1.480

Table 5. The project effort of Mediawiki

Members (G1=Group 1) (G2=Group 2)		Number of lines of code modified (KLOC)	Thousand number of revisions
G1	Core (6)	516.440	17.602
	Associate (34)	75.836	2.585
G2	Core (5)	133.471	4.549
	Associate (10)	29.576	1.008

Figure 3 shows the percentage distribution of development effort of core members and associate members in the two projects. In all four cases, it can be seen that core members are responsible for over eighty percent of the development effort, while associate members are responsible for less than twenty percent.

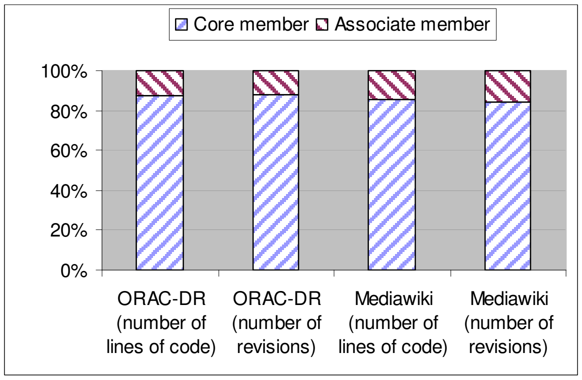

Figure 3. Percentage development effort by core member and associate member in two projects

To study the similarities or differences among the four distributions statistically, we presented the following two null hypotheses.

- *H_{01}: There is no significant difference of the distributions of the project effort in terms of KLOC modified by core members and associate members in ORAC-DR project and Mediawiki project.*
- *H_{02}: There is no significant difference of the distributions of the project effort in terms of thousand revisions by core members and associate members in ORAC-DR project and Mediawiki project.*

The obvious way to test these hypotheses is to apply the chi-square test. We construct two 2×2 contingency tables based on the data shown in Table 5 and Table 6. The results are that in both two tests, the significance (p-value) is greater than 0.05; we can not reject the null hypotheses and conclude that there is no significant difference of the distributions of the project effort (in terms of thousand of lines of code modified and thousand of revisions) by core members and associate members in ORAC-DR project and Mediawiki project.

4. Conclusions and future work

This paper presented a model to represent the interactions of software developers. Case studies were performed on ORAC-DR project and Mediawiki project using this approach. The study shows that the interaction model combined with the data mining techniques is an effective way to study the developer organization, especially the developer roles of distributed open-source software development. The model can be refined to take into account several limitations such as examining context to disregard routine or non-development work focused interactions.

5. References

[1] C. Reis, and R. Fortes, "An Overview of the Software Engineering Process and Tools in the Mozilla Project", *Proceedings of Workshop on Open Source Software Development*, Newcastle, UK, February, 2002.

[2] C. Jensen, W. Scacchi, "Collaboration, Leadership, and Conflict Negotiation in the NetBeans.org Community", *Proceedings of the 4th Workshop on Open Source Software Engineering*, Edinburgh, UK, May 2004.

[3] J.A. Espinosa, R.E. Kraut, S.A. Slaughter, J.F. Lerch, J.D. Herbsleb, A. Mockus, "Shared Mental Models, Familiarity, and Coordination: A Multi-Method Study of Distributed Software Teams", *Proceedings of the 23rd International Conference on Information Systems*, December, 2002, Barcelona, Spain, 425–433.

[4] M. Elliott, W. Scacchi, "Free Software Developers as an Occupational Community: Resolving Conflicts and Fostering Collaboration", *Proceedings of the ACM International Conference on Supporting Group Work*, Sanibel Island, FL, November 2003, pp. 21–30.

[5] W. Scacchi "Socio-Technical Interaction Networks in Free/Open Source Software Development Processes", *Software Process Modeling*, Springer, New York, 2005, pp. 1–27.

[6] G. Madey, V. Freeh, R. Tynan, "Modeling the F/OSS Community: A Quantative Investigation", *Free/Open Source Software Development*, Idea Group Publishing, Hershey, PA, 2004, pp. 203–221.

[7] A.K. Jain, M. N. Murty, and P.J. Flynn, "Data Clustering: A Review", *ACM Computing Surveys*, vol. 31, no. 3, 1999, pp. 264–323.

[8] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publishers, 2001.

[9] ORAC-DR project, http://www.oracdr.org/

[10] http://www.mediawiki.org-/wiki/MediaWiki