Mining Roles Structure of OSS projects in SourceForge

Lin Yuan, Huaimin Wang, Gang Yin, Dianxi Shi, Yanxu Zhu

School of Computer, National University of Defense Technology, fkefss@gmail.com

Abstract

Open Source Software (OSS) evaluation uses various metrics to compute the scores or levels of OSS. However the metrics used in existing evaluation models are difficult to rate and costly to obtain, and the rates of metrics are often tendentiously subjective and inconvincible. This paper tries to find some kinds of data in OSS repositories which can be easily obtained and used as metrics for more practical OSS evaluation. By mining and analyzing nearly 8,000 SourceForge projects, this paper achieves three interesting observations: (1) The Pearson correlation coefficients between the role number and the rank of projects are surprisingly high which means they are strongly related; (2) the centralities of roles reveal a irradiative deduction that the projects with higher rank usually tend to have more rational and balanced role structure; (3) the centralities and correlation of roles provide valuable clues for constructing a reference role structure model. These results show that the role structure in OSS projects has a great influence on their quality and provide a new dimension of metrics for more automatic and practical OSS evaluation.

Keywords*: open source software, data mining, social network, software repository*

1. Introduction

Open source software is increasingly considered and adopted for business purposes. OSS provides free and vast software resources ranging from low-quality individual efforts to high-quality enterprise solutions. Deciding which software package to adopt is always a challenging and costly task because there are so many candidates and evaluation factors, such as issues of compatibility, usability, scalability, and even legality. For example, SourceForge contains over 230,000 open source projects with source codes, documents and various resources related to the projects.

Different approaches have been presented to evaluate OSS projects. The Open Source Maturity Model developed by Bernard Golden of Navicasoft [1] and the CapGemini Open Source Maturity Model [2] aim to offer a methodology for assessing and evaluating the suitability of open source software. The Center for Open Source Investigation at Carnegie Mellon West, and Intel Corporation have developed the Business Readiness Rating model (BRR) [3], which lets IT managers quickly make informed and educated decisions about open source software. Atos Origin Corporation conceived and formalized the method for Qualification and Selection of Open Source software (QSOS) [4].

In the process of evaluating OSS projects, many metrics in these models are not only hard to obtain, but also difficultly to score even after they had been obtained, such as end-user UI experience, quality of professional support, number of books in Amazon, leading team, training, documentations, and so on. Therefore it is very important to find some evidence in software repository, which would be obtained easily and quantified convincingly.

After comprehensively studies of nearly 8,000 SourceForge projects, we find that, generally the projects with higher rank are more inclined to have special roles to undertake different works. Therefore , role structure of an OSS project not only represents the reasonable degree of labor division, but also reflects the quality of some properties to a certain extent. In most of OSS communities, the participators and roles information of projects can be obtained easily. Then, if the participators and role structure of a project can reflect some properties of OSS projects truly and effectively, could we take them as evidence in OSS evaluation, or even as alternatives of some existing metrics which were obtained difficultly and used unconvincingly? By mining the data of roles and participators in the projects selected from SourceForge, the above question is discussed in this paper. We hope our conclusion will be helpful to evaluate OSS projects quantitatively, effectively, and automatically.

The rest of the paper is structured as follows. Section 2 describes related conceptions and data collection. Section 3 evaluates the Pearson correlation coefficient between roles of project participators and project rank in SourceForge. Section 4 presents the centralities of roles and the layered role structure of projects. Section 5 discusses the results and presents a conjecture. Section 6 mentions limitations of our work, and the future work. Section 7 presents some related work and section 8 concludes.

2. Related conceptions

SourceForge is the largest OSS development web site in the world. By June 2009, more than 230,000 software projects have been registered by more than 2 million users. It provides large amounts of real data covering the entire software life cycle for academic research. The SourceForge data used in this paper is collected from the Flossmole web site [5].

There are 7 kinds of projects statuses in the SourceForge, which are planning, pre_alpha, alpla, beta, production/stable, mature, and inactive.

In SourceForge, the recommendation and rank of projects are determined by activity scores. It means the rank of project is actually one kind of quantization of project's recent activity, in generally, we are able to consider that the projects with higher ranks indicate that they have enjoyed higher popularity and won more widespread applications. So the rank can reflect the quality of projects to a certain degree.

In nearly 230,000 SourceForge projects, roles are used to distinguish different development tasks. After combining the roles which actually undertaking the same activities, finally we get 27 roles in all SourceForge projects, which are listed in Table 1:

TABLE 1. The Roles in SourceForge

Index	Role Name	Index	Role Name
1	Advisor/Mentor/Consultant	2	Cross Platform Development
3	All-Hands Person	4	Project Manager
5	Analysis/Design	6	Requirements Engineering
7	Build Engineer	8	Researcher
9	Content Management	10	Sr. Customer Support Manager
11	Database Administrator	12	Support Manager
13	Developer	14	Support Technician
15	Director of Operations	16	Tester
17	Distributor/Promoter	18	Translator (I18N/L10N)
19	Doc Writer	20	Systems Programmer/Analyst
21	Editorial/Content Writer	22	Unix Admin
23	Graphic/Other Designer	24	UI Designer
25	Packager(.rpm, .deb, etc)	26	VP of Media Operations

3. Relationship between role numbers and rank of projects

In our studies, the selection of data source relies on two factors: project status and its rank. We selected the projects which are beta, production/stable, or mature status, and before 10000 in rank list. There are total 7,779 projects meeting both conditions in SourceForge in Jun 2009.

3.1 Distribution of roles

The distribution of roles in SourceForge projects is calculated by:

$$role\_num[i] = \sum_{j=1}^{N} project_j[i] \qquad i \in [1, n]$$

$$role\_percent[i] = \frac{role\_num[i]}{\sum_{i=1}^{n} role\_num[i]}$$

Where N is the number of target projects, n is the number of roles; projectj[i] is the participators number of role i in the target project j; role_num[i] describes the total number of role i in all of target projects; role_percent[i] shows the percentage of the role i in total participators of all target projects.

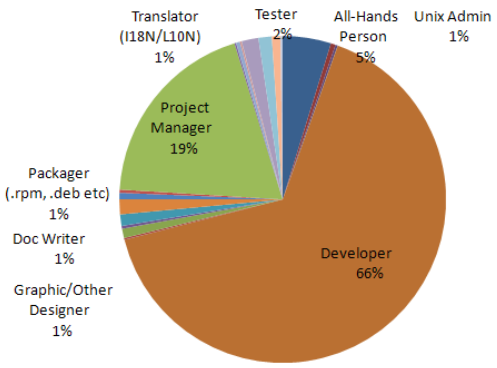

Figure 1. Percentages of roles in SourceForge projects

Fig.1 illustrates the proportions of each role in all target projects. It shows that the most participators are developers(66%), and the second are project managers(19%). The percentages of the other roles are much smaller and even many kinds of roles no more than 1%. Fig.1 indicates that the participators of OSS projects are mainly composed of developers and project managers.

3.2 The number of roles in projects

For further studying the relationship between role numbers and rank of projects, we divide the rank range from 1 to 10000 into 10 intervals averagely, and analyze the target projects in different intervals. Parts of the results are described in Fig.2 and Fig.3, and several observations can be concluded from them.

i. A quite number of projects in SourceForge only have one or two roles, and a very small proportion of projects own more than 7 kinds of roles;

ii. Comparing the different intervals (Fig.2 X-axis), in the rank range from 1 to 1000, the proportions of projects owning more than 3 roles are obviously higher than others intervals, and there is a downward tendency of role numbers in projects as the rank declines.

iii. Comparing the role numbers (Y-axis), the proportions of projects with 1 or 2 roles in the ranges [1,1000] in Fig.2 and [1,100] in Fig.3 are clearly lower than behind ranges; and the proportions of projects with more than 3 roles in front ranges are larger than behind.

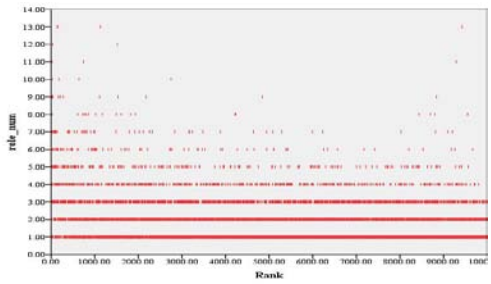

Figure 2. Role numbers of projects in rank range [1,10000]

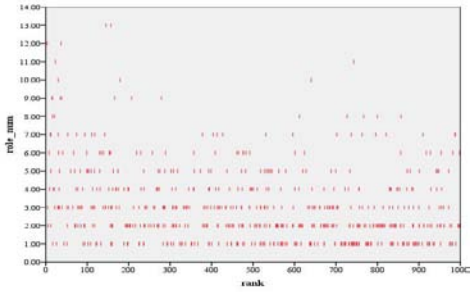

Figure 3. Role numbers of projects in rank range [1,1000]

For discovering the relationship between the number of roles and rank of projects, we further mined and analyzed the related data in the target software repositories.

3.3 Average number of roles in different rank intervals

Based on the situation that there are only few projects with more than 7 roles, we only take the projects with role numbers from 1 to 7 into account.

Fig.4 and Fig.5 illustrate the proportions of the projects with the same role numbers in different rank ranges. And we can find in both figures as the role number (X-axis) increases, proportions of the projects with higher rank increase too. It means that the projects with higher rank would have the tendency to own more roles. In other words,

we can believe the projects with higher rank usually have the better division of work than the projects with lower rank.

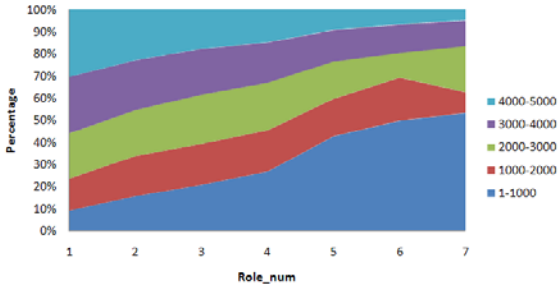

Figure 4. Distribution of projects based on role numbers in rank [1,5000]

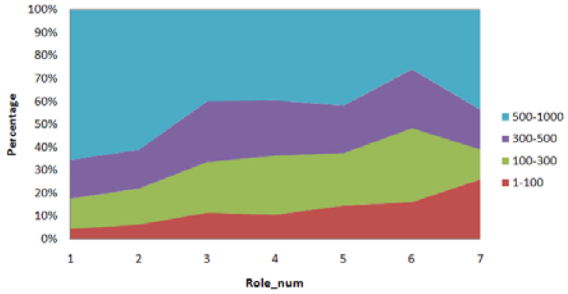

Figure 5. Distribution of projects based on role numbers in rank [1,1000]

We calculated the average number of roles in different rank intervals by the follow formula.

$$avg\_role\_num = \frac{\sum_{j=1}^{N_i} role\_num_j}{N_i}$$

Where N_i is the total number of projects in the *i-th* interval, $role\_num_j$ represents the role number of the project *j*. The results are described in Table 2 and Table 3.

TABLE 2. average role numbers in rank rang [1,1000]

Rank intervals	Project numbers	Average role numbers
0-100	64	4.64
100-200	65	3.86
200-300	51	3.20
300-400	60	2.81
400-500	58	3.24
500-600	62	2.61
600-700	62	2.73
700-800	64	2.61
800-900	62	2.44
900-1000	52	2.92

TABLE 3. average role numbers in rank rang [1,10000]

Rank intervals	Project numbers	Average role numbers
0-1000	645	2.70
1000-2000	708	1.91
2000-3000	878	1.70
3000-4000	1022	1.57
4000-5000	1097	1.46
5000-6000	1232	1.47
6000-7000	1241	1.46
7000-8000	1192	1.41
8000-9000	1182	1.48
9000-10000	1029	1.42

In order to observe the tendency of average role numbers in different intervals, Fig.6 and Fig.7 are given as follows.

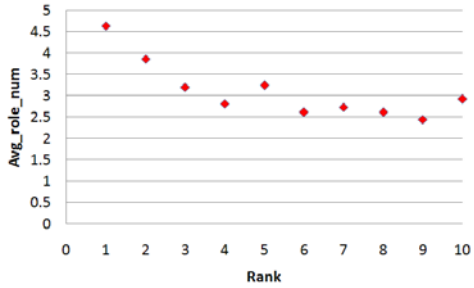

Figure 6. Average role numbers in the rank range [1,1000]

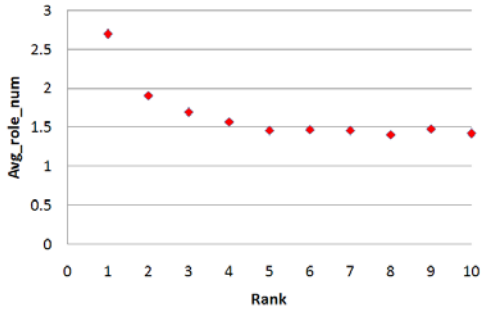

Figure 7. Average role numbers in the rank range [1,10000]

3.4 Pearson correlation coefficients

The Pearson correlation coefficient is a measure of the correlation between two variables, which is defined as:

$$corr(X,Y) = \frac{\sigma_{XY}}{\sigma_X \times \sigma_Y}$$

Where σ_X, σ_Y and σ_{XY} can be calculated as follows:

$$\sigma_{XY} = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})$$

$$\sigma_X = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2}$$

$$\sigma_Y = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(Y_i - \bar{Y})^2}$$

Where X and Y are the variables to be measured, $\bar{X}$ and $\bar{Y}$ are their means respectively; n is the number of samples. The Pearson correlation coefficient ranges from -1 to 1, and the detailed interpretation is shown in Table 4.

TABLE 4. meaning of Pearson correlation coefficent

Correlation	|corr(X,Y)|
None	0.0 to 0.09
Small	0.1 to 0.3
Medium	0.1 to 0.3
Large	0.5 to 1.0

Based on the data in Fig.6, the values in X-axis are taken as the samples of variable *X*, and the same to *Y*, then the correlation coefficient |corr1-1000| is calculated. Based on the data in Fig.7, we consider the different rank intervals, within the intervals from 1 to 5, the |corr1-5000| is calculated; within the intervals from 6 to 10, the |corr5000-10000| is calculated; and within the whole intervals, the |corr1-10000| is obtained. The results of Pearson correlation coefficients between the average role numbers and projects rank intervals are shown as follows:

|corr1-1000| = 0.790 |corr1-5000| = 0.902
|corr5000-10000| = 0.406 |corr1-10000| = 0.752

The correlation coefficient of rank range [1, 1000] shows there is strong correlation between role number and rank of projects. The coefficient of range [1, 5000] is even higher than 0.90, and further confirms the conclusion. Most of the projects in the range [5000, 10000] own only 1 or 2 roles, which cause the coefficient of this range is much lower than other intervals.

So we can infer that there is strong relationship between projects' role number and their rank. The projects with higher rank would be inclined to have more roles, especially in the rank range [1, 5000]. In other words, it means that the projects with higher rank would be organized better, and the work of these projects would be divided and assigned better. Based on the analyzing of Pearson correlation coefficients between the numbers of projects' roles and their rank, we can obtain the first Conclusion.

Conclusion 1: *The number of roles in an OSS project has a great influence on its rank.*

Ⅳ. Role structure of OSS projects

In order to retrieve the role structure model that can provide better support (e.g. helping projects to get higher rank) for the development of OSS projects in SourceForge, we calculated the absolute centralities and relative centralities of each role in different rank intervals.

4.1 Mathematical Models

The calculation method of role centrality is defined as follows. It can be described by absolute centrality and relative centrality.

$$absolute\_centrality[i, j] = \frac{\sum_{y=1}^{n} role\_matrix_i[j, y]}{project\_num_i}$$

$$relative\_centrality[i, j] = \frac{absolute\_centrality[i, j]}{\sum_{x=1}^{n} absolute\_centrality[i, x]}$$

Where the variable *n* is the number of major roles we considered, $role\_matrix_i$ is a matrix which describes the relation between roles, the variable *i* denotes the *i-th* interval, i∈ [1, 10]; $role\_matrix_i[x,y]$ describes the times of both role *x* and *y* occur in one same project in the *i-th* interval, and $role\_matrix_i[x,y] = 0$. Here, we don't take those projects with only one role into account; the variable project_numi is the number of projects which own two or more roles in the *i-th* interval; *absolute_centrality[i,j]* and *relative_centrality[i,j]* are respectively the absolute centrality and relative centrality of role *j* in the *i-th* interval.

The absolute centralities and relative centralities of major roles are shown in Table 5 and Table 6.

TABLE 5. absoulute centralities of major roles

Role	Rank intervals									
	[1,1000)	[1000-2000)	[2000-3000)	[3000-4000)	[4000-5000)	[5000-6000)	[6000-7000)	[7000-8000)	[8000-9000)	[9000-10000]
Developer	2.2636	1.7072	1.386	1.37	1.2874	1.3723	1.1437	1.3088	1.3324	1.2027
projectmanager	1.7841	1.313	1.0052	1.1314	0.9824	1.0951	0.8534	1.0294	1.1108	0.897
tester	0.7932	0.4464	0.2358	0.2279	0.2111	0.1902	0.0205	0.1294	0.1946	0.196
DocWriter	0.6932	0.3536	0.1839	0.1743	0.0997	0.1495	0.0381	0.1147	0.1324	0.1561
Translator	0.6932	0.3478	0.1554	0.1421	0.1378	0.106	0.0411	0.0853	0.0838	0.1694
WebDisigner	0.6409	0.3275	0.1865	0.1501	0.176	0.1413	0.0792	0.1618	0.2324	0.1063
Allhandsperson	0.6273	0.4261	0.3782	0.2681	0.3109	0.2527	0.2522	0.3059	0.327	0.3289
Packer	0.5795	0.258	0.1658	0.1903	0.1026	0.1793	0.1114	0.1294	0.1054	0.1163
Graphic	0.4386	0.1391	0.0751	0.0697	0.1437	0.0788	0.044	0.0382	0.1568	0.0764
advisor/Mentor/Consultant	0.3773	0.2174	0.1451	0.1019	0.0997	0.1603	0.1056	0.0765	0.1811	0.1329
porter	0.3	0.1884	0.0492	0.0885	0.0557	0.0543	0.0264	0.0471	0.0541	0.1163
EditorialContentWriter	0.2727	0.1101	0.0699	0.059	0.0059	0.0245	0.0088	0.0265	0.0811	0.0498
SupportManager	0.2614	0.1275	0.044	0.0697	0.0528	0.0625	0.044	0.0647	0.0514	0.0831
ContentManagement	0.2159	0.1333	0.0311	0.0429	0.0352	0.0707	0.0117	0.0412	0.0703	0.0432
SupportTechnician	0.2068	0.0899	0.0337	0.0161	0.0557	0.0082	0.0264	0.0324	0.0108	0.0399
UI	0.1591	0.0551	0.0233	0.0188	0.0381	0.0136	0.0059	0.0206	0.0216	0.0399
DistributorPromoter	0.1545	0.1188	0.0907	0.0241	0.0381	0.038	0.0059	0.0706	0.0432	0.093
analysisDesign	0.1159	0.0435	0.0311	0.0322	0.0059	0.0217	0.0088	0.0324	0.0324	0.0066
requirementEngineering	0.05	0.0435	0.0026	0.0107	0.0205	0.0082	0.0117	0.0147	0.0108	0.0399

TABLE 6. Relative centralities of major roles

Role	Rank intervals									
	[1,1000)	[1000-2000)	[2000-3000)	[3000-4000)	[4000-5000)	[5000-6000)	[6000-7000)	[7000-8000)	[8000-9000)	[9000-10000]
Developer	0.213	0.2646	0.3229	0.3271	0.3331	0.3389	0.4029	0.3509	0.3148	0.3089
projectmanager	0.1679	0.2035	0.2342	0.2702	0.2542	0.2705	0.3006	0.276	0.2625	0.2304
tester	0.0746	0.0692	0.0549	0.0544	0.0546	0.047	0.0072	0.0347	0.046	0.0503
DocWriter	0.0652	0.0548	0.0428	0.0416	0.0258	0.0369	0.0134	0.0308	0.0313	0.0401
Translator	0.0652	0.0539	0.0362	0.0339	0.0357	0.0262	0.0145	0.0229	0.0198	0.0435
WebDisigner	0.0603	0.0508	0.0435	0.0359	0.0455	0.0349	0.0279	0.0434	0.0549	0.0273
Allhandsperson	0.059	0.066	0.0881	0.064	0.0804	0.0624	0.0888	0.082	0.0773	0.0845
Packer	0.0545	0.04	0.0386	0.0455	0.0266	0.0443	0.0393	0.0347	0.0249	0.0299
Graphic	0.0413	0.0216	0.0175	0.0166	0.0372	0.0195	0.0155	0.0103	0.037	0.0196
advisor/Mentor/Consultant	0.0355	0.0337	0.0338	0.0243	0.0258	0.0396	0.0372	0.0205	0.0428	0.0341
porter	0.0282	0.0292	0.0115	0.0211	0.0144	0.0134	0.0093	0.0126	0.0128	0.0299
EditorialContentWriter	0.0257	0.0171	0.0163	0.0141	0.0015	0.006	0.0031	0.0071	0.0192	0.0128
SupportManager	0.0246	0.0198	0.0103	0.0166	0.0137	0.0154	0.0155	0.0174	0.0121	0.0213
ContentManagement	0.0203	0.0207	0.0072	0.0102	0.0091	0.0174	0.0041	0.011	0.0166	0.0111
SupportTechnician	0.0195	0.0139	0.0078	0.0038	0.0144	0.002	0.0093	0.0087	0.0026	0.0102
UI	0.015	0.0085	0.0054	0.0045	0.0099	0.0034	0.0021	0.0055	0.0051	0.0102
DistributorPromoter	0.0145	0.0184	0.0211	0.0058	0.0099	0.0094	0.0021	0.0189	0.0102	0.0239
analysisDesign	0.0109	0.0067	0.0072	0.0077	0.0015	0.0054	0.0031	0.0087	0.0077	0.0017
requirementEngineering	0.0047	0.0067	0.0006	0.0026	0.0053	0.002	0.0041	0.0039	0.0026	0.0102

4.2 Charactering relationship between roles

Based on the analysis of relationship between roles, we get the Fig.8 and Fig.9, in which the node is role, and the edge represents the relation between roles. The weight of edge in different rank ranges is defined as:

$$weight_{x,y} = role\_matrix_i[x, y]$$

If $weight_{x,y} \geq 30$, we consider it as strong relationship; if $15 \leq weight_{x,y} < 30$, we consider it as medium relationship; and if $1 \leq weight_{x,y} < 15$, we consider it as weak relationship. In Fig.8 and Fig.9, the thick solid lines, thin solid lines and dotted lines represent strong, medium and weak relationship respectively.

From Fig.8 and Fig.9, though only relationships in the first and second intervals are shown in this paper, they indicate very clearly as the rank declines, the relationships between roles tend to decline too. It can be interpreted as the degree of work division in OSS projects is unclearly more and more when their rank declines.

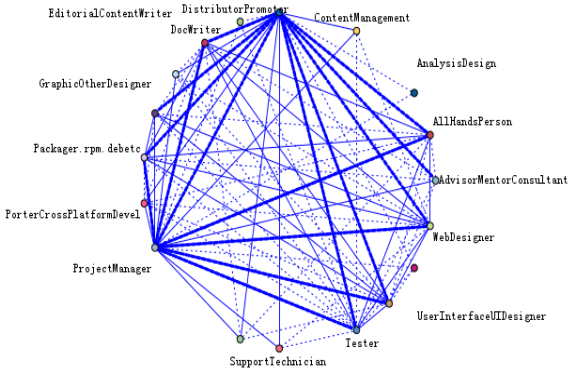

Figure 8. Relationships of major roles in the rank [1-1000]

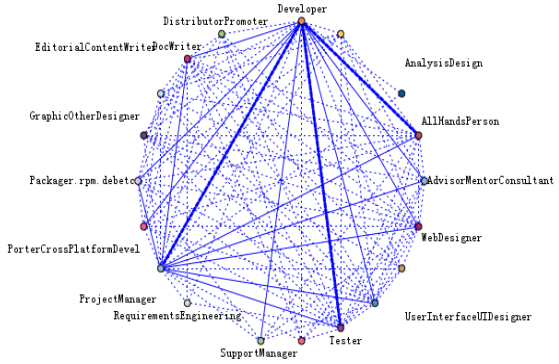

Figure 9. Relationships of major roles in the rank [1000-2000]

4.3 Relative centralities of role

In order to further characterize the role structure of OSS projects in SourecForge, we compare the relative centralities of some major roles in different rank intervals.

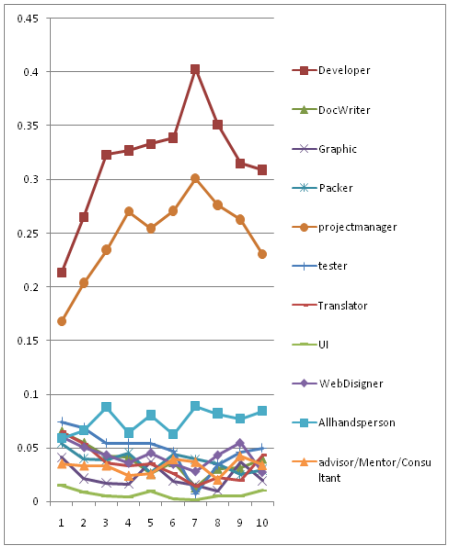

Figure 10. Relative centralitis of major roles in different intervals.

Fig.10 shows that: (1) For the OSS projects in SourceForge, the most important role is developer, and the second is project manager. (2) From the 1st to 7th rank interval, the centralities of both developer and project manager rise while most of the other roles decline. (3) With the rank declines, the centralities of all-hands person are obviously higher than other roles except the roles of developer and project manager. This phenomenon indicates that the clarity degree of work division in projects declines as the rank of projects declines. In other words, the development tasks are not properly divided and assigned to special participators in most of the projects with lower rank. Therefore, we can obtain the second Conclusion:

Conclusion 2: *In OSS projects, absence of some specific roles impacts the projects' quality to a certain extent, and the existence of some roles will be beneficial to projects evolution*

4.4 Role structure of OSS projects in SourceForge

According to the Fig.8 and Fig.10, there are some reasons to believe that the distribution of roles and the degree of project work division in the rank range [1, 1000] are the most reasonable and suitable for OSS development. Therefore, the role structure of OSS projects in SourceForge is constructed as Fig.11. Some of roles which occurred very few times are neglected.

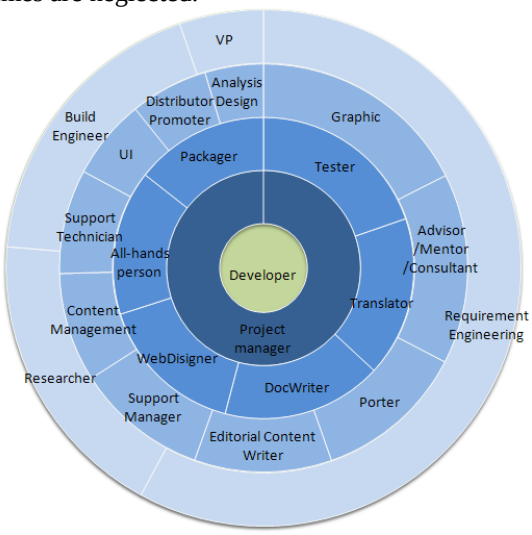

Figure 11. Role structure of OSS projeccts in SourceForge

According to the relative centralities of some major roles, the role structure is categorized into five layers:

Developer is the most fundamental and crucial role and placed at the core position;

Project manager, which is responsible for dividing project work, coordinating each role's, and pushing the project forward etc., is also the key role in OSS development and placed at the second layer;

The roles in the third layer, such as doc writer, translator, tester, packager, etc., are important roles in development of OSS software. They undertake the necessary work of OSS projects, such as projects' standardization, related documents, quality, or popularization. However, in some OSS projects, which are small in size and only own a few participators, these work may be undertook by all-hands people.

The roles in the fourth layer usually undertake some auxiliary projects work. These roles can help development of projects and improve project influence.

The centralities of roles in the outside layer are lower than others. Comparing with the other roles, we think they have only limited help to the developments of OSS projects.

By analysis of the role structure of OSS projects in SourceForge, we believe that the existing roles can reflect quality and characteristic of OSS projects to a certain extent. So we can obtain the third Conclusion.

Conclusion 3: *Analysis of the role structure in OSS projects conduces to OSS evaluation.*

5. Discussion

The purpose of our work is to find some evidence in software repository for OSS evaluation. The evidence, which will be used as the metrics of some properties in OSS evaluation, should be more objective and more easily retrieved. This paper focuses on the roles of OSS projects in SourceForge. By comparing the roles with the detailed metrics in BRR and QSOS models, we try to construct the connections between the roles and the properties of these models, which are showed in Fig.12.

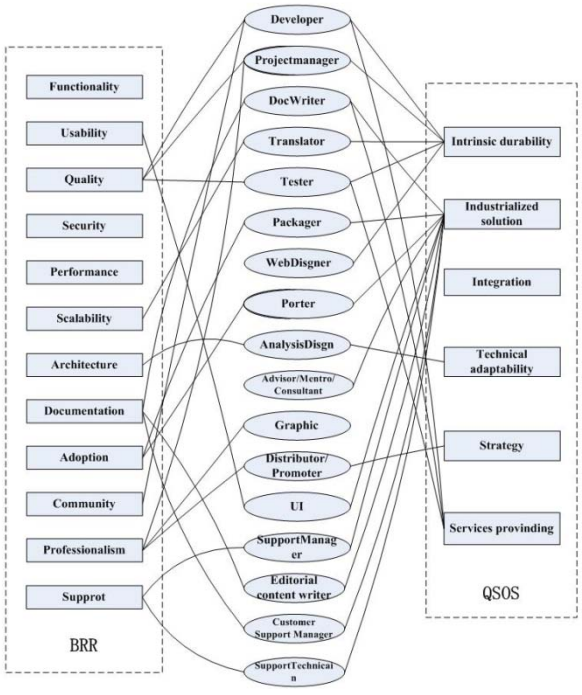

Figure 12. Roles and OSS evaluation properties

Fig.12 is a reference model of such connections that may be used to mapping the project roles to the properties (and thus metrics) in OSS evaluation models. This reference model is based on the following hypothesis:

Conjecture 1: *The information of role structure can improve or even replace some metrics in the existing evaluation models and will help evaluate OSS projects efficiently and automatically.*

6. Limitation and future work

We must admit that a successful OSS project may be only one participator; in fact, there are many projects in SourceForge with only 1 or 2 participator(s). Currently, our work doesn't take some important factors of OSS projects into account, such as project size, total number of participators, etc., and we will continue processing in-depth studying in the future.

In addition, the capabilities and contributions of the participators undertaking specific roles are usually different, and these will surely impact on the related properties too. In the future work, we will consider the abilities of every participator in projects, which can be calculated by one's historical behavior in the OSS communities.

7. Related work

In recent years, Software engineering has increasingly come to understand that there are predictable relationships between the structure of code and the social structure of the development team; better understanding of social structure can improve development planning. Academic research examining OSS development is rapidly growing and has begun to investigate the social structure of projects. Researchers have studied project size and action patterns, concentrating on code production and interaction between project members, but until now, we don't find any papers concerning the roles in OSS projects.

A noteworthy contribution in this field is the onion model [9], Mockus, etc. shows how developers and users are positioned in communities. The focus of their studies has largely been on the contribution of code, therefore they have largely discussed development centralization. In this model, it is possible to differentiate among core developers (those who have a high involvement in the project), co-developers (with specific but frequent contributions), active users (contributing only occasionally) and passive users. At the center of the onion are the core developers, who contribute most of the code and oversee the design and evolution of the project. In the next out layer are the co-developers who submit patches (e.g., bug fixes) which are reviewed and checked in by core developers. Further out are the active users who do not contribute code but provide use-cases and bug-reports as well as testing new releases. Further out still, and with a virtually unknowable boundary, are the passive users of the software who do not speak on the project's lists or forums.

Kevin Crowston and James Howison address the question of whether OSS projects exhibit consistency in their social structure [10]. They chose to examine the network centrality of communication during the bug-fixing process, and found that the centralizations are negatively correlated with number of developers and active users who contributed to the bug reports, and the centralization of OSS projects engaged in bug-fixing is in fact widely distributed, with a few highly centralized projects, a few decentralized and most somewhere in the middle (the distribution cannot be distinguished from a normal distribution). Their study demonstrates that a particular pattern of communications centralization or decentralization is not a characteristic of OSS projects when engaged in the task of bug-fixing.

Gregorio Robles shows how to better understand the evolution of the most active group of developers contributing to an OSS software project in [11]. They believe the stability and permanence of this group of most active developers is of great importance for the evolution and sustainability of the project. The "code gods" scenario means that the activity of the different core teams is high over their whole history, and the "code gods" scenario means the composition of the core group changes seldom. An important open problem was discussed in their work, to which extent other, non-coding activities (such as discussion, writing of documentation, or even mediation between developers) should be considered to better identify the core team of developers.

8. Conclusion

In this paper, the relationship between roles and projects rank in SourceForge community are mined and studied. The roles used in SourceForge are wholly examined and finally a reference role structure of OSS projects is proposed based on 3 conclusions: (1) **The number of roles in an OSS project has a great influence on its rank;** (2) **In OSS projects, absence of some specific roles impacts the projects' quality to a certain extent, and the existence of these roles will be beneficial to projects evolution;** (3) **Analysis of the role structure in OSS projects conduces to OSS evaluation.**

Based on the work in this paper, we believe that the data on roles and participators in OSS projects can be used to improve or even replace some metrics in the existing evaluation models, and will help evaluate OSS projects efficiently and automatically. We also believe that this paper initiates the research on evaluation and health detection of OSS projects by using the roles, tasks and contributions of projects participators, and thus opens a window towards more automatic and practical OSS evaluation.

9. Acknowledgement

This work was partially funded by National High-Tech Research and Development Plan of China under Grant (2007AA010301), National Natural Science Foundation of China under Grant (No.60903043), and the “ Core electronic devices, high-end general chip and fundamental software” Major Project (2009ZX01043-001-04).

10. References

[1] Golden's OSMM, http://www.navicasoft.com/pages/
[2] CapGemini's OSMM. http://www.seriouslyopen.org/nuke/html
[3] OpenBrr, Business Readiness Rating for Open Source (BRR2005). http://www.openbrr.org/BRR2005.pdf
[4] Atos Origin, Method for Qualification and Selection of Open Source software (QSOS), version1.6. http://www. qsos .org/download/qsos-1.6-en.pdf
[5] Flossmole, http://www.flossmole.com
[6] TRUSTIE-STC, Software Trustworthiness Evidence Framework Specification (V2.0), http://www.trustie.net, 2009.5.
[7] TRUSTIE-STC, Software Trustworthiness Classification Specification (V2.0), http://www. trustie.net , 2009.5.
[8] WANG Huaimin, TANG Yangbin, YIN Gang, LI Lei. Trustworthiness of Internet-based software. Science in china Series F: Information Sciences, vol.49, no.6, 759-773, 2006.
[9] Mockus, R. T. Fielding, and J. D. Herbsleb. Two case studies of Open Source software development: Apache and Mozilla . ACM Transactions on Software Engineering and Methodology, vol. 11, no. 3, pp. 309–346, 2002
[10] Kevin Crowston, James Howison, The social structure of open source software development. First Monday, 2005,Vol.10, No.2
[11] Gregorio Robles, Jesus M. Gonzalez-Barahona, Israel Herraiz. Evolution of the core team of developers in libre software projects. In Proceedings of the International Workshop on Mining Software Repositories (MSR), pages 167–171, 2009.
[12] Megan Conklin, James Howison, and Kevin Crowston, 2005. Collaboration Using OSSmole: A repository of FLOSS data and analyses. In Proceedings of the Mining Software Repositories Workshop of the International Conference on Software Engineering (ICSE 2005). St. Louis, MO, USA. May 17, 2005
[13] James Howison and Kevin Crowston, 2004. The Perils and Pitfalls of Mining Sourceforge. In Proceedings of the Mining Software Repositories Workshop at the International Conference on Software Engineering (ICSE 2004). Edinburgh, Scotland.
[14] T. Dinh-Trong and J. M. Bieman, “Open source software development: A case study of freebsd,” in Proceedings of the 10th International Software Metrics Symposium, Chicago, IL, USA, 2004
[15] D. E. Perry, N. Staudenmayer, and L. G. Votta. People, organizations, and process improvement. IEEE Softw., 1994.
[16] P. M. Johnson, H. Kou, M. G. Paulding, Q. Zhang, A. Kagawa, and T. Yamashita. Improving software development management through software project telemetry. In IEEE Software, 2005