Archetypal Internet-Scale Source Code Searching

Medha Umarji[1], Susan Elliott Sim[2], and Crista Lopes[2]
1 University of Maryland Baltimore County,
Department of Information Systems
Baltimore, Maryland, USA
medha1@umbc.edu,
WWW home page: http://userpages.umbc.edu/~medha1/
2 University of California, Irvine, Department of Informatics,
Irvine, California, USA
{ses, lopes}@ics.uci.edu,
WWW home page: http:// www.ics.uci.edu/~ses/
WWW home page: http:// www.ics.uci.edu/~lopes/

Abstract. Programmers often search for Open Source code to use in their projects. To understand how and why programmers search for source code, we conducted a web-based survey and collected data from 69 respondents, including 58 specific examples of searches. Analyzing these anecdotes, we found that they could be categorized along two orthogonal dimensions: motivation (reuse vs. reference example) and size of search target. The targets of these searches could range in size from a block (a few lines of code) to a subsystem (e.g. library or API), to an entire system. Within these six combinations of motivations and target sizes, nine repeating motifs, or archetypes, were created to characterize Internet-scale source code searching. Tools used for searching and the criteria for selecting a component are also discussed. We conclude with guidance on how these archetypes can inform better evaluation of Internet-scale code search engines, as well as the design of new features for these tools.
Keywords: *Source code, search engine, empirical study, Open Source.*

1 Introduction

The growth of Open Source has made a "rich base of reusable software" available on the Internet that programmers leverage in their own software development projects [1]. As this opportunistic software development becomes increasingly prevalent, finding the right snippet, component or application to reuse becomes a common activity for developers. However, little is known about how they go about these searches. To this end, we conducted an online survey of programmers and asked them to give us specific examples of how they searched for source code, the tools they used, and criteria for making a final selection.

A total of 69 participants responded to our survey and contributed 58 anecdotes about how they searched. By analyzing the anecdotes qualitatively and inductively, a number of patterns or "archetypes" were identified. These anecdotes could be categorized along two orthogonal dimensions: motivation and size of search target. With respect to motivation, participants tended to be looking for code that could be reused with little or no modification, or they were looking for code to use as a reference example. Often a search began by looking for code to reuse without modification, but when one was not found, a close match was used as a reference example. Within these categories, we identified nine different archetypes.

The remainder of this paper is organized as follows. We describe our research method in Section 2. In Section 3, we present our categorization and archetypes. In Section 4, we discuss how our participants conducted their searches. We explore how these archetypes can be applied in Section 5 and we make some concluding remarks in Section 6.

2 Online Survey

Surveys have become an established empirical method, especially for Internet usage [2][3]. We designed an online survey with 13 closed-ended questions and two open-ended questions. The survey also had questions about the different tools used during search, programmer background, and criteria for finally selecting a component. In this paper, we will be focusing on the data from one of the open-ended questions, which asked:

> *Please describe one or two scenarios when you were looking for source code on the Internet.*
> *(Please address details such as: What were you trying to find? How did you formulate your queries? What information sources did you use to find the source code? Which implementation language were you working with? What criteria did you use to decide on the best match?)*

We solicited responses using convenience sampling by sending invitations to participate in the survey to the following newsgroups and mailing lists: Javaworld mailing list, the CGI beginner's list on perl.org, comp.software-eng, comp.lang.c, and comp.lang.java. While our responses give us a broad overview of how software developers search for source code, there is no way to determine whether we obtained a representative, random sample. Consequently, we present our results as qualitative archetypes rather than quantitative statistics.

An archetype is a concept from literary theory. It serves to unify recurring images across literary works with a similar structure. In the context of source code searching, an archetype is a theory to unify and integrate typical or recurring searches. The archetypes were produced by analyzing the anecdotes for recurring patterns using open coding [4] and a grounded theory approach [5].

3 Archetypes

In this section, we present the archetypes that we found in our study. Each theme or archetype was a combination of a search motivation and size of search target with additional details about the context.

Detailed analysis of scenarios showed that respondents were either searching for *reusable code* (34) or *reference examples* (17). A key difference between the two motivations is how much modification was expected to reuse the code. Reusable code is source code that they could just drop into their program and use right away, such as a component implementing a quick sort algorithm. A reference example is a piece of code that illustrates a particular solution but needs to be re-implemented or significantly modified; for instance, the syntax of a particular method call in Java.

Across both types of searches, the size of the search target varied in a similar fashion. The size was a continuous variable ranging from a few lines to an entire system, but we defined three broad categories in our analysis. The sizes of the search targets were classified as a block (11), a subsystem (32), or a system (8).

Table 1: Motivation by Target Size

	Description	Code for Reuse	Reference Example	Row Total
Block	Wrappers, parsers, code excerpts	7	4	11
Subsystem	Code of algorithms and data structures, GUI widgets , Library, APIs	21	11	32
System	Stand-alone Applications	6	2	8
Column Total		34	17	51

3.1 Motivation: Reuse

As can be seen in Table 1, there were 34 searches that were motivated by reuse without modification. Upon performing a thematic appraisal, we were able to identify four archetypes or themes of searches within the reuse motivation. The archetypes are presented in increasing order of target size.

Looking for code snippets, wrappers or parsers: The seven searches where developers were looking for a block of code to reuse included searches for a "Java wrapper code for the native pcap library" and an "RSS feed parser." Also included were searches that performed some small functionality, like "encode/decode a URL" and "convert uploaded images of all types to jpeg". At this level of granularity we

included searches for source code components that solved a small problem, or for code snippets to add functionality (with little or no modification).

Reusable data structures, algorithms and GUI widgets to be incorporated into an implementation: Participants were looking for subsystems such as algorithms and data structures; "two-way hash tables," "B+ trees," "trie trees," and "binary search algorithm" were some of the eight searches included in this archetype. These searches are popular, as there is a close match between the vocabulary in the code and the vocabulary for describing the search. There were also seven searches for graphical user interface (GUI) widgets in our data, for example "inserting a browser in a Java Swing frame." GUI widgets are easy to reuse, and provided the widget was written using the same GUI framework as the project being developed, there could be relatively few compatibility problems.

A reusable library to be incorporated into an implementation: There were six searches for a reusable package or API. Some examples of the searches were for "speech processing toolkits", "library for date manipulation in Perl" and "Java implementations of statistical techniques."

A reusable system to be used as a starting point for an implementation: In six searches, developers were looking for "stand-alone tools" or a "backbone for an upcoming project" or just a "big piece of code that does more or less what I want." Examples of search targets included an application server, an ERP package or a database management system. We conjecture that this type of search is common, because a system does not need to be de-contextualized before it is used in a new project. Also, systems are easy to find because they typically have web sites or project pages that contain text descriptions of the software.

3.2 Motivation: Reference Examples

There were 17 searches for reference examples, as shown in Table 1. We also used thematic appraisal on these searches, and coincidentally found four archetypes.

A block of code to be used as an example: Programmers were not looking for reusable components, but their goal was to learn through examples, such as "examples of thread implementation in python." Respondents reported that "reference examples of language syntax and idioms" were helpful when working with an unfamiliar programming language, and "…mostly I look for code for syntax, I don't always like to refer to books for syntax if it is readily available on my desktop."

Example of how to implement a data structure, algorithm or GUI widget: In three instances, developers looked for source code snippets to verify their solution or to aid in re-implementation, e.g. "to verify the implementation of a well-known algorithm." An implementation is more informative than a description or pseudocode, and a running program has a lot of credibility. There were also two

searches for examples of how to use GUI widgets. These included searches for code samples on how to use a particular component from Swing and Microsoft Foundation Classes.

Example of how to use a library: Developers looked for examples of how to use a library in six instances, such as, "Sometimes; I did a source code searching when I don't know how to use a class or a library." Libraries and APIs can be complex to use, and a reference example is easier to understand than documentation, because it gives the programmer a starting point that can then be tweaked to suit the situation.

Looking at similar systems for ideas: While developing or modifying a system programmers look at existing similar systems for ideas. Two searches were for systems from which the logic/principles can be borrowed to construct new systems. This technique was mainly used when it was easier to construct a new system rather than adopt an existing one, or on a proprietary Closed Source project that could not be contaminated with Open Source code.

3.3 Information about Defects

There were five searches that did not fit the above motivations and search target matrix. These were searches for information on resolving defects.

Confirmation and resolution of defects: Developers prefer to search for a patch or quick-fix by forming natural language queries with the keywords from an error message or keywords based on the functionality deviation caused by a bug or defect. There were five searches that were looking for solutions to a defect. Sometimes the solution was in the form of a report or post to a forum that confirmed the presence of a bug. At other times, there was a patch that repaired the defect. Three of the searches led developers to find relevant information in mailing lists and forums.

4 Search Tools

A majority of the developers that responded to our survey were programmers in Java (54), C++ (58) and Perl (32). More than half of the respondents (37) had contributed to an Open Source project. Most of the respondents (41) had experience working on small teams with 1 to 5 people.

Sixty of the participants reported turning to a general-purpose search engine, such as Google or Yahoo! to search for source code. Project hosting sites came next, with 34 of the respondents using them for source code search. Only 11 out of 69 respondents reported using a code-specific search engine. Others were highly skeptical, with one participant writing, "In short, I would never rely on a 'code search engine'. This idea is just plan silly…sort of ivory tower. If you want to find something usable you have to look for people already using it." Some did just this, by using social tagging sites, such as (del.icio.us) to find code

While selecting which component to reuse, the most important single criterion was functionality. The type of software license came in second. Help from a local expert, an electronic forum, a mailing list archive, or active users, were other major considerations while choosing Open Source software. One respondent said he looked for "… issues which are then found by people, solved and posted on mailing lists of discussion forums." Overall, social characteristics of the project, such as the level of activity, seem to have precedence over characteristics of the source code, such as whether the code is peer reviewed and tested.

5 Applications of Archetypes

The search archetypes described here are a succinct representation of a developer's search for source code. We intend our results to inform further enhancement and design of these source code search engines, such as Koders, Google Code Search, and Krugle.

Inform the evaluation of code search engines: The archetypes can be instantiated into "scenarios" that closely represent the user's situation and are easily understandable by users. These scenarios can then be used to compare and evaluate the effectiveness of different search engines. For example, a generic archetype is a "Search for a reusable data structure." Now this can be instantiated into a scenario such as "Consider that you are looking for the Trie tree data structure written in C to use in your project". Such a scenario will provide the necessary contextual information for users to judge whether the results returned by search engine A are better or worse than those from search engine B.

Inform the design of additional features for these search engines: Our archetypes also point the way to possible new features in Internet-scale code search engines. For example, we found that users search for *information about a particular bug* within an open source project. This kind of search is currently not well supported by code search sites. Although a bug may not be resolved completely, people on forums and mailing lists may have discussed it, and users have to perform a text-based search to find this information. Another possible feature is *providing usage examples* for a library or package. In case a user searches for a particular library, package or API, his search experience will be enhanced if results also contain links to usage examples. Finally, based on our survey findings we observe that a *user rating or recommendation for each item returned* by a search engine will be helpful to a developer trying to choose between available options.

6 Conclusions

The main contribution of this work is insight into the varieties of Internet-scale source code searches that programmers perform. Until now, there has been little research about what types of source code programmers look for and how they find and use it. We conducted a web-based survey of how people look for code on the Internet. We identified a categorization for searches along two dimensions (motivation and size) and presented nine archetypes of searches within these categories.

The results of this study are applied to the design of tool features, and in the formation of scenarios for evaluating the existing search tools. Tools for locating source code would be complemented by recommender systems and reviews; more like Amazon.com and less like a library catalog. One of the future directions for this line of research is the use of scenarios to evaluate different search engines. We are continuing with experiments in this vein and hope to continue making contributions to Internet-scale source code searching.

References

[1] D. Spinellis and C. Szyperski, "Guest editors' introduction: How is open source affecting software development?" *IEEE Software* vol. 21 pp. 28-33 2004

[2] V. M. Sue and L. A. Ritter, *Conducting online surveys*: Sage Publications, 2007.

[3] S. E. Sim, C. L. A. Clarke, and R. C. Holt, "Archetypal source code searches: A survey of software developers and maintainers " in *Proceedings of the 6th International Workshop on Program Comprehension* IEEE Computer Society, 1998 pp. 180

[4] M. B. Miles and A. M. Huberman, *Qualitative data analysis*: Sage Publication, 1994.

[5] A. Strauss and J. Corbin, *Basics of qualitative research: Grounded theory procedures and technique*. Thousand Oaks: Sage Publications, 1990.

[6] S. Bajracharya, T. Ngo, E. Linstead, Y. Dou, P. Rigor, P. Baldi, C. Lopes. *Sourcerer: A Search Engine for Open Source Code Supporting Structure-based Search*. October 2006. Companion to the 21st ACM SIGPLAN conference on Object-oriented programming languages, systems, and applications OOPSLA '06. ACM Press.