OSSEAN: Mining Crowd Wisdom in Open Source Communities

Gang Yin, Tao Wang, Huaimin Wang, Qiang Fan, Yang Zhang, Yue Yu, Cheng Yang
National Laboratory of Parallel and Distributed Computing
School of Computer, National University of Defense Technology
ChangSha, Hunan, China
yingang@nudt.edu.cn, taowang.2005@outlook.com, whm_w@163.com

***Abstract*—Nowadays open source software represents a successful crowd-based software production model and is becoming an ecosystem combining huge amounts of software producers (such as software developers) and consumers (such as software users and customers). Lots of research work has been conducted on analyzing software artifacts created by producers, but few of them reveal the power of feedback from consumers which we believe is very important for the evaluation and evolution of open source software. This paper introduces OSSEAN, a platform for Open Source Software Evaluating, Analyzing and Networking. OSSEAN divides the open source communities into two groups: software production communities and software consumption communities. The former contain structured software artifacts such as projects, source code and issues, while the latter are full of textual documents with rich semantics of user feedback. We show the power of OSSEAN with some interesting demos by analyzing more than 200 thousands of open source projects and 10 million documents.**

***Keywords*—Open Source; Crowd Wisdom; Software Production Communities, Software Consumption Communities; OSSEAN**

I. INTRODUCTION

Open source communities successfully leverage the power of the crowd in software production, and have great impacts on many stages of software development and applications in a global open source ecosystem. With the development and division of open source communities, software programmers and testers are attracted by software *production communities* such as SourceForge and Github, while the newcomers, users and customers are more likely going to software *consumption communities* such as StackOverflow, Slashdot and CSDN (http://www.csdn.net, the biggest IT community site for Chinese speaking users in the world). These two kinds of communities complement each other and greatly expand the scope of traditional software development activities to global software evaluation and evolution.

The production communities mainly help software developers manage their development processes and artifacts. For example, Github provides development tools such as version control and issue tracking, and social communication tools such as @mention [1]. SourceForge provides more complete toolkits for distributed collaboration and management, including mailing lists, feature request, etc. These tools store huge amounts of software engineering data for structured software artifacts. On the other hand, the consumption communities are usually the crowd-oriented knowledge sharing platforms attracting tens of millions of users. The data generated in in these communities are usually textual posts which are reacting more quickly than the development requests submitted in production communities. For example, StackOverflow has an answer rate above 90% and a median answer time of only 11 minutes [2], while the average responding time for an issue in Android issue tracking system (a typical production community) is about 31 days. The consumption communities are becoming the source of the crowd wisdom for the evaluation and evolution of open source software in production communities.

We propose a new approach, OSSEAN (Open Source Software Evaluating, Analyzing and Networking), to leverage the crowd wisdom in consumption communities to support the software development in production communities. OSSEAN is composed of two steps: firstly collects the set of documents in consumption communities for each software in production communities, then use the documents to make evaluation, comparison and ranking for the software. According to our experiments, OSSEAN successfully discovers more than 8 million documents for more than 238 thousands projects. In this paper, we use three promising demos to show the potential applications of OSSEAN.

The structure of this paper is as follows. In Section II some closely related systems are introduced and discussed. In Section III key mechanisms of our approach are described. In Section IV, we show some preliminary but promising demos of OSSEAN. We conclude the paper in Section V.

II. RELATED WORK

A lot of research work has been conducted on collecting and mining data in open source communities.

The large amounts of high-quality source code publicly available over the internet have attracted great attention from researchers. Sushil et. al constructed an Internet-scale software repository Sourceer [7]. It employs the structural information like reference in source code, the dependences between libraries and so on to achieve large scale source code indexing and searching. OCEAN [8] present a federated search engine that simultaneously retrieves source code from existing open source code search engine sites including Koders, Krugle, Merobase and Google Code.

In industry, many analysis tools and services have been provided by different companies. Coverity Scan [9] mainly focus on analyzing the quality and security of open source software by providing scan and test services. Converity Scan helps developers to identify critical quality and security defects that are hard to find by other methods, and can provide valuable information for users to locate and fix the identified defects. Until March 2014, the lines of source code they scan have reached 300 million. Many famous open source projects

such as FreeBSD, HBase and CloudStack benefit much from this service.

As a famous open source software management solution provider, Black Duck [10] has built a massive open source knowledge base named BlackDuck KnowledgeBase. It collects more than 1 million open source software from 7.5 thousands open source sites. In this knowledge base, they cover more than 100 million lines of source code and more than 2 thousands of different licenses. They build code search engine Koders and open source community OpenHub.

Most of these works focus on the static source code of open source software, and little attention has been paid on the user feedback in consumption communities.

III. THE OSSEAN PLATFORM

To make better use of the data generated in open source ecosystems, we design and implement OSSEAN—an Open Source Software Evaluating, Analyzing and Networking platform. OSSEAN aims to bridge the production communities and consumption communities to retrieve the knowledge hidden among different communities, which are very important for software evaluation and evolution in Internet. In this section we introduce some important design considerations and the data source of the platform.

A. Architecture

The OSSEAN architecture mainly consists of three layers: acquisition layer, analysis layer and presentation layer. Through the cooperation of the three parts, OSSEAN achieves the automatic data processing and mining for multiple production communities and consumption communities. Figure 1 presents an overview of the architecture.

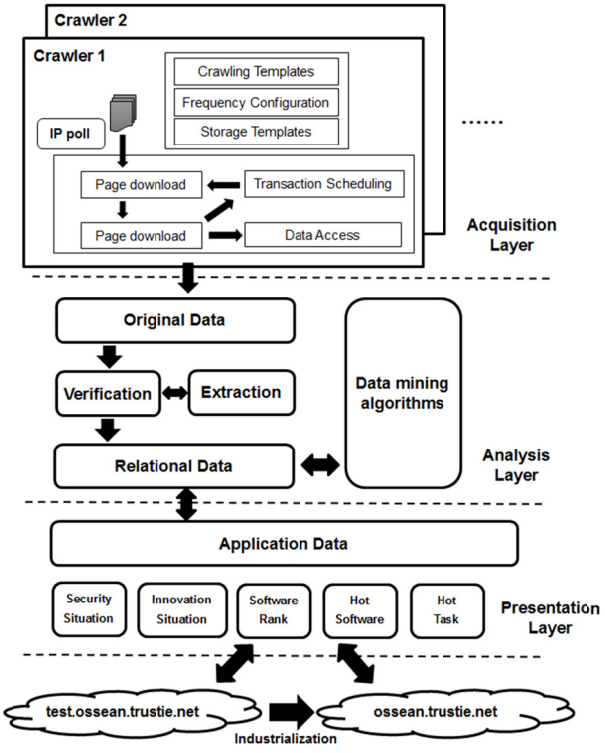

Figure 1: The architecture of OSSEAN

OSSEAN cuts the correlation between different modules in the three layers to a fullest extent. The communication between different modules is achieved through event service and database operations, so that each module can be developed and deployed by independent teams. Each module retrieves its input data from database and then notifies the next module by sending a specified event, with a relatively stable interface between each other. All modules in OSSEAN cooperate as in an assembly line.

The crawler module in acquisition layer is the engine of OSSEAN by crawling raw web pages from the two kinds of communities. A crawler will be created and deployed when a community is added as a new data source. Each crawler monitors a specified community site and retrieves data (raw web pages or API data) according to the predefined data model. Finally, the data crawled by crawler will wait for being analyzed by data extraction module. Here, the wrong data will be moved and the missing data will be re-crawled. Then, the data analysis module will construct the mappings between software projects and documents, and use the result to evaluate each project, such as analyzing the trend of projects and ranking them by popularity.

The modules in presentation layer are mainly designed for demonstration of the above results, including operations such as searching and ranking. Each module obtains data from the database schema of previous module, and saves result in its own schema. The results of data analysis will be placed in the data pool, prepared for presenting in platform. In order to provide a high quality data service, we have two online systems for OSSEAN: one is a testing system, and the other is a production system.

B. Date Source

OSSEAN collects data from both production communities and consumption communities. The first problem we met is what communities should be selected as data sources. Take development communities as example, Wikipedia lists more than 20 open source hosting websites such as SourceForge and Github, and there are thousands of community websites for interdependent open source organizations such as Linux, Apache, Eclipse, OW2, etc. As to consumption communities, there are far more candidates for consideration, such as StackOverflow, Slashdot, CSDN and OsChina (the last two are famous in China). We rank the communities according to their popularity.

(1) Data source from production communities:

We select two of the most popular development communities (SourceForge and GitHub) and one directory community (OpenHub). The number of projects and Alexa rank are shown in table 1. GitHub is a repository hosting community. It hosts 10,392,468 repositories up to August 2014. However, usually a software project can be forked many times in Github, and therefore a lot of repositories can be created for one project. So we use the number of root repositories as the number of projects in Github.

Among the total projects in these communities, there are lots of projects that are created just for a try of the community tools, these projects have few value and are the main cause of disturbance in later data processing. Another data disturbance is the duplicates of projects among different communities. To make sure that only active and useful projects are selected as

data source, we implement the following filtering polices: (1) remove the projects that either of its number of stars or downloads is zero; (2) rank the root repositories by their scores in Github and chose the first 116,850, where the score of each repository is the sum of its forks, watchers and commits in Github; (3) rank the projects by their scores in OpenHub and chose the first 98,833, where the score of each repository is the sum of its number of users, reviews and contributors in OpenHub. The statistical results are shown in table 1.

TABLE I. SOFTWARE PRODUCTION COMMUNITIES

	Site Name	*Total Projects*	*Selected Projects*	*Alexa Popularity Ranking*
1	GitHub	3,099,089	116,850	98
2	SourceForge	411,716	48,712	217
3	OpenHub	664,620	98,833	28,419
Total		4,175,425	264,395	—
Without Duplicates		—	238,467	—

(2) Data source from consumption communities:

There are much more consumption communities than production communities in Internet. We select most popular and hottest communities as our data source, mainly referring to their Alexa Rank (a websites ranking system provided by alexa.com according to the frequency of visits). Table 2 shows the number of posts collected from popular communities up to December 2014, including some famous Chinese communities. The posts are various kinds of documents generated in these communities for Q&A, discussion and sharing among users.

TABLE II. SOFTWARE CONSUMPTION COMMUNITIES

	Sites	*Posts*	*Alexa Popularity Ranking*
1	http://stackoverflow.com/	7,205,198	59
2	http://www.csdn.net/	848,998	442
3	http://www.cnblogs.com/	4140	1,241
4	http://www.codeproject.com/	187,692	1,346
5	http://slashdot.org/	45,203	1,484
6	http://www.oschina.net/	382,200	1,632
7	http://www.iteye.com/	92,545	2,441
8	http://www.dewen.io/	16,229	159,508
9	http://www.zdnet.com/	43,644	18,400
10	http://www.lupaworld.com/	29,541	210,871
Total		8,855,390	—

C. Data Acquisition

The two kinds of communities are publicly available, but distributed dispersedly over the whole Internet. It is a great challenge to retrieve such kind of data continuously and automatically. The task of acquisition layer is obtaining data from these community sites. OSSEAN uses APIs provided by community web site and web crawlers to construct a data flow storage and management platform. For data crawling, it mainly consists of two interdependent processes: firstly crawls the raw web pages, and then extracts the attributes from the web pages. We know that most data collecting platform extracts the page information simultaneously while downloads web pages, but this is not the good solution for the changing and growing sites.

This design is necessary for the insurance of data quality, which is very important for the retrieval of open-source data. We cannot identify whether an error is created in the process of crawling or extraction when extracted information is wrong. Because of this, the conventional methods may introduce a lot of low-quality data and data checking task is very difficult, and thus easily bring a lot of problems to the next stage of data analyzing. In order to improve the quality of acquired data, we add a new verification step into extracting process. The following are the details about these three processes.

Crawling process: A crawler in OSSEAN uses target-oriented retrieval policies. This process contains two steps. A crawler firstly crawls all the lists from a community and extracts the links of every item (here an item is an open source project). Then, the target pages referred by these links are downloaded and stored in database. Here we use multi-threaded techniques to improve performance. However, in order to prevent disturbing the performance of target communities, we set our crawling speed to a very low level.

Extracting process: This process extracts key attributes from raw web pages. Usually, different community sites have totally different page templates. Thus, we design corresponding extraction templates separately for each site. OSSEAN will automatically select right template to extract the input page according to its "site" name. The extracted attributes will be stored in database (here we use MySQL to stall these structured data).

Verifying process: This process verifies the potential errors exist in the crawling and extracting process. Verification of the extracted attributes will be invoked just after all pages are extracted. Different attributes have different verification methods. For example, a "time" related attribute (such as the "publish time" of a post) will be checked whether it has a right time format; while for the title of a post, it will be checked whether it is a valid textual string. In this process, the error handing module of each site will also locate the position of the error, either in download module or extraction module. The verification results will be logged for providing further bug-fixing instructions.

OSSEAN provides a plugin-based structure for adding new communities. The code framework for the three processes will be created and developers need only to change the extraction templates and verification templates.

D. Data Bridging

OSSEAN builds connections between production communities and consumption communities. Bridging two kinds of communities can be seen as a classification of posts to software project names, but we cannot use the conventional method to complete this classification. First, there is no training set for us to train a classification model, and clustering algorithm has the problem of discerning the body of each class. Second, there are many projects germinating every day, which makes it hard to build a stable model. Therefore, we use text matching method to solve this problem. For each post, we consider three attributes – tags, title and content. OSSEAN performs the matching in three steps – data preprocess, index building and entity matching. The following is the detail about the three steps.

Data preprocess: This step integrates the data from different communities. The data from different communities have different structure, so we define a standard structure for two kinds of communities. For example, the basic data structure of a post is a 13-tuple (subject, abstract, content, author, number of replies, number of views, type, category, created time, updated time, crawled time, URL, site), while that

of a project is a 13-tuple (name, description, language, number of followers, number of downloads, number of views, number of committers, category, created time, updated time, crawled time, URL, site).

For production communities, we also aggregate the multiple occurrence of a same project across different communities into one project.

Index Building: All the posts in OSSEAN are indexed according to their three attributes (tags, title and content). Each post is segmented and then its inverted index is created. Therefore, it will be very fast to answer queries such as whether a post contains a project name.

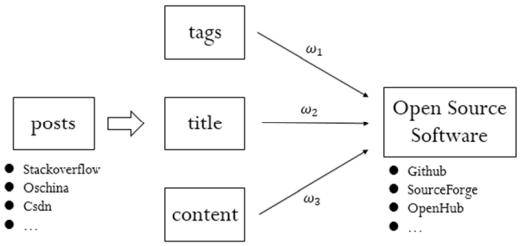

Figure 2. The flow of the matching process

Entity Matching: Given a post, each kind of its attributes is attached a weight ω_i, as shown in figure 2. OSSEAN uses string comparison to match tags matching, i.e., if the tag list of a post contains the software name then we get a positive match. OSSEAN uses index-based search for title and content matching, i.e., the name of the software will be used as a query term to search in the index. The score of the match is the sum of ω_i, where the match i should be successful. Currently, OSSEAN set ω_1, ω_2 and ω_3 to 1, 0.8 and 0.5 respectively.

Text matching method has many advantages in data bridging, but it also performs poorly in some cases. For example, it cannot confirm whether a term in a post refers to a software project even when the term is identical to the name of that project. Another case is that different projects may share similar or even same names. We are using some techniques to resolve these problems, such as Apriori and SVM.

IV. Preliminary Results

With the continuous growth of community data, OSSEAN can provide lots of interesting services by mining the crowd wisdom in open source communities. This section shows some preliminary results and demos.

A. Ecosystem Measurement

OSSEAN can answer some basic questions related to the scope and scale of the global open source ecosystem, such as: (1) How many open source organizations and software projects in production communities? (2) How many posts (or documents) in consumption communities are related to open source projects?

These questions are related to the global development of open source software ecosystem. As shown in figure 3, from 2008 to 2014, the number of organizations in Github goes up to the highest point 118845. Furthermore, we investigate the quantity of answer posts and question posts in StackOverflow. Both the number of answers and questions are increasing steadily to the highest points 12517744 and 7148960 as shown in figure 4. This indicates that the open source communities attract more and more organization as well as users to contribute and participate.

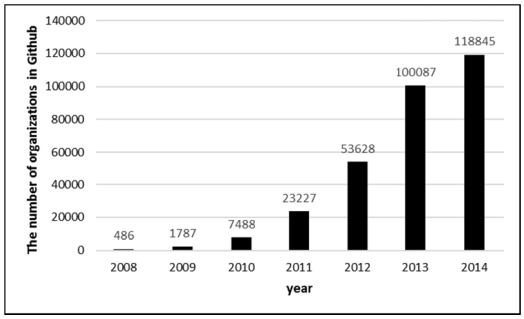

Figure 3: The number of organizations in Github in different years

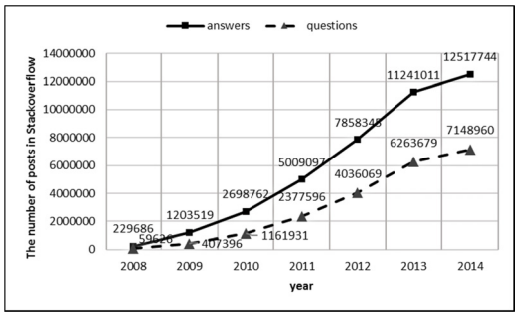

Figure4: The number of posts in StackOverflow in different years

B. Software Ranking

OSSEAN can help users find the favorable software based on the popularity of its related posts. Some key issues should be properly addressed, such as: (1) How to compute the popularity of a post? (2) What is the rank of a software project in the global open source ecosystem?

Compared with the traditional software ranking mechanism, we sort the software projects by the set of posts related to them. The basic idea is that, the more posts a software project successfully matches, the more popular this project is and should be sorted forward. In our observation, we find that different posts have different impacts. In order to distinguish these differences, we use the number of comments for judging the popularity of a post. We denote the popularity of a post by the score of this post. The following is the formula for computing the score of post p:

$$\begin{cases} Score(p) = comments(p) / timediff(p) \\ timediff(p) = time_{now}(p) - time_{open}(p) \end{cases}$$

Where $Score(p)$ is the score of post p. $comments(p)$ is the number of comments of post p. $timediff(p)$ is the time interval between post p is opened and now.

The rank of a software project is determined by the sum of scores of its posts. However, we find that in many cases one post may discusses more than one project. To avoid bias, we use the average score to calculate the software rank.

$$Rank(s) = \sum_{i=1}^{n} \frac{Score(p_i)}{software(p_i)}$$

Where $Rank(s)$ is the rank of project s. $software(p_i)$ is the number of software discussed in the post p_i. n is the number of posts associated with the software s.

Table III shows the top-10 software in the database category after our ranking process.

TABLE III. THE TOP-10 SOFTWARE IN THE DATABASE CATEGORY

Rank.	*Software*	*#Posts*	*Rank Value*
1	MySQL	289659	5793.18
2	PostgreSQL	33744	745.66
3	MongoDB	37283	710.48
4	SQLite	35524	674.88
5	cassandra	5381	205.84
6	Redis	5642	112.84
7	Solr	10292	107.62
8	Teradata	1044	55.64
9	Filemaker	575	20.88
10	habase	2782	11.50

C. Opinion Analysis

OSSEAN can further analyze the associated posts to find the opinion discussed about the software and answer some interesting questions related to the hot topics and development trend of the global open source communities, such as: (1) What are the hot topics of a software? (2) What is the current development trend in the global open source communities?

We utilize a clustering method to classify the words of posts into topics. This clustering method gives a similarity degree of association between words and topics. The method calculates a degree of similarity $sim(word_i,topic_j)$ between words $W=\{word_1,\ldots,word_i,\ldots,word_n\}$ and existing topic clusters $T=\{topic_1,\ldots,topic_j,\ldots,topic_n\}$. The word is classified into the maximized cluster $topic_j$ when $sim(word_i,topic_j)$ exceeds threshold τ. On the other hand, if $sim(word_i,topic_j)$ is less than threshold τ, we classify the word into a new cluster $topic_{k+1}$. The threshold τ is determined empirically. Finally, we can find the hot-topics in the discussion of the software.

As shown in figure 5, in the global open source communities, "Android" is the hottest topic, which indicates that more and more users or developers focus on the development of mobile software.

Figure 5: Hot topics in OSSEAN

More specifically, hot topics can be used to indicate the trend of user requirements and technical movements. There are lots of researches can be conducted in OSSEAN.

V. CONCLUSIONS

We propose a new approach, named OSSEAN, for mining software community data with a global vision in mind. OSSEAN treats open source communities as a global interconnected ecosystem. It divides Internet-based communities related to open source software into two groups: production communities and consumption communities. We believe OSSEAN will give more practical help to improve the quality and efficiency of global software engineering using data mining and machine learning tools in near future.

Currently, OSSEAN emphasizes on constructing and improving its data flow storage and management platform based on the two kinds of communities. More interesting demos powered by advanced data analysis algorithms will be launched in http://ossean.trustie.net.

ACKNOWLEDGMENT

Many thanks should be given to Trustie team members, especially to Chongming Gu, Chenxi Song, Yiang Gan, Ming Wu, Fang Zhang, Yun Zhan, Bingxun Liu and Ming Xiao for their valuable engineering work. We also want to thank reviewers for their valuable comments. The research is supported by the National High Technology Research and Development Program of China (Grant No. 2012AA011201) and National Science Foundation of China (Grant No.61432020 and 61472430).

REFERENCES

[1] Yue Yu, Huaimin Wang, Gang Yin, Charles X. Ling. Reviewer Recommender of Pull-Requests in GitHub. The 30th IEEE International Conference on Software Maintenance and Evolution (ICSME 2014 TOOLS), Victoria, Canada, 2014.

[2] Lena Mamykina1, Bella Manoim2, Manas Mittal3, George Hripcsak1, Björn Hartmann3. Design lessons from the fastest q&a site in the west. Proceedings of the SIGCHI conference on Human factors in computing systems. ACM, 2011: 2857-2866.

[3] M. Hahsler and S. Koch. Discussion of a large-scale open source data collection methodology. In HICSS, 2005.

[4] J. Howison, M. Conklin, and K. Crowston. Flossmole: A collaborative repository for floss research data and analyses. International Journal of Information Technology and Web Engineering, 1(3):17–26, 2006.

[5] Audris Mockus: Amassing and indexing a large sample of version control systems: Towards the census of public source code history. MSR 2009: 11-20.

[6] K. Inoue, R. Yokomori, H. Fujiwara, T. Yamamoto, M. Matsushita, and S. Kusumoto. Component rank: Relative significance rank for software component search. In ICSE'03, pages 14–24, 2003.

[7] Bajracharya S, Ossher J, Lopes C. Sourcerer: An internet-scale software repository. In Proceedings of the ICSE Workshop on Search-Driven Development-Users, Infrastructure, Tools and Evaluation. 2009: 1–4.

[8] Kokkoras F, Ntonas K, Kritikos A, Kakarontzas, G, Stamelos, I. Federated Search for Open Source Software Reuse. In Software Engineering and Advanced Applications (SEAA), 2012 38th EUROMICRO Conference on. 2012: 200–203

[9] Guo, Philip J., and Dawson R. Engler. "Linux Kernel Developer Responses to Static Analysis Bug Reports." USENIX Annual Technical Conference. 2009.

[10] Bagley C E, Lane D. Black Duck Software. Case Study. Harvard Business School Publishing, 2006.