A-SCORE: Automatic Software Component Recommendation Using Coding Context

Ryuji Shimada Yasuhiro Hayase Makoto Ichii Makoto Matsushita Katsuro Inoue
Graduate School of Information Science and Technology, Osaka University
{r-simada, y-hayase, m-itii, matusita, inoue}@ist.osaka-u.ac.jp

Abstract

Reusing software components (e.g. classes or modules) improves software quality and developer's productivity. Unfortunately, developers may miss many reusing opportunities since current keyword based component search systems cannot provide reusable components if the developers do not use them. This paper proposes a new automatic component recommendation system which supports various usage scenarios and procedures for component reuse.

1 Automatic Component Recommendation

Reusing software components (e.g. modules or classes) improves software productivity and quality [3]. There exists a countless number of software components but developers cannot easily benefit from their reuse, since components are sometimes poorly documented and not designed for reuse.

To find reusable components satisfying certain requirements, developers often use keyword-based search systems. However, keyword-based search systems recommend components only upon an explicit developer interrogation; consequently, developers may miss a significant number of opportunities for reusing some software components.

To deal with the problem, Ye *et al.* proposed an automatic component recommendation system: *CodeBroker*[4]. Automatic component recommendation systems collect search conditions and requirements while developers edit the source code, and recommend possible reusable components without being explicitly instructed. These systems have two advantages over ordinary keyword search systems:

1. the system can recommend components even if the developer thinks there are no reusable components because it automatically searches possible reusable components;
2. the system can search components even if the developer cannot provide good search keywords because it automatically extracts search conditions and requirements from the source code.

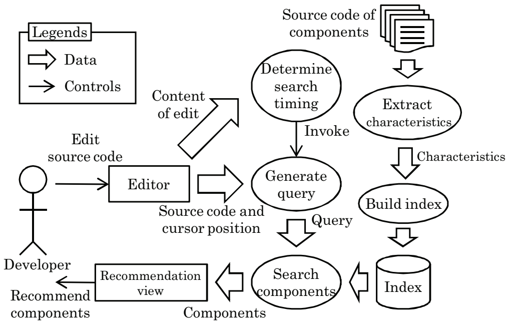


Figure 1. Overview of our approach

Although there are many way to reuse software components, CodeBroker supports only the replacement of a new method with an already existing one; other ways to reuse a component are not supported by the tool.

This paper proposes a new automatic component recommendation system. Our system extends CodeBroker's approach at component reuse and supports not only method replacement, but also modified component imports and code fragment copy-and-paste operations. The system uses comments and identifiers to extract developer's requirements and search conditions.

2 Approach

In our approach components are represented by Java classes. Recommended components are similar to those the developer is editing in the meaning of Latent Semantic Indexing (LSI)[1]. The co-occurrence matrix between documents and words required for LSI searching is computed using components source code as documents and their characteristics as words.

Our approach is divided in two phases: index building and recommendation (see Figure 1). Each phase is described below.

Indexing 1: Characteristic Extraction. Firstly, the component's source code is parsed for extracting elements including characteristics. The elements are document/normal comments, class/method/field/local-variable declarations, and method invocations. Then, each element is divided into discrete words.

Indexing 2: Index Building. Firstly, the characteristic-by-component co-occurrence matrix is built. The (i, j) element of the matrix represents the number of occurrence for the i-th characteristic in the j-th component. A row vector of the co-occurrence matrix is called *component vector*. Then, the dimension relative to component's characteristics is reduced and the matrix is transformed into an LSI-index.

Recommendation 1: Search Invocation. Editing operations are monitored and searches initiated at appropriate time. Specifically, searches are triggered when a statement delimiter (semicolon, brace or end-of-comment mark "$\star/$") is typed, or when the cursor is moved out of a statement or comment after a modification.

Recommendation 2: Query Generation. A query is represented by a set of tuples $< characteristic, weight >$. Characteristics are extracted from the edited source code in the way illustrated at the *Indexing 1* clause. The weight is a positive real value attenuated by the characteristic's distance from the cursor, and represents the relevance of the characteristic for the search.

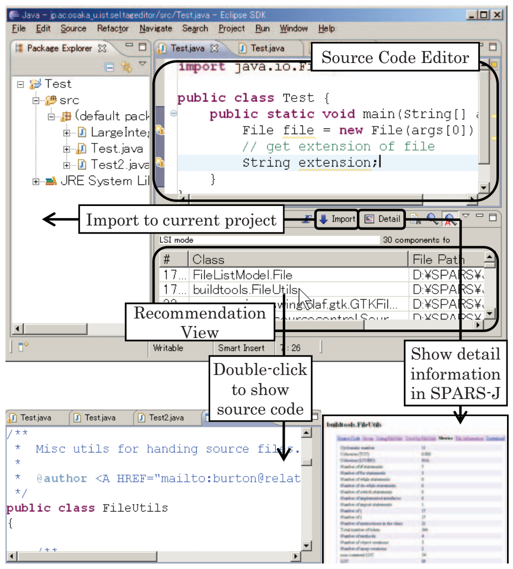
Recommendation 3: Components Search. The index is used to search components matching the query previously built, and the results are presented to the developer. Firstly, the query is transformed into a pseudo-component vector, a sequence of the query's weights in the same order of the component vector; the weight value of characteristics not present in the query is zero. Then, the several components are presented to the developer in descendant order of their cosine similarity value to the pseudo-component vector.

3 Tool Implementation

This section describes A-SCORE (**A**utomatic **S**oftware **Co**mponent **R**ecommendation **E**nvironment), the tool implementing our approach.

A-SCORE is structured as a client-server system: the server is implemented as a web service and the client is implemented as an Eclipse plug-in. The elements in Figure 1 from *Source code of components* to *Search Components* are implemented in the server, and the rest in the client.

Figure 2 shows a snapshot of A-SCORE. The bottom-right view in the main window is the A-SCORE user interface. While a developer is editing the source code,



```
import java.io.Fi
public class Test {
    public static void main(String[] a
        File file = new File(args[0])
        // get extension of file
        String extension;
    }
```

#	Class	File Path
17...	FileListModel.File	D:¥SPARS¥
17...	buildtools.FileUtils	D:¥SPARS¥
20...	...swing.plaf.gtk.GTKFil...	D:¥SPARS¥

```
/**
 *  Misc utils for handing source files.
 *
 *  @author <A HREF="mailto:burton@relat
 */
public class FileUtils
{
```

Figure 2. A Screen Snapshot of A-SCORE

if A-SCORE's automatic recommendations are enabled, reusable component candidates are automatically presented in the recommendation view. If the developer double-clicks a candidate in the view, the source code of the selected component is displayed in a new editor window. A right-click on a candidate allows the developer to display the details of the selected candidate using SPARS-J[2] or, if the developer decides to reuse the component, to automatically download and import the component into the current project.

Acknowledgment

This work was supported by KAKENHI (18650006) and supported in part by "Global COE (Centers of Excellence) Program" of the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

[1] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. *J.Am.Soc.Inf.Sci*, 41(6):391–407, 1990.

[2] K. Inoue, R. Yokomori, T. Yamamoto, M. Matsushita, and S. Kusumoto. Ranking Significance of Software Components Based on Use Relations. *IEEE Trans. on Softw. Eng.*, pages 213–225, 2005.

[3] C. Krueger. Software reuse. *ACM Computing Surveys (CSUR)*, 24(2):131–183, 1992.

[4] Y. Ye and G. Fischer. Reuse-Conducive Development Environments. *Automated Softw. Eng.*, 12(2):199–235, 2005.