Ranking Significance of Software Components Based on Use Relations

Paper in journals: this is the first page of a paper published in ***IEEE Transactions on Software Engineering.***

[***IEEE Transactions on Software Engineering***] **31**,213-225 (2005)

IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 31, NO. 3, MARCH 2005

Ranking Significance of Software Compo Based on Use Relations

Katsuro Inoue, *Member*, *IEEE*, Reishi Yokomori, *Member*, *IEEE*, Tetsuo Yamamoto, *Member*, *IEEE*, Makoto Matsushita, and Shinji Kusumoto, *Member*, *IEEE*

Abstract—Collections of already developed programs are important resources for efficient development of reliable software systems. In this paper, we propose a novel graph-representation model of a software component library (repository), called *component rank model*. This is based on analyzing actual usage relations of the components and propagating the significance through the usage relations. Using the component rank model, we have developed a Java class retrieval system named SPARS-J and applied SPARS-J to various collections of Java files. The result shows that SPARS-J gives a higher rank to components that are used more frequently. As a result, software engineers looking for a component have a better chance of finding it quickly. SPARS-J has been used by two companies, and has produced promising results.

Index Terms—Component rank, graph representation model, reuse models, program analysis, reusable libraries.

1 Introduction

Computer systems are becoming core infrastructures of effective and efficient activities of everyday life. The software that exists in these computer systems is becoming ever larger and more complex, and demand for high software quality is becoming stronger. One promising approach to the efficient development of quality software is to leverage software reuse.

A lot of research on structuring and retrieving software libraries (repositories) for reuse has been performed [10], [14], [16], [19], [21], [23]. However, we do not know much about successful cases, in the sense that library reuse is widely prevalent in software development organizations. Library reuse is vital for efficient development of quality software in the organization.

Mili et al. have extensively investigated and precisely classified a wide variety of research on retrieval of software libraries, and have shown the nature and various characteristics of classified technologies [17]. Their results suggest that a promising approach to a practical reuse system is to employ the information retrieval method based on a textual analysis of software. This method can be highly automated with a low operational cost, and we can easily apply various techniques developed for natural language and HTML documents. However, since the repository and retrieval structures are generally very simple, we usually get a broad result for a query. Thus, it is essential to introduce a mechanism to narrow the query result.

In this paper, we propose a novel ranking method to narrow retrieved software components from reusable libraries. We define a *component rank model* based on a graph representation scheme of the component library [9]. In this model, a collection of software components is represented as a weighted directed graph, i.e., the nodes of the graph correspond to components and the edges linking the nodes correspond to cross component usage. Similar components are clustered into one node so that the effect of duplicated components is removed. The nodes in the graph are ranked by their weights, which are defined as the elements of the eigenvector of an adjacent matrix for the directed graph. The resulting rank, named *component rank*, is used to prioritize the query result so that highly ranked components are quickly seen by the user. The idea behind component rank originates from computing impact factors (called *influence weights*) of published papers [20]. This approach has been extended to ranking Web documents on the Internet [18].

Using the component rank, we have developed a component search system, named SPARS-J, which treats the source files of Java classes as components. This system has been applied to various collections of Java programs, such as JDK, programs downloaded from the Internet, and business applications from two companies.

The results show that a class frequently invoked by other classes (such as those that implement fundamental and standard data structures) generally has a high rank, and that nonstandard and special classes typically have a low ranking. Two companies use SPARS-J for automatic management of their software assets, and SPARS-J shows very promising results.

In Section 2, we propose a component rank model. Section 3 shows the Java component search system SPARS-J based on the component rank model. The results of

- *K. Inoue, R. Yokomori, M. Matsushita, and S. Kusumoto are with the Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University, 1-3, Machikaneyama-cho, Toyonaka-city, Osaka, 560-8531, Japan. E-mail: {inoue, yokomori, matusita, kusumoto}@ist.osaka-u.ac.jp.*
- *T. Yamamoto is with the Department of Computer Science, College of Information Science and Engineering, Ritsumeikan University, Noji Higashi 1-1-1, Kusatsu City, Shiga 525-8577, Japan. E-mail: tetsuo@cs.ritsumei.ac.jp.*

Manuscript received 19 May 2004; revised 15 Dec. 2004; accepted 20 Dec. 2004; published online 20 Apr. 2005.
Recommended for acceptance by W. Frakes.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number TSE-0095-0504.

0098-5589/05/$20.00 © 2005 IEEE Published by the IEEE Computer Society

The following is a comment on the published paper shown on the preceding page.

Ranking Significance of Software Components Based on Use Relations

INOUE Katsuro
(Graduate School of Information Science and Technology)

Introduction

Collections of already developed programs are important resources for efficient development of reliable software systems. In this paper, we propose a novel graph-representation model of a software component library (repository), called component rank model[1]. This is based on analyzing actual usage relations of the components and propagating the significance through the usage relations.

Using the component rank model, we have developed a Java class retrieval system named SPARS-J and applied SPARS-J to various collections of Java files. The result shows that SPARS-J gives a higher rank to components that are used more frequently. As a result, software engineers looking for a component have a better chance of finding it quickly. SPARS-J has been used by two companies, and has produced promising results.

Component Rank Model

Software systems are modeled by a weighted directed graph, called a component graph. A node in a graph represents a software component, and a directed edge e from node x to y represents a use relation, meaning that component x uses component y. Fig. 1 shows a component graph with computed weights, where v_1 has two outgoing edges, and weight 0.4 is evenly divided between two outgoing edges with 0.2 each. Here, v_3 has two incoming edges, each with a weight of 0.2, so that the weight of v_3 is 0.4. The weight of each node $w(v_i)$ is determined by the following equation, and it is computed as the eigenvector.

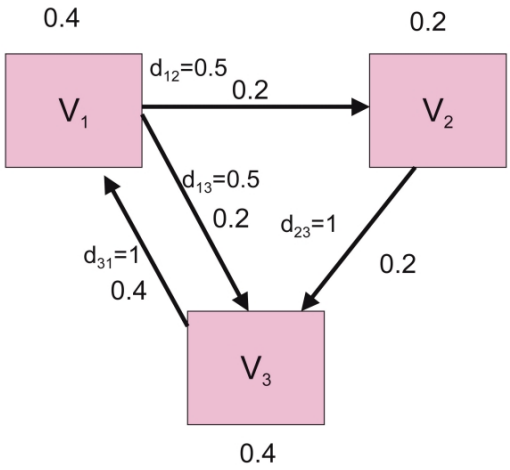


Fig. 1

$$\begin{pmatrix} w(v_1) \\ w(v_2) \\ \vdots \\ w(v_n) \end{pmatrix} = \begin{pmatrix} d_{11}\ d_{12} \cdots d_{1n} \\ d_{21}\ d_{22} \cdots d_{2n} \\ \vdots\ \ \vdots\ \ \ddots \vdots \\ d_{n1}\ d_{n2} \cdots d_{nn} \end{pmatrix}^t \begin{pmatrix} w(v_1) \\ w(v_2) \\ \vdots \\ w(v_n) \end{pmatrix}$$

If we assume that the movement of a software developer's focus on the target components is represented by a probabilistic state transition, the component graph is understood as a Markov chain model. Thus, computing the weights of the nodes in the graph corresponds to attaining a stationary distribution of the chain. This model is inspired by computing the impact factor of publications[2] and the rank of HTML documents[3].

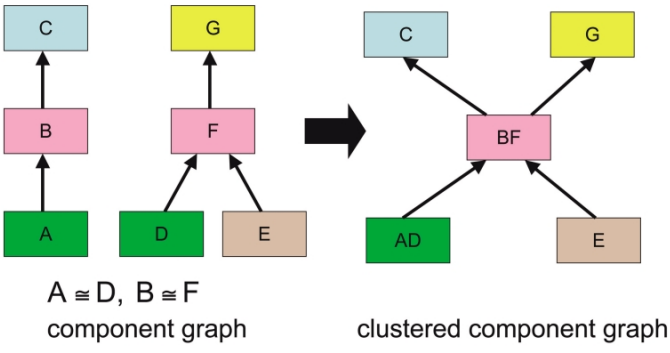
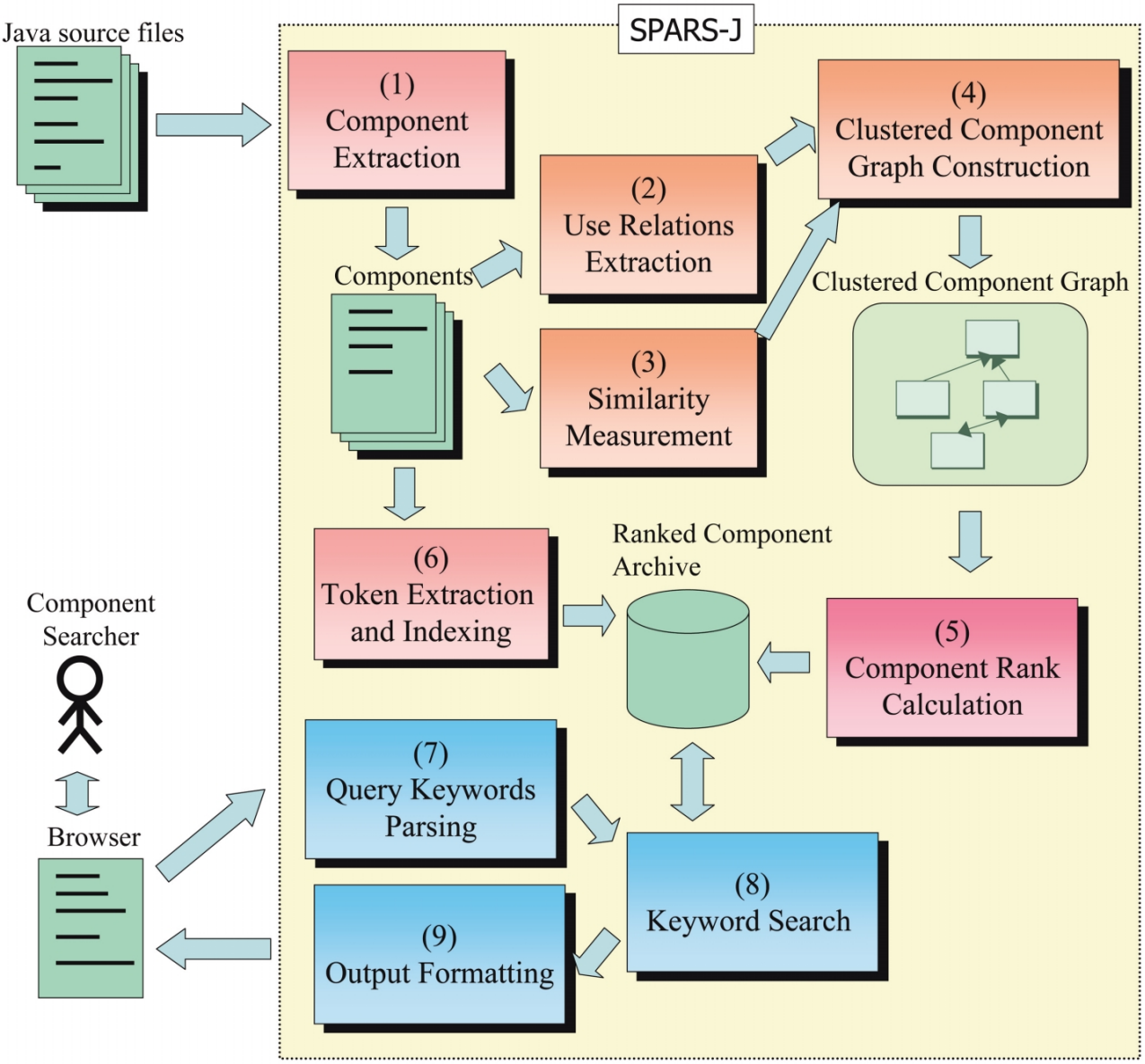


Fig. 2

As a specific feature of software components, we have devised a method of clustering similar software components. In many systems, components are duplicated inside a single system and also they are shared with other systems. To remove the effect of component duplication, we merge similar components into a single one. Figure 2 shows this process. In the left-hand side graph, we detect similar components B and F, and also A and D. Those pairs are merged into single nodes BF and AD, as shown in the right-hand side.

Fig. 3



SPARS-J

Based on the component rank model, we have designed and implemented SPARS-J (Software Product Archiving and Retrieving System for Java) to compute the component rank and to search components for Java programs. Fig. 3 shows the architecture of SPARS-J. Fig. 4a shows an example screenshot of the resulting component list for given query keywords. The details of a component can be seen by clicking an item on the list, as shown in Fig. 4b. On this screen, we can obtain various views of the component, such as its source code (A), similar components (B), components that use this component (C), components used by this component (D), metrics values of the component (E), and others.

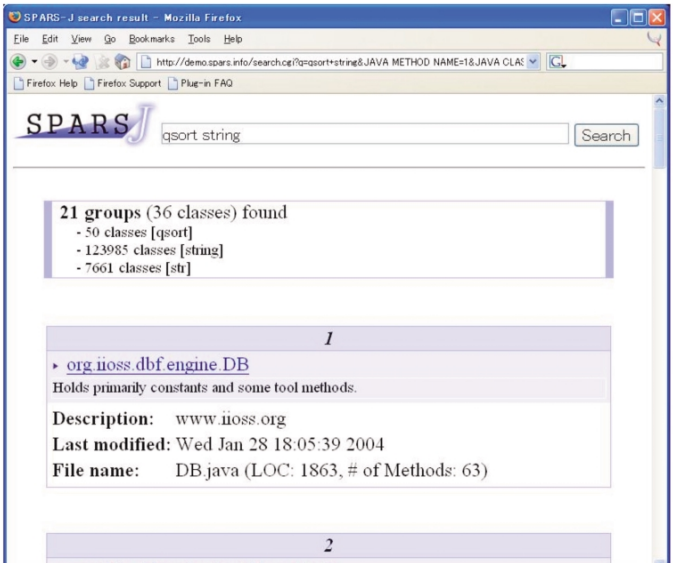


Fig. 4 a

rank	class name	weight
1	java.lang.Object	0.16126
2	java.lang.Class	0.08712
3	java.lang.Throwable	0.05510
4	java.lang.Exception	0.03103
5	java.io.IOException	0.01343
6	java.lang.StringBuffer	0.01214
7	java.lang.SecurityManager	0.01169
8	java.io.InputStream	0.01027
9	java.lang.reflect.Field	0.00948
10	java.lang.reflect.Constructor	0.00936
...		...
1256	sunw.util.EventListener	0.00011
...		...
1256		

Table 1

Experiment with JDK

All source programs of Java 2 Software Development Kit, Standard Edition 1.3.0 are the target of the application. It is composed of 1877 .java files of totally 575,000 lines of code in Java. These files include the classes which are very important and fundamental ones to develop various Java applications.

Table 1 shows the resulting Component Rank values for each file, listed from the highest rank to the lowest one. The highest one, `java.lang.Object` class, is the superclass of any class in Java, so that this class is used directly or indirectly by any class, causing it on the top of the ranking. Other highly ranked classes are also fundamental ones that are possibly invoked or inherited from many other classes. The 3rd class, `java.lang.Throwable`, is the superclass of any error or exception handlers so that it is used by many classes with error or exception handling. There are 622 classes with the lowest (1256th) rank. These classes are not used by any other classes at all. The overall result of Component Rank for JDK 1.3.0 matches to our intuition such that very general and core classes are ranked high, and specific and independent classes are ranked low.

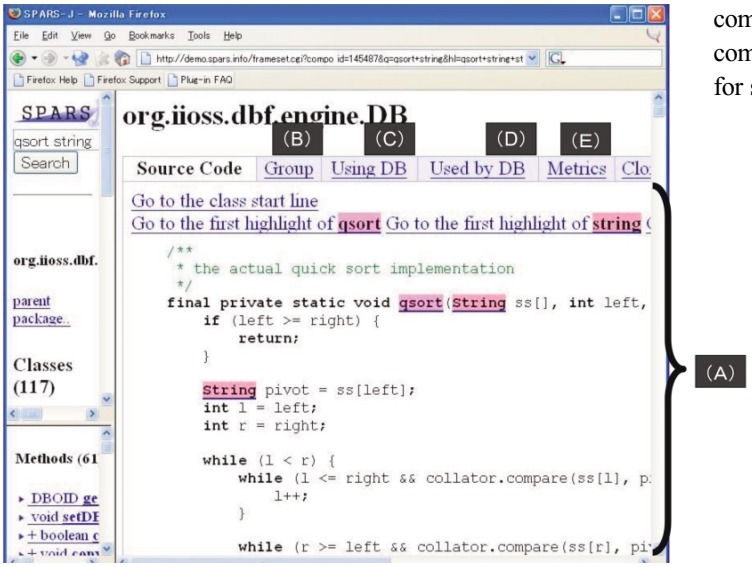


Fig. 4 b

Case Study at Daiwa Computer

Daiwa Computer, located in Osaka, Japan, is a software company with about 180 engineers. In this company, five Web-based data management applications have been developed. These five applications and the framework itself form the target software library of the ranking. The number of components in the framework is 250, and the overall library contains 1,538 components in total, which are clustered into 339 nodes. We investigated the highly-ranked classes and found that those classes are the definition of data structures and their containers. For example, the first-ranked class is the definition of a record class for database management. These results confirm our approach, i.e., it is easy to identify core and fundamental components by their ranking.

Case Study at Suntory Ltd.

Suntory Limited is Japan's leading producer and distributor of alcoholic and nonalcoholic beverages, where hundreds of Java applications have been developed for various activities such as sales, deliveries, accounts, and so on. To evaluate SPARS-J, the company provided about 2,400 components (classes) that are used in the many application programs developed in the company. The evaluation result shows that the SPARS-J was supported by the engineers and managers. The display features provided by SPARS-J (such as using and used-by relations) score highly, as the ranking feature does. Furthermore, some engineers reported that it is easy to grasp the structure of the application and to perform an impact analysis for modification of a component. Currently, SPARS-J is daily used in Suntory as a company-wide software component repository.

Conclusion

The approach of SPARS-J shows a lot of promise for use in various situations of software development, such as searching, exploring, checking, investigating, reminding, or referring to software components, as we use dictionaries and libraries when writing a composition. SPARS-J can be considered as a Google-like system for software engineers.

References

[1] Inoue, K., Yokomori, R., Fujiwara, H., Yamamoto, T., Matsushita, M., and Kusumoto, S., "Component Rank: Relative Significance Rank for Software Component Search," *Proc. 25th Int'l Conf. Software Eng.* (ICSE2003), 14-24 (2003)

[2] Pinski, G. and Narin, F., "Citation Influence for Journal Aggregates of Scientific Publications: Theory, with Application to the Literature of Physics," *Information Processing and Management,* **12**, 297-312 (1976)

[3] Brin, S. and Page, L., "The Anatomy of a Large-Scale Hypertextual Web Search Engine," *Computer Networks and ISDN Systems*, **30**, 107-117 (1998)