Improved Social Trustability of Code Search Results

Florian S. Gysin
Software Composition Group
University of Bern, Switzerland
flo.g@students.unibe.ch

ABSTRACT

Search is a fundamental activity in software development. However, to search source code efficiently, it is not sufficient to implement a traditional full text search over a base of source code, human factors have to be taken into account as well.

We looked into ways of increasing the search results code trustability by providing and analysing a range of meta data alongside the actual search results.

1. RESEARCH PROBLEM AND MOTIVATION

One of the major problems of code search engines (CSE) concerns the *trustability* of search results. When trying to achieve effient levels of software reuse, the effort to implement the reused software must be minimized. This automatically eliminates the solution of the user just reading through the source code, trying to understand it, and making sure it does actually acomplish what it is supposed to. We need ways of ensuring code trustability for minimal costs. Let us consider an example:

Using a CSE Luke has found some results matching his specifications. He would now like to use the found source code in his project - but he does not know under which license the found code was published, and whether it is still maintained. Luke tries to contact the original author but cannot find any name nor can he figure out from which original project the code snippet was taken. Luke gets frustrated and decides to implement the code on his own.

The portrayed situation may be exaggerated but it captures one of the problems of code search. The higher the trust the user has in the search results, i.e. the more he knows about them, the more likely he is to reuse the code.

Human trustability factors are, for example, which developers worked on a project, and what other projects they worked for. Social trustability factors are, for example, how well people (developers, users, ...) like certain projects, and how renowned the developers are who worked for them.

ICSE '10, May 2-8 2010, Cape Town, South Africa

We are suggesting a solution comprising different answers to sub problems of the overall code search topic. We created a base of meta data linked to our source code repository to provide this information to users. Trough this, trustability of results is increased.

2. BACKGROUND AND RELATED WORK

Searching for reusable source code is a fundamental activity for developers.[1] Based on this finding the field of source code search showed an increased activity over the last years which resulted in the CSEs available today. However, to support search-driven development it is not sufficient to implement a full text search over a base of source code, human factors must be taken into account as well. At last year's SUITE workshop, *suitability* and *trustability* have been major issues in search-driven development, besides—of course—relevance of search results.

Google Code Search[1], *Krugle*[2] and *Koders*[3] are examples of commercial CSE's having rich code bases but supply very little additional information that helps estimating the result trustability: All three only provide the license of the source code and the name of the originating project.

Sourcerer[4] by Bajracharya et al. [2] is a CSE providing structured search over open source Java code. It supplies the developer with the same information as the above mentioned projects but furthermore allows the user to specify unit tests which the found results must comply with [4]. Unit tests are one of the technical trustability factors of source code: The users trust greatly increases when he knows that the found results comply with his unit tests. A new version of *Sourcerer* with trustability data is in development, though it has not yet been published[5].

Code Conjurer [3] is a CSE based on *Merobase*[6], a structured CSE. It as well allows specification of unit tests as a mean to increase code trustability throug a technical factor.

Ohloh[7] is a social networking platform for open source software projects where projects (or rather their developers) can specify additional information. *Ohloh* is not a CSE. *Ohloh* collects statistics about developers, users and the source code of projects. Users can express for both projects

[1] http://www.google.com/codesearch/
[2] http://www.krugle.org/
[3] http://www.koders.com/
[4] http://sourcerer.ics.uci.edu/
[5] Personal communication with Sushil Bajracharya.
[6] http://www.merobase.org/
[7] http://www.ohloh.net/

and developers whether and how much they like them by rating projects and giving kudos to certain developers.

The issue of providing meta data for search results and so increasing trustabilty has not been widely covered and we are addressing this with the work in this project. Especially the estimation of the human and the social factors in source code trustability for a search result is hardly possible with any of the aforementioned CSEs.

3. APPROACH AND UNIQUENESS

We developed a prototype, *JBender*, enriching code search results with trustability information. To add to the information content we join two main sources. On the one hand there is the actual code base of the search engine over which an index is created. On the other hand we created a database of metadata to the projects in the code base.

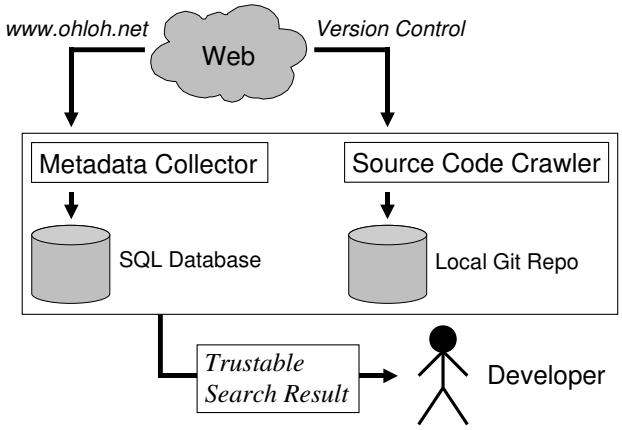

Figure 1: Architecture of trustability enhanced code search engine.

Our source of meta data is the *Ohloh* project. It provides a variety of information about open source projects, composing valuable information for search users. However *Ohloh* does not allow users to actually search through or interact with the source code.

The code base to the open source projects in *Ohloh* is collected from the different projects code repositories from various versioning control systems. Our prototype *JBender* parses the code in Java projects and builds an index over it.

The result for the search over this index does provide additional information alongside the source code as shown in Figure 1. Beside the technical aspects of trustability, which can for example be ensured through unit testing [4, 3], the focus lies on the human and social trustability aspects. Through knowing who worked for a certain project (and for which other projects they worked), how many users the project has, how it is rated, etc. a search user gains an estimate of the trustability of the search results.

As a last step this raw trustability estimate is automated using a trust function. This trust function takes into account several of the collected meta parameters and calculates a trust metric for each result according to which the results can be sorted.

Currently we are building up our metadata and code bases. Upon reaching a sufficient level we plan do a user study to evaluate the effect of metadata on result trustability.

3.1 Improving trustability of search results

Trustability is a big issue for reusing source code. For a result to actually be helpful and serve the purpose originally pursued with the search it is not enough to just match the entered keywords. It is essential that the developer knows at least the license under which certain source code was published, otherwise he will not be able to use it legally. Furthermore a developer can at least guess about the quality of a piece of code when he knows who developed it, and in which project it was originally used. Again for efficiency's sake, this information must be accessible fast and easily, i.e. must preferably be provided by the code search engine alongside the source code.

JBender provides a source code search over various parts of the source code like method/class names and their bodies, comments, visibility, dependencies and implemented interfaces. Its novelty however lies in the underlying data. In addition to the indexed source code base *JBender* is endowed with a database of metadata. This metadata is linked to the projects in the searchable code base - upon finding results from the latter *JBender* can supply the meta information stored for the result's originating project. Metadata stored in the database includes (inter alia): Description of original project, project homepage, rating of the project, list of current repositories (type, url, last time of update, ...), licenses of files in the project (exact type of license, number of files), employed programming languages (percentage of total, lines of code, comment ratio, ...) and developers who worked on the project (kudos, experience, ...).

4. RESULTS AND CONTRIBUTIONS

We developed ***JBender***, a code search engine for Java source code written in Ruby and Java, using the search engine framwork Lucene[8]. Its emphasis lies in encreasing the users trust in the offered search results. This is achieved by maintaining a database of meta information and providing the user with a rich set of metadata alongside the results to increase code trustability. This metadata in particular contains social and human trustability factors, *e.g.* the rating of source projects, the no. of users, the experience of involved developers and whether the project is still maintained.

5. REFERENCES

[1] S. Bajracharya, A. Kuhn, and Y. Ye. Suite 2009: First international workshop on search-driven development - users, infrastructure, tools and evaluation. In *Software Engineering - Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on*, pages 445–446, 2009.

[2] S. Bajracharya, T. Ngo, E. Linstead, Y. Dou, P. Rigor, P. Baldi, and C. Lopes. Sourcerer: a search engine for open source code supporting structure-based search. In *OOPSLA '06: Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*, pages 681–682, New York, NY, USA, 2006. ACM.

[3] O. Hummel, W. Janjic, and C. Atkinson. Code conjurer: Pulling reusable software out of thin air. *Software, IEEE*, 25(5):45–52, 2008.

[4] O. A. L. Lemos, S. K. Bajracharya, J. Ossher, R. S. Morla, P. C. Masiero, P. Baldi, and C. V. Lopes. Codegenie: using test-cases to search and reuse source code. In *ASE '07: Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering*, pages 525–526, New York, NY, USA, 2007. ACM.

[8] http://lucene.apache.org/