Using an Information Retrieval System to Retrieve Source Code Samples

Renuka Sindhgatta
Infosys Technologies Limited
Electronics City, Bangalore
India
renuka_sr@infosys.com

ABSTRACT
Software developers often face steep learning curves in using a new framework, library, or new versions of frameworks for developing their piece of software. In large organizations, developers learn and explore use of frameworks, rarely realizing, several peers may have already explored the same. A tool that helps locate samples of code, demonstrating use of frameworks or libraries would provide benefits of reuse, improved code quality and faster development. This paper describes an approach for locating common samples of source code from a repository by providing extensions to an information retrieval system. The approach improves the existing approaches in two ways. First, it provides the scalability of an information retrieval system, supporting search over thousands of source code files of an organization. Second, it provides more specific search on source code by preprocessing source code files and understanding elements of the code as opposed to considering code as plain text.

Categories and Subject Descriptors
D.2.3 [**Coding Tools and Techniques**]: Program Editors D.2.3 [**Coding Tools and Techniques**]: Object Oriented Programming D.2.13 [**Reusable Software**]: Reuse Models

General Terms
Experimentation, Languages.

Keywords
Source code repository, Search, Information retrieval

1. INTRODUCTION
Software developers often use common frameworks or libraries for developing their piece of software. In most cases, using a framework or a common library is not easy as each library has a set of operations that need to be performed in a particular order to provide a particular functionality. Often there is a steep learning curve involved in learning to use of most of these frameworks. In large organizations (> 20,000), it is possible that several people are working on the same framework or library but are unaware that others in the organization are working on the same thing or have already finished working with them. In such situations, work done by a set of developers can be used as sample code for others. Development based on samples code repository provides the benefits of code reuse, development efficiency and code stability.

Motivated by the need to provide examples of code, much work has been in the past few years [1, 2, 3, 4, and 5], in providing tools that work with development environments, understand the users' context and provide relevant examples of code. In large organizations as ours, where there are departments working in multiple domain, languages, versions, platforms, the repository of reusable code can grow at an alarming pace and hence the systems built should be capable of handling such large repositories. Hence, the challenge here is to develop tools that can support varied languages and versions across a large repository of sample source code quickly.

Our ongoing effort, from which we draw our work reported here, is focused on using an information retrieval system to support a large code repository. Currently, text based information retrieval systems have been successfully used to locate relevant documents. They have been widely used in the organizations to mine the organization data. These systems are known for their scalability and simplicity. When the same systems are used to search source code, they do not always provide relevant results as source code is structured where both a keyword and its location in the code needs to be considered. A keyword present in the comment of code can mean different as compared to its occurrence in the method block. Hence, extensions need to be provided over a standard information retrieval system to enable source code search.

We propose a tool where sample source code, published by the developers is preprocessed and indexed. The indexes contain information relevant to the programming language and enable more specific search on source code. To investigate the approach, we built the tool, JSearch, with support for Java Language over the Lucene[1] Java Search Engine Library. The client portion of the tool is available as a plug-in in Eclipse[2] IDE and as published website within the organization. The server portion of the tool consists of an Information retrieval system that creates indexes on the code repository and enables querying on indexes. The evaluation of JSearch is based on identifying various types of developer queries, the tool is capable of handling, where the developers queries on the discussion boards of the organization is analyzed.

The paper presents the sample scenario on use of the tool. The paper compares the approach of this tool with other related work

ICSE'06, May 20-28, 2006, Shanghai, China.
ACM 1-59593-085-X/06/0005.

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

[2] http://www.eclipse.org/

in the area, describes the tool in detail and presents the evaluation results of the tool.

2. SAMPLE SCENARIO

XML documents are commonly used in most software systems. It is often required to create an XML document and store it into a file. Consider a developer who wants to store an XML document to a file. The developer would like to search the repository for available code. The two keywords the developer would associate his requirement would be "Document" and "File". As Document is a common keyword the client plug-in identifies the relevant Document object from the declarations and adds the import directive "org.w3c" to the query. In the eclipse plug-in, the developer enters it as a comment and invokes search. The results of the query are presented with details of the class, the location and the rank of relevance as shown in Fig 1. The snippet of the code matching the user query is available to the developer when the matching result is selected. In case of the web based client, the developer formulates the query to search relevant Java code as "import:org.w3c AND Document AND File". A help is provided to the developer which is similar to the "Advanced Query" option provided in most information retrieval systems to refine the query. The option enables the developer to be very specific in defining requirements of the sample code.

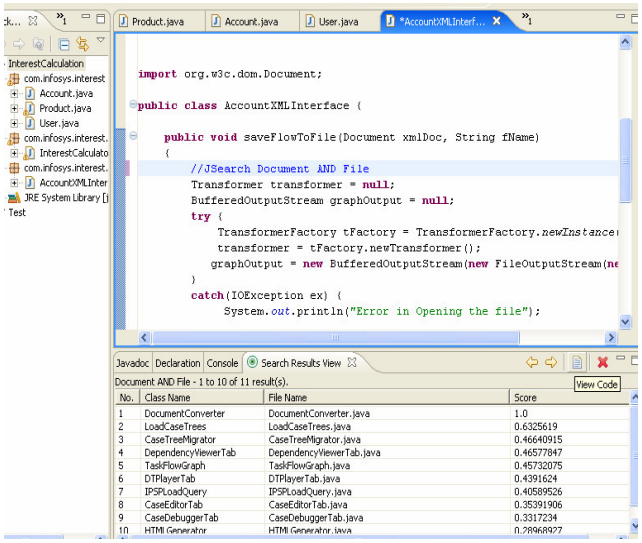

Fig 1: JSearch Eclipse Plug-in

The developer can browse the code and verify if the code fragment is sufficient or if it requires addition of some more classes. The developer can copy the code and make relevant changes to them based on the context of usage.

For the creation of the repository of source code, the developer can select a class or a set of classes and submit to the repository. The files are preprocessed, the syntactic elements of the code are identified based on the programming language and indexed.

3. RELATED WORK

JSearch, we claim, uses an information retrieval system to search a repository of source code. The developer is capable of creating a query representing requirements as a structured text string. The query is searched within the existing indexes and the results are provided.

Eclipse IDE provides search on the code in a project. However, search is limited to single term in projects that are currently open in the environment. There are two limitations - First, single term search is restrictive for a user to express a requirement. Second, as search is within a set of projects open in the IDE, the sample code repository is limited.

CodeFinder [1] is a tool that is very close to what JSearch provides to the developer. CodeFinder uses Spreading Activation technique to search example source code. The advantage of CodeFinder is that it helps the user reformulate and refine the query. However, as the repository of sample code increases, spreading activation may provide several unrelated results when used in the context of searching for a framework or library. For example, in a repository containing several source code elements related to Java Swing Library, it is very likely that common classes like JPanel would be referred in most cases. A user searching for samples of code on a particular swing component may have results containing many other swing components by their association with common classes. Hence, the relevance of results needs to be evaluated for such scenarios.

Prospector [2] is another Eclipse plugin that retrieves samples of code from the Eclipse and Java Standard Development Kit (JSDK). The query language of the tool comprises of a pair of (Tin,Tout) where Tin and Tout are class types. Prospector takes in the query and traverses through all the paths from Tin class to Tout class in the source code graph. The query supported by Prospector is very useful where a developer is interested in converting Tin to Tout. However, not all of a developers programming intent can be represented with an input and an output class.

Stranthcona [3] is another Eclipse plug-in that uses structural context from a code fragment and retrieves samples with similar context. It extracts three types of structural contexts: i) Inheritance heuristics - classes having the same parent as the class in the current context ii) Calls Heuristics - methods calling the same methods that is being called in the current context iii) Uses heuristics - methods referring to same data types as referred in the current context of the code. While the tool addresses all contexts, there may be scenarios where a developer would be well aware that a particular heuristic used by the tool is not important for samples of code he is interested but the tool may include the heuristic in creating the context.

CodeBroker [4] is another tool that does the similarity analysis between components based on concept similarity or constraint compatibility. Concept similarity is identified based on comments in the source code. Constraint similarity is identified based on the method signatures. It further refines the query with inputs from the user. The tool is similar to Strathcona in helping the user to formulate queries.

The earlier approach of using an Information retrieval system on source code [5] considered source code files as free text documents. This method resulted in low recall and precision. This paper details the work done to improve recall, precision and query of an information retrieval system when applied to source code using JSearch. One of the biggest advantages of JSearch tool is the flexibility and scalability given to a user. The repository of sample code can store code across multiple

platforms and versions. However, the tool assumes that the developer has just enough information on what library is to be used but does not know how to use.

4. JAVA SOURCE CODE SEARCH

This section describes the implementation of JSearch tool. A repository of sample code is created by developers submitting their code. A developer wanting to implement a task enters the query string into the JSearch tool. The tool presents the user with results ranking them based on the match between the query and the source code indexes. The developer scans through the results which that contains the code snippets and can use them for completing a task or portion of a task.

4.1 Populating Source Code Repository

Software developers or projects could publish their code. This involves either uploading their files to the server using a web interface or registering the URL from which all the source code files are downloaded regularly and indexed. The Progressive Open Source method [6] suggests three tiers. Inner Source - source available to all developers within the firewall, Controlled source - source available to restricted partners and Open source - Source that is available on the Internet. Similarly, in an organization, various levels of sharing of source code needs to be enabled. There could be a concept of inner source, controlled source and open source within the organization that ensures adherence to license agreement and intellectual property. Based on the project, relevant source code files can be extracted to form an organization wide repository.

Registration of a URL or upload of code files is followed by submitters having to provide details on the version of libraries used, a brief description of the functionality, external files and dependencies for the code to work. The user can also add additional information. The information related to source code is stored in a configuration file.

4.2 Preprocessing Source Code Repository

The key metric used by most information retrieval systems for ranking the relevance of a document is the product of term frequency and inverse document frequency. Term frequency is the ratio of the frequency of the term (keyword in the query) to the maximum frequency of a term in the document. Inverse document frequency is logarithmic ratio of the number of document that exists in the repository to the number of documents that contain the term.

Software developers when looking for sample code in Java, typically have code queries related to Classes (C) and method calls (M). In the source code, the term frequency of C would be much lower than actual as in most places the term C would be referred through variable names. Fig 2 shows a sample Java file containing a member variable of *Document* type. This member variable is referred through in the entire source code. The relevance rank of results gets impacted with the use of variables in source code due to reduced term frequency. Preprocessing involves parsing the code files and translating the java file to a temporary file where the variable names are converted to their classes. Fig 2 shows one such method before and after preprocessing. The preprocessed file is used for creating indexes.

```
public static Document convertProductToXML(Product p){
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        Document document = null;
        try {
            DocumentBuilder builder = factory.newDocumentBuilder();
            document = builder.newDocument();
        }
        catch(ParserConfigurationException ex) {
            ex.printStackTrace();
            return null;
        }
        Element root = document.createElement("Product");
        root.setAttribute("name", p.name);
        root.setAttribute("price", p.price);
        document.appendChild(root);

        return document;
}
```

```
public static Document convertProductToXML(Product p){
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        Document document = null;
        try {
            DocumentBuilder builder = DocumentBuilderFactory.newDocumentBuilder();
            Document = DocumentBuilder.newDocument();
        }
        catch(ParserConfigurationException ex) {
            ParserConfigurationException.printStackTrace();
            return null;
        }
        Element root = Document.createElement("Product");
        Element.setAttribute("name", Product.name);
        Element.setAttribute("price", Product.price);
        Document.appendChild(Element);

        return Document;
}
```

Fig 2: Preprocessing source code

4.3 Indexing Source Code

Indexing code requires analyzing code and creating source code specific indexes. In retrieval systems that index documents of conference proceedings, indexing can be done for each of the relevant elements of a document - title, author, body, and conference name to support efficient querying. Hence, each of these elements is considered to be a different field in the document and is indexed separately.

In the case of Java language, the syntactic elements of a java class such as import declaration, classes it implements and extends, member variables, method names, method code and comments can be considered as fields, for indexing. Indexes can be created for each of the syntactical elements of the source code. The important fields defined in JSearch are - the class name, the class it extends, the method names, the return types, the comments and the import declarations. JSearch parses each file using a Java AST Parser and extracts the fields of each Class and indexes them. This allows for very specific querying that will be discussed in the following section.

Indexes created should be optimal and ensure that only the relevant aspects of code are considered An Information retrieval system for documents analyzes the text and processes them - removes commonly occurring words (known as stop words like a, an, the, etc.), parses words and applys stemming algorithms to remove morphological and inflexion endings. In the context of Java Source code, there are certain keywords of the language that need to be discarded in the process of indexing source code. In JSearch, most of the Java language keywords are discarded to optimize the size of the index. The limitation of this approach is the loss of information about patterns of code - method that contains a *switch* statement, *for* statement or a *private* variable is not available in the indexes. In the context of search being limited to Classes and method calls; it is assumed that such information may not be required.

4.4 Querying Source Code

Indexing multiple fields of a source code document provides the advantage of being able to support flexible queries. Queries containing Boolean algebra have been considered complex for users. We believe that software developers are exposed to the programming language and are comfortable forming queries due to common use of Boolean algebra in programming. Table 1 lists the queries that are possible by a software developer when looking for sample codes available in the repository. The flexibility provided to the developer is very high. Hence, once the user is aware of the class that needs to be used or needs to be extended, specific queries can be created to get relevant results.

Query Expression	Matches Java Class that ….
extends:JDialog code:JTable	extends the class JDialog and uses JTable class in the code
code:Document +import:com.w3c.*	Contains Document in the code and definitely has com.w3c in the import definition
parameter:JTable	Contains JTable as parameter in the a method
parameter:JGraph code:cell	Contains JGraph as a parameter and/or cell in the Code
method:paint -class:Color	Contains a method named paint but does not have Color as the class name
method:paint +parameter:Graphics	Contains a method named paint and definitely has Graphics in any of the method parameters

Table 1: Queries on Source Code

5. EVALUATION

The system was evaluated for performance, scalability and relevance of code reuse. The index creation time increased with additional steps of preprocessing and code element extraction. However, as most information retrieval systems are capable of handling large number of documents, the additional time taken is still capable of indexing large number of source code files. The response time increased from an average of 12 ms for indexing each source code file as plain text to about 30ms for each file with additional preprocessing and parsing. This has been observed in our proof-of-concept system and hence improvements can be made in this regard.

To evaluate the relevance of using such a system for search, we analyzed the discussion board for Java within the organization. A sample of 500 queries from the discussion board for a period of 4 months was extracted. The queries were broadly classified into 3 types. Type 1 - the developer has no idea of what libraries can be used to solve his problem. One such query is illustrated - "*I need to execute a Shell script (Perl script) from Java application. Is there any way to do this?"* In such scenarios, JSearch may not be effective as the user is not aware of the classes or methods required for performing the task. Type 2 - developer has queries related to the run time environment. An example of such query is; *I have a java class for sending mails using javax.mail package. It is giving me a runtime exception. Can any one suggest a solution?* Queries related to run time environment cannot be addressed by a code search tool. Type 3 - the developer knows what should be used and wants some information on how it should be used. Samples of such queries are provided here; (i) *what methods need to be called while using log4j to turn off logging. (ii) Can someone post a code snippet where the process of opening the Connection, setting the transaction boundary and committing the transaction is done?* Such queries can be handled by JSearch as the queries can be formulated as a query expression.

In evaluating the types of queries from the discussion board, there were approximately 23% of the queries that could be handled by JSearch while the remaining queries were of the first and second types of queries. However, we believe that several developers would be using informal mechanisms of finding example source code to solve their task at hand. Hence, a complete deployment populating code samples from internal projects of an organization would be more widely used than indicated by the percentage of queries analyzed on the discussion board. The pilot deployment of the tool in the organization and auditing the usage of the tool for a period of three to six months is required to quantify usage and usefulness.

6. CONCLUSION

Information Retrieval systems have matured over the past few years and have become the default mechanism for searching and sharing information from a large repository of text documents. Organizations having several developers working across geographical locations on different languages, platforms have many developers solving similar problems. Using an Information Retrieval system to search work done by developers would be scalable and easy to use. Such systems would also require a process to be defined to ensure the shared content does not violate any of the agreed licenses, the shared content meets the quality standards and it has sufficient information along with it to make it usable with minimal changes. The quality of the results will need to be compared with the current code query tools. These will be the future directions of JSearch

7. REFERENCES

[1] Henninger, S. Retrieving Software Objects in an example based programming environment. *Procs. of 14th Int'l Conf. On Research and Development in Information Retrieval,* 251-260,1991

[2] Mandelin D et.al. Jungloid Mining: Helping to Navigate the API Jungle. *Procs.of PLDI* , pages 48-61 ,2005

[3] Homes, R. and Murphy, CG. Using Structural Context to Recommend Source Code Examples. In *Proc. of Int'l Conf. on Software Engineering*, pages 117-125, 2005

[4] Y. Ye and G. Fischer. Supporting reuse by delivering task-relevant and personalized information. *In Proc. Of the 24th Int'l Conf. on Software Engineering*, pages 513–523, 2002.

[5] Frakes, BW and Nejmeh BA..Software Reuse through Information Retrieval. *SIGIR Forum*, Vol 21, pages 31-36 1987

[6] Dinkelacker, J et. al, Progressive Open Source. *In Proc. Of the 24th Int'l Conf. on Software Engineering*, pages 177–184, 2002.