Searching for Reputable Source Code on the Web

Rosalva E. Gallardo-Valencia
Department of Informatics
University of California, Irvine
Irvine, CA 92697, USA
rgallard@ics.uci.edu

Phitchayaphong Tantikul
Department of Informatics
University of California, Irvine
Irvine, CA 92697, USA
ptantiku@ics.uci.edu

Susan Elliott Sim
Department of Informatics
University of California, Irvine
Irvine, CA 92697, USA
ses@ics.uci.edu

ABSTRACT
Looking for source code on the Web is a common practice among software developers. Previous research has shown that developers use social cues over technical cues to evaluate source code candidates. However, current source code search engines do not take full advantage of social information. We present a prototype, an extension of Sourcerer, that shows reputation information for each developer involved in the project and the developer's activity level. From this implementation, we have learned that the effectiveness of this approach depends on the amount of reputation information available on the Web, which is currently scarce. We also learned that a ranking algorithm that relies on both relevance and reputation would be beneficial. More research needs to be done to explore ranking algorithms that combine both social and technical information and also reputation information for source code and people.

Categories and Subject Descriptors
H.5.3 [**Information Interfaces and Presentation**]: Group and Organization Interfaces – *web-based interaction.*

General Terms: Design, Human Factors.

Keywords: Reputation, Code Search on the Web

1. REPUTATION IN CODE SEARCH
Developers are using the Web as a giant repository of source code that can be used to solve their software development problems. Finding an appropriate piece of source code on the Web has an impact on how quickly and successfully development problems are solved. In our initial research, we found that when evaluating retrieved source code, developers made relevance judgments to create a short list of candidates for further investigation and they subsequently made suitability judgments to select a match to use in their project [1].

Previous research found that during the selection process, developers use social characteristics of the projects, such as level of activity and recommendations by peers over characteristics of the source code [2]. In a web survey [2], respondents indicated that they tend to value the opinion of "other people" in their final selection.

GROUP'10, November 7–10, 2010, Sanibel Island, Florida, USA.

When asked about what functionality developers would ideally like to have in a source code search engine they included characteristics as "review by other independent users," "user feedback," "reviews/ranking for the code," "other people satisfaction level," and "number of users actively using it."

This finding is consistent with Madanmohan and De' study [3] that suggests that developers must look at the names and number of people working in the source development because the participation of reputable developers makes the code less error prone and increases reliability. The study suggests that developers must also look at the quality of documentation, recent activity level in the repository, and number of stable code implementations. Chen et al. [4] also suggest that developers look for reputation of components and suppliers when they are evaluating components they want to use.

When developers are making a final selection of source code to include in their systems, one of the questions they ask themselves is: Do I trust enough this code to reuse it in my system? Based on previous research, to answer this question, developers are looking for other people's opinion about the candidate piece of software. In other words, developers are looking for the reputation information related to the source code.

We have implemented a prototype that collects reputation information for developers from Ohloh[1], a social networking site for developers, and shows this information along with the source code results presented by Sourcerer [5], a code search engine. For each match the prototype shows the authors of the source code including their KudoRank, overall ranking, experience, total number of commits, total lines changed, and comment ratio. The prototype also shows the number of users for a given project.

We identified three key challenges for our approach. The first one is that the effectiveness of our prototype relies on the reputation data found on Ohloh. We found that only 5% of the projects that Sourcerer hosts has reputation information. The second challenge we faced was how to effectively use the reputation information to rank the source code results. Finally, we need to better understand differences in the reputation of developer and the reputation of source code.

2. OHLOH AND REPUTATION
Recently, some social networking sites have appeared with the goal of connecting open source contributors. These social sites allow members of the community to rate their peers and also their projects.

[1] http://www.ohloh.net/

Ohloh is a social network for open source developers. The site lists around 19,500 projects. This site offers multiple services including search for projects, people, and forums, and also it allows comparing metrics for multiple languages and projects. It has an API available to access information in its repository. It computes statistics for each member of the community and contributors based on experience and contribution to open source projects.

Ohloh provides reputation information for projects in the form of reviews and number of downloads, as well as reputation for developers in the form of KudoRanks. People who write reviews or download open source projects are generally users of the project and not necessarily developers who want to reuse a piece of source code. Potential users of an open source project care about what other users think about the projects. In contrast, developers looking for a piece of source code to reuse in an open source project care about the reputation of the developers who contribute to the open source because they correlate good reputation of contributors with good quality of source code [3]. For this reason, we use the KudoRanks of contributors to calculate the reputation of the source code of an open source project.

Members of the Ohloh community can rank other members or contributors to show their appreciation or respect. This ranking is called Kudo. Ohloh members can send Kudos to as many as other members or contributors they want and they can also take back Kudos if desired. Ohloh distinguishes between members and contributors. An Ohloh member is a person who is registered as an Ohloh user. A contributor is a person who is a committer of a project but is not a member in Ohloh. Ohloh periodically computes the Kudos to calculate a number between 1-10 called KudoRank[2] . A KudoRank of 9 is rare and it is given to only 2% of the population. The KudoRank is influenced by the following facts: the more Kudos one receives, the higher one's KudoRank becomes; influence increases as KudoRank improves; giving away more Kudos dilutes a member's opinion, and "stacks" (projects added) improve KudoRank.

Although reputation information for open source projects is available on the Web, source code search engines such as Koders[3], Krugle[4], and Google Code Search[5] do not use this information to rank the results or to show them to the users. In this paper we present a prototype to bridge this gap by finding the reputation information related to a match of source code (at the contributor and project level). We include the reputation information in the ranking of the search results.

3. PROTOTYPE

We have implemented a prototype as a proof of concept by augmenting an existing code search engine, Sourcerer, with reputation information from Ohloh, a social network for open source developers and users. The goal of this prototype is to help us evaluate if our approach is possible and if it is possible, how beneficial it is.

3.1 Approach

Sourcerer [5] is a search engine for open source code that extracts fine-grained structural information from the code. It was developed at University of California, Irvine. It supports search for Java language source code. Sourcerer allows searching by components, functions, fingerprints, and all of the previous ones. For each type of search, it allows the user to enter keywords to search for.

Since the results of the search engine are pieces of source code that belong to open source projects, it would be helpful for the users to see not only some activity statistics about the project but also reputation information for the project and the authors of the project. For each match we will include some reputation information at the project level such as the average KudoRank and the average ranking from Ohloh. Both of these averages will be calculated based on their respective property of all the contributors of the project. KudoRank refers to an author reputation given by other members of the community. Ranking refers to the position the author will have with respect to his/her peers, given the KudoRank he has earned.

For each author we will show his/her reputation, namely, KudoRank and ranking, along with some activity information. This information will include the author experience in Java, the number of total commits, the total lines changed, and comment ratio for contributions on projects in Java.

We will show the user the reputation information, and use this information, specifically the average KudoRank per project, to re-rank the search results. Users will easily see in the results if an experienced programmer with good reputation has participated in writing the source code that the user is thinking about reusing. This would provide information to support users' decisions.

3.2 Ohloh Crawler and API Calls

Ohloh offers an API that is a REST-based programming interface to access Ohloh data. In order to use the API, we needed to register as members of Ohloh and register our application to get a key. Ohloh has 17 different functions available for the API. We used three of them (contributors for project, account info, and project info) to get information about the contributors (id, name, account id, and account name), its respective reputation (KudoRank and ranking), and project info (number of users and total number of lines of code). Some information such as the contributor activity (experience, number of total commits, number of total line changed, comment ratio) was not available using the Ohloh API. To get this information we developed a crawler, which goes to a specific page given some parameters such as the project name and contributor id and parses the html of the related page.

We used both API calls and the crawler to get KudoRank and ranking. The API has this information available but only for members of Ohloh and not for contributors. We used the API to get the KudoRank and ranking if the contributor was registered as an Ohloh member and we used the crawler if the contributor was not an Ohloh member. We could have decided just to crawl instead of checking if the user was or not a member, but we decided to use the API first when it was possible because the response time for the API is faster than the response time for the crawler. A call to the API to get the KudoRank and ranking (only one call) takes in average around 90 milliseconds and the crawler to get the KudoRank and ranking (two different pages) takes in average around 1,200 milliseconds.

[2] http://www.ohloh.net/about/kudos/

[3] http://www.koders.com/

[4] http://www.krugle.com/

[5] http://www.google.com/codesearch

We implemented three crawler functions. The first function gets the KudoRank given the project name and contributor id. The second one gets the ranking given the contributor id. The third one gets the experience, number of total commits, number of total lines changed, and comment ratio, given the project name and contributor id.

After getting the reputation information using both the API calls and the crawler, we stored the information in files whose data was later uploaded into the database. Currently, our database has 3,590 projects. From all these projects, only 244 have any information about contributors. We have information on a total of 1,908 contributors.

3.3 Presentation of Results

The version of Sourcerer that we extend originally shows the following information for each match: name of the entity (ie: method, variable), the rank number, the relations of this entity, and the fingerprints. It has the option to see the source code inline, in full page, also browse the project, download the entire entity or just a slice. It also shows project information such as the name, version, license, and category if available.

We augmented Sourcerer by adding the information that is marked in red rectangles and labeled with numbers in Figure 1. First, the results are shown to the user ranked by reputation. Thus, results whose contributors have a greater average KudoRank will be shown at the top. The user has the option to change the ranking to be only based on relevance. See 1 in Figure 1.

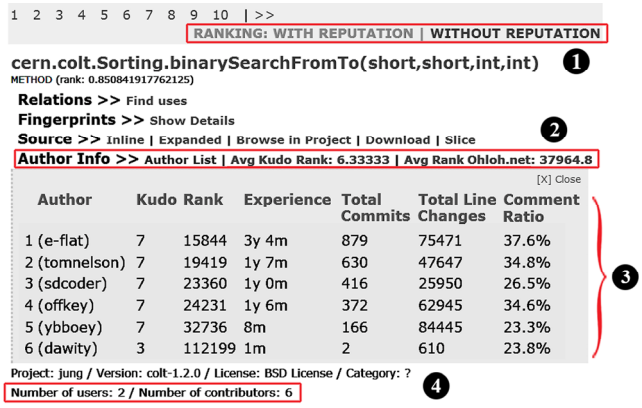

Figure 1. Reputation Code Search Application User Interface

Second, a new line for Author Info (see 2 in Figure 1) has been included for each match. This line includes a link to the Author List, the average KudoRank for all the contributors of the project, and the average ranking for all project contributors.

Third, when the user clicks on the Author List, the list of contributors to the project is shown (see 3 in Figure 1). For each contributor, the system shows the contributor's name, the user name in parenthesis, KudoRank, ranking, experience using Java, the total number of commits, the total number of line changes, and the comment ratio for projects in Java.

Finally, below the project information (see 4 in Figure 1), another line was added to include the number of users of the project and the total number of contributors. The number of users is a link to the Ohloh website that shows more information about the users of the project. The number of contributors includes a link to the Ohloh web page that lists the contributors for the projects and gives more information about them.

4. CHALLENGES AND OPPORTUNITIES

In this section, we discuss the challenges in implementing our approach to use reputation information in code search on the Web. We also discuss potential opportunities to face these challenges.

4.1 Availability of Reputation Information

Online reputation systems face many challenges including dealing with a small amount of reputation information due to the fact that people may not provide feedback. It is also difficult to ensure honest reports, to obtain feedback that is not extremely positive or negative, to track members of the community using different names, and to present reputation information that is useful for making decisions about whom to trust [6].

When augmenting Sourcerer with reputation information, we also faced some of these challenges. The most significant challenge we found was related to the low percentage of open source projects in Sourcerer that have reputation information in Ohloh. Sourcerer indexes 4,679 Java open source projects. We found 3,590 (77%) of these projects registered in Ohloh. However, only 244 projects (5%) had reputation information. The other 3,346 projects (72%) did not have information for contributors because a source code repository was not reported for these projects. In the end, we had reputation information only for 5% of projects in our repository.

Previously, an evaluation of Sourcerer was conducted using a set of 10 queries [7]. These queries were used because they have a reasonable number of hits that are large enough to have diversity of results and small enough to be manually analyzed. Also, the search intent of these queries is obvious and it is easy to agree on the relevance of the hits. Some examples of these hits are bounded buffer, quick sort, ftp server, and chat server. To evaluate the impact of the small quantity of reputation information available, we ran the same 10 queries using our prototype and calculated the percentage of matches with reputation information. We found that in average, 14% of the results for a query have reputation information.

The relatively small amount of data is a common problem in reputation systems in general. However, we have identified some opportunities to increase the amount of reputation data in our system. First, our current prototype is limited to the Java projects indexed by Sourcerer. We can explore implementing our approach using other code search engines that are not limited to a programming language and show source code from open source projects, such as Koders, Krugle, or Merobase[6]. Second, we can explore using other social sites to gather reputation information for developers such as Advogato[7] that certifies its users in three levels: Apprentice, Journeyer, and Master. Third, instead of using projects from a code search engine to look for their reputation information in social sites, as is currently done in our implementation, we can explore using a set of projects in a social site as a starting point to look for their repository information as suggested by Gysin [8]. The effectiveness of this approach remains to be evaluated.

4.2 Evaluating Rankings

Another challenge we faced was how to use the reputation information to rank results. Our prototype currently uses the average KudoRank of developers in a project to rank results. We compared

[6] http://merobase.com/

[7] http://www.advogato.org/

reputation and relevance ranking and found promising results using both reputation and relevance as a factor of ranking.

To understand the effect of using reputation in the rankings, we compared our results against those obtained by Bajracharya et al. [7]. For the 10 queries used in their study, they identified the 57 best hits as an oracle for evaluating the results. Sourcerer returned only 32 of these best hits, which sets a ceiling on our results. We compared the position of these hits in the search results using reputation alone with the position returned by Sourcerer using relevance alone.

We found that 18 of the 32 best hits (56.2%) had a higher position (closer to the top), 11 (34.4%) had lower positions, and 3 were unchanged when ranked using only reputation instead of using only relevance. Among the hits that moved up, the biggest improvement went up 191 positions and the smallest change was 2 positions. Among the results that went down, the largest change was 647 positions and the smallest was 22 positions.

Our results suggest that using reputation ranking can be a plausible way to rank source code results using social factors. The similar number of results that went up (18) and the number that went down (11) suggest that using both relevance and reputation would be beneficial. However, it is not clear the relationship between relevance and reputation. Giving more importance to reputation could obscure very relevant results from low reputable projects. Our approach is promising also because it is consistent with results from previous research [2, 9]: developers use reputation along with technical information such as the functionality, license, user support, and level of project activity.

4.3 Reputation of People versus Reputation of Source Code

We used the average KudoRank assigned to developers on a project to calculate the reputation of source code. Thus, we used the reputation given to people to calculate the reputation of source code. We believe that KudoRank is a good indicator of quality for projects, but we can go further by adding information from other sources and by calculating the reputation at a lower level of granularity (e.g. classes).

One way to improve our reputation metric for source code is by including reputation information at the project level such as the number of downloads, number of users, and number of recommendations. This information is currently shown in source code repositories such as SourceForge and could be mined from there. Another option is to calculate the reputation metrics at a lower level of granularity. In our prototype, we include the reputation at the level of contributors and projects. However, reputation can also be shown at the file or package level. Further research is needed to identify if a lower level of granularity will be useful for developers looking for source code on the Web.

5. FUTURE WORK

With our prototype, we learned that a ranking algorithm that relies on both relevance and reputation would be beneficial. To further investigate the effectiveness of including reputation information in the ranking, we plan to evaluate different ranking algorithms that combine both technical information and reputation information. We also need to evaluate if reputation ranking improves the quality of results and also if reputation is useful for developers looking for source code.

We would also like to investigate how people use reputation information. We plan to conduct a laboratory study to better identify what reputation information is used and how is it used by developers while evaluating source code results from the Web. Data from this study will help us identify what reputation information needs to be crawled from the Web and how to display it to the users.

6. ACKNOLEDGEMENTS

Thanks to Sushil Bajracharya who helped us extending Sourcerer. This material is based upon work supported by the NSF under Grant No. IIS-0846034. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessary reflect the views of the NSF.

7. REFERENCES

[1] Gallardo-Valencia, R. E. and Sim, S. E. 2009. Internet-Scale Code Search. In *Proceedings of the 2009 ICSE Workshop on Search-Driven Development-Users, Infrastructure, Tools and Evaluation* (Washington, DC, USA, 2009). IEEE Computer Society, pp. 49-52.

[2] Sim, S. E., Umarji, M., Ratanotayanon, S. and Lopes, C. V. 2012. How Well Do Internet Code Search Engines Support Open Source Reuse Strategies? *ACM Transactions on Software Engineering and Methodologies* To appear.

[3] Madanmohan, T. R. and De', R. 2004. Open Source Reuse in Commercial Firms. *IEEE Software*, 21, 6 (2004), pp. 62-69.

[4] Chen, W., Li, J., Ma, J., Conradi, R., Ji, J. and Liu, C. 2008. An Empirical Study on Software Development with Open Source Components in the Chinese Software Industry. *Software Process: Improvement and Practice*, 13, 1 (2008), p. 12.

[5] Linstead, E., Bajracharya, S., Ngo, T., Rigor, P., Lopes, C. and Baldi, P. 2009. Sourcerer: Mining and Searching Internet-Scale Software Repositories. *Data Mining and Knowledge Discovery*, 18, 2 (2009), pp. 300-336.

[6] Resnick, P., Kuwabara, K., Zeckhauser, R. and Friedman, E. 2000. Reputation Systems. *Communications of ACM*, 43, 12 (2000), pp. 45-48.

[7] Bajracharya, S., Dou, Y., Ngo, T., Baldi, P., Linstead, E., Lopes, C. and Rigor, P. 2007. *A Study of Ranking Schemes in Internet-Scale Code Search*. Technical Report, Institute for Software Research. University of California, Irvine.

[8] Gysin, F. 2010. Improved Social Trustability of Code Search Results. In *Proceedings of the International Conference on Software Engineering* (Cape Town, South Africa, 2010).

[9] Umarji, M., Sim, S. E. and Lopes, C. 2008. Archetypal Internet-Scale Source Code Searching. In Ifip International Federation for Information Processing, Volume 275: Open Source Development, Communities and Quality, B. Russo, E. Damiani, S. Hissam, B. Lundell and G. Succi, Eds. Boston: Springer, pp. 257-263.