Relevancy Based Semantic Interoperation of Reuse Repositories

Ying Pan
Software Institute, EECS
Peking University, 100871
Beijing, P.R.China
pany@sei.pku.edu.cn

Lei Wang
Computer Department
Tsinghua University, 100084
Beijing, P.R.China
wanglei96@tsinghua.org.cn

Lu Zhang
Software Institute, EECS
Peking University, 100871
Beijing, P.R.China
zhanglu@sei.pku.edu.cn

Bing Xie
Software Institute, EECS
Peking University, 100871
Beijing, P.R.China
xiebing@sei.pku.edu.cn

Fuqing Yang
Software Institute, EECS
Peking University, 100871
Beijing, P.R.China
yang@sei.pku.edu.cn

ABSTRACT

Software reuse is a promising solution to the software crisis. Reuse repositories are the basic infrastructure for software reuse. During the past decade, various academic, commercial, governmental, and industrial organizations have developed many Internet-enabled reuse repositories to provide access to software components and related resources. It has necessitated semantic interoperation to allow distributed maintenance and management of these repositories while enabling users to efficiently and conveniently access resources from multiple reuse repositories via a single representation view. In this paper, we have proposed an approach to enhancing the semantic interoperability of reuse repositories, called the *improved relevancy matching and ranking (IRMR)* method, based on analyzing the correlation of terms in representation methods of the repositories. A prototype system, the *Virtual Repository supporting Semantic Interoperation (VRSI)*, is presented to illustrate the application of this approach to support the semantic interoperation of reuse repositories. Experimental results on real world reuse repositories demonstrated significant improvement in terms of searching effectiveness.

Categories and Subject Descriptors

D.2.12 [**Interoperability**]: Data mapping; D.2.13 [**Reusable Software**]: Reusable libraries; D.2.2 [**Design Tools and Techniques**]: Computer-aided software engineering, Software libraries; H.3.3 [**Information Search and Retrieval**]: Query formulation, Retrieval models,Search process

General Terms

Algorithms, Management

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SIGSOFT'04/FSE-12, Oct. 31–Nov. 6, 2004, Newport Beach, CA, USA.
Copyright 2004 ACM 1-58113-855-5/04/0010 ...$5.00.

Keywords

Software reuse, interoperability, semantic analysis, reuse repository, automatic matching

1. INTRODUCTION

It is widely believed that software reuse, a development method of using reusable components to create new systems, is a feasible way to improve both the productivity and quality of software development [4]. However, the developers must be capable to efficiently and conveniently acquire enough desired components before developing with reuse. Current reuse repositories (such as Component-Source [15], Download.com [17], SourceForge [50], Open-Components [43], and Netlib [38]) provide access to software packages (including reusable software components, Web Services, and related documents) via Internet or Intranet.

The existence of many independent software repositories has its advantage to allow each repository to tailor its contents and services [11]. However, multiple independent repositories also result in redundancy and inefficiency, and it is usually inconvenient for users to access each repository separately. Therefore, interoperation has been necessitated by allowing distributed maintenance of these repositories while enabling users to access resources from multiple repositories via a single interface [10].

Current approaches to repository interoperation mainly focus on a uniform data model for software catalog records [8]. However, there are more other barriers than the inconsistent catalog formats among repositories. In this paper, we focus on alleviating semantic inconsistency among representation methods in reuse repositories, and propose a novel method based on analyzing the relevancy of the terms in these representation methods. In our approach, co-occurrent components are used to calculate the basic correlations between the terms, and the correlations are improved through considering the frequency of the term in each representation method and the number of components associated with each term. Based on the *IRMR* method, the users can retrieve components from multiple repositories via a single representation view. The users' queries represented in the vocabulary of one repository are mapped automatically to the queries represented in the vocabulary of another repository. Furthermore, we present a prototype system named the *Virtual Repository supporting Semantic Interoperation (VRSI).*

Our approach has been applied to two different types of real independent repository pairs: 1) ComponentSource (denoted as *CS*) and Download.com (denoted as *DL*), and 2) Open-Components (denoted as *OC*) and SourceForge (denoted as *SF*). The experimental results of the proposed approach show significant improvement in terms of searching effectiveness compared with other techniques such as the word-matching-based method [3] in information retrieval.

2. MOTIVATION

2.1 Information Islands in Reuse Repositories

Many empirical reports about reuse in industry show that the users repeatedly re-implement the same function because they are unaware of these reusable components [16, 21]. It is a key factor for promoting reuse to make components known to the users [46, 33]. Although multiple repositories facilitate the users to find more desired components, they can lead to information islands to embarrass this retrieval.

Suppose repository A is familiar and repository B is unknown to the user, three knowledge levels of retrieving components from the two repositories can be identified (see Fig. 1). The ovals represent the information space of the components in the two repositories, and the rectangle represents the actual information space of the users' desired components (labelled as L3). Generally, the users who are only familiar with repository (A) can retrieve components in L1 easily, but they can hardly reuse the components in the area of (L3-L1) [22]. The components in (L3-L1) become *information islands* [19], which are irretrievable to the users of A without appropriate tools.

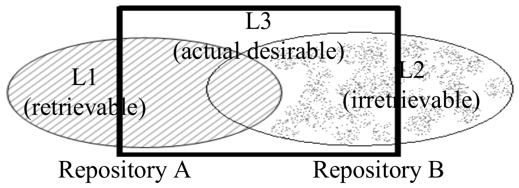

Figure 1: Different levels of users' knowledge about components in two reuse repositories.

2.2 Cognitive Barriers of Multi-Repository Retrieval

It has been widely accepted that component retrieval in a single repository is primarily a cognitive activity beginning with a problem in the mind of the retriever. To get the solution of this problem, the users have to translate the problem into a series of operations of a reuse repository system [36]: from the actual problem to user's intention, from the intention to queries, and from the queries to retrieved components. Each transformation might be a barrier for component retrieval if not being addressed appropriately.

Retrieval of components from multiple repositories is also a cognitive activity. The cognitive model for multi-repository retrieval is shown in Fig. 2. After understanding the retrieval problem, the users must formulate their intentions as queries represented by the vocabulary in each repository. In this retrieval process, three worse cognitive barriers exist than that in single-repository retrieval process.

Repository systems usually provide representation methods in special disciplines by experts to help the users to retrieve and employ the components efficiently [24]. As a result, the vocabulary of one repository might be quite different from the vocabulary of another repository. A representation method in a repository is usually associated with a vocabulary that contains terms [24]. Users should learn the syntax and the semantics of the vocabulary to formulate their intention as queries. Vocabulary learning is a major part of the cognitive barrier to retrieve in reuse repository [7]. It is particularly difficult for the users to retrieve components from multiple repositories, because the users need to learn more vocabularies.

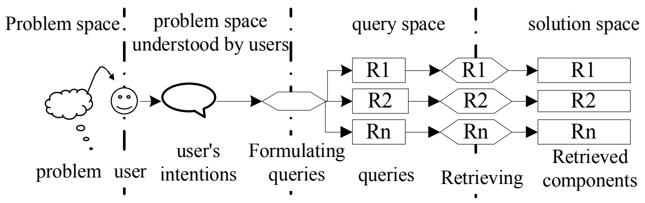

Figure 2: A cognitive model for multi-repository retrieval.

Another barrier related to the difference among vocabularies used in different repositories is the conceptual gap among these vocabularies. For example (see Section 4.1), the same meaning may be represented by entirely different terms, and some terms that share common words may describe different aspects. Learning the vocabulary for one repository may confuse or contradict the learning of another vocabulary.

The users need to formulate their intention as the queries in the system model of the repository after learning the vocabulary. With the unknown system model, the users cannot formulate effective queries [56]. This conceptual gap between the problem space understood by users and the system model of the repository is also a cognitive barrier to retrieve in a reuse repository. It is a big burden for the users to know all the system models in multiple repositories in order to formulate their intention as different queries suitable for different repositories.

2.3 Current Approaches

In the literature, there have been some approaches proposed to alleviate the information islands problem in multiple-repository retrieval. These approaches mainly focus on two levels of interoperation between repositories [10]: 1) the level of catalog information that describes the resources [10, 28, 29, 30]; and 2) the level of the actual resources [14, 51, 5].

In these approaches, the users can browse or search for resources from all the interoperating repositories through a single interface. However, because there still exist some cognitive barriers in multiple-repository retrieval, these approaches cannot fully solve the information islands problem in multiple repositories. With our IRMR method, the users can concentrate on learning the vocabulary and the system model for one repository while they still can retrieve components in other repositories. This way of retrieving components from multiple repositories can effectively reduce the cognitive barriers, and thus further alleviate the information islands problem.

3. IMPROVED RELEVANCY MATCHING AND RANKING METHOD

The basic idea of our approach is to map the queries for one repository (denoted as A) to those for the other (denoted as B) using the correlation between terms in two different repositories. Based on the correlation, a query (q) represented by the terms in A can be automatically transformed to a query (q') represented by the terms in B. The users can use these queries to retrieve components in B transparently. The approach consists of three essential steps:

- **Correlation calculation:** In our approach, the correlations are automatically calculated based on the mutual components in both repositories and recorded in a term-to-term association matrix.
- **Query transformation:** Given a query represented by the terms in one repository, the query is transformed into a query represented by the terms in another repository based on the correlation to retrieve in the repository.
- **Results ranking:** The retrieved results are ranked using the calculated correlation.

3.1 Correlation Calculation

The first step in our approach is to calculate the correlation between the terms in two different repositories. Let H be the number of terms in one repository (denoted as A) and V be the number of terms in the other repository (denoted as B). The term-to-term correlation can be defined as a matrix with H rows and V columns, denoted as $\vec{M}_{AB} = (m_{ij})$, called the **relevancy matrix**. Each element m_{ij} represents the correlation between the i-th term in A and the j-th term in B. Note that this representation is applicable to any two repositories represented by terms using the library and information science methods [24].

3.1.1 *The Matrix Calculating Algorithm*

The relevancy matrix could be specified manually by specialists who are familiar with both repositories [6]. However, this way would be subjective and expensive, since there are dozens of repositories and usually hundreds of terms in each repository. As terms in many repositories may change from time to time, the relevancy matrix may have to be updated accordingly. Frequent manual updates of the relevancy matrix may be particularly burdensome. To address these issues, we propose an algorithm to calculate the relevancy matrix automatically as follows.

Usually there are some components stored in both repositories A and B, denoted as C_{AB}. For example, there are 172 components stored in both ComponentSource and Download.com. These mutual components in both repositories are the only information that we can exploit. The Bayesian network model [45] can be applied to calculate the term-term correlation (see Fig. 3) where the set C_{AB} is the sample space.

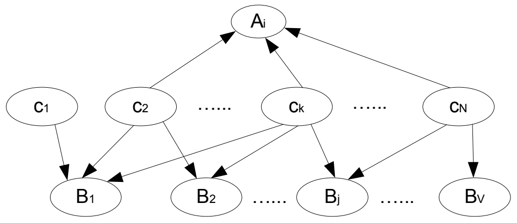

Figure 3: A term-term correlation Bayesian network model.

In this network model, the term A_i in repository A is modelled as a network node associated with a random variable. The value of this variable is 1 whenever A_i completely covers the mutual component space C_{AB}, so $P(A_i)$ is the degree of coverage of the space C_{AB} by A_i. A term B_j in repository B is also modelled as a network node associated with a random variable. $P(B_j)$ is the degree of coverage of the space C_{AB} by B_j. The correlation between terms A_i and B_j is interpreted as a concept matching relationship and reflects the degree of coverage provided to the concept B_j by the concept A_i, i.e. $P(B_j|A_i)$.

By the application of Bayes' theorem, we can write

$$P(B_j|A_i) = P(B_jA_i)/P(A_i)$$

Based on the Bayesian network model,

$$P(B_j|A_i) = \frac{\sum_{\forall\vec{c}}(P(B_jA_i|\vec{c}) \times P(\vec{c}))}{P(A_i)}$$

$\vec{c}$ is a binary vector associated the set C_{AB}, i.e., the value of each c_k is 1 or 0. From the network of Fig. 3, the portions of A_i and B_j in the network are logically separated by instantiation of the mutual component nodes.

$$P(B_j|A_i) = \frac{\sum_{\forall\vec{c}}(P(B_j|\vec{c}) \times P(A_i|\vec{c}) \times P(\vec{c}))}{P(A_i)}$$

We need to specify the conditional probabilities $P(B_j|\vec{c})$, $P(A_i|\vec{c})$, $P(A_i)$ and $P(\vec{c})$.

Let $g_k(\vec{c})$ be a function that returns the value of c_k in a binary N-dimensional vector.

$$\begin{aligned}
\vec{c}_k &= \vec{c}|(g_k(\vec{c}) = 1 \wedge \forall_{h\neq k} g_h(\vec{c}) = 0) \\
\vec{B}_j &= \vec{c}|(g_k(\vec{c}) = 1 \Leftrightarrow c_k \text{ is represented by the term } B_j) \\
\vec{A}_i &= \vec{c}|(g_k(\vec{c}) = 1 \Leftrightarrow c_k \text{ is represented by the term } A_i)
\end{aligned}$$

A straightforward way of calculating $P(B_j|\vec{c})$ is as follows:

$$P(B_j|\vec{c}) = \begin{cases} 1 & \text{if } \vec{c} = \vec{c}_h \wedge g_h(\vec{B}_j) = 1 \\ 0 & \text{otherwise} \end{cases}$$

However, considering that different components in C_{AB} may not contribute equally, we extend the ITF-DF method [47] in information retrieval to calculate $P(B_j|\vec{c})$. Here, we consider two factors for this calculation: 1) the term frequency tf: the number of terms in B representing the component c_h and 2)the component frequency cf: the number of occurrences of term B_j in the whole collection of the mutual components (i.e. C_{AB}). Intuitively, the larger the number of terms in B representing c_h is, the more terms in B are correlated to term A_i through c_h, and thus the less c_h contributes to the correlation of one particular term in B to term A_i. Similarly, the larger the number of occurrences of term B_j in C_AB, the less a particular component in C_{AB} contributes to $P(B_j|\vec{c})$. We summarize in Table 1 some of the notations and their definitions in detail.

N	number of components in C_{AB}
N_i	number of components in C_{AB} represented by A_i
V	number of terms in B
cf_j	component frequency of the term B_j, that is number of components in C_{AB} represented by term B_j
t_h	number of terms in B representing the component c_h
icf_j	inverse component frequency of the term B_j i.e. $icf_j = log\frac{N}{cf_j}$
$tf_{h,j}$	frequency of term B_j representing component c_h, i.e. $tf_{h,j} = \begin{cases} 1/t_h & \text{if } B_j \text{ represents } c_h \\ 0 & \text{otherwise} \end{cases}$
$w_{h,j}$	weight of term B_j representing component c_h i.e. $w_{h,j} = tf_{h,j} \times icf_j$

Table 1: Notations and definitions in calculating the relevancy matrix.

We can have

$$P(B_j|\vec{c}) = \begin{cases} \frac{w_{h,j}}{\sqrt{\sum_{h=1}^{N} w_{h,j}^2}} & \text{if } \vec{c} = \vec{c}_h \wedge g_h(\vec{B}_j) = 1 \\ 0 & \text{otherwise} \end{cases}$$

Further, define

$$P(A_i|\vec{c}) = \begin{cases} 1 & \text{if } \vec{c} = \vec{c}_h \wedge g_h(\vec{A}_i) = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$P(A_i) = \frac{N_i}{N}$$

$$P(\vec{c}) = \begin{cases} \frac{1}{N} & \text{if } \vec{c} = \vec{c}_k \\ 0 & \text{otherwise} \end{cases}$$

We can get the correlation between A_i and B_j:

$$m_{ij} = P(B_j|A_i) = \frac{\sum_{\forall \vec{c}}(P(B_j|\vec{c}) \times P(A_i|\vec{c}) \times P(\vec{c}))}{P(A_i)}$$

$$= \frac{\sum_{h=1}^{N} \frac{w'_{h,j}}{\sqrt{\sum_{h=1}^{N} w_{h,j}^2}}}{\frac{N_i}{N} \times N} = \frac{\sum_{h=1}^{N} w'_{h,j}}{N_i \times \sqrt{\sum_{j=1}^{V} w_{h,j}^2}}$$

where

$$w'_{h,j} = \begin{cases} w_{h,j} & \text{if } c_h \text{ is represented by term } B_j \text{ and } A_i \\ 0 & \text{otherwise} \end{cases}$$

3.1.2 An Example

There are five terms in the representation method of repository A, and the representation method of repository B consists of four terms. The set of mutual components in repositories A and B:$C_{AB} = \{c_1, c_2, c_3, c_4, c_5\}$. They are represented by terms in repository A and B as shown in Table 2.

terms in repositories	mutual components				
	c_1	c_2	c_3	c_4	c_5
A_1		√	√		
A_2	√			√	√
A_3		√		√	
A_4	√	√		√	
A_5				√	√
B_1		√		√	√
B_2	√		√	√	√
B_3	√	√			
B_4				√	

Table 2: An example: mutual components are represented by the terms in repositories A and B.

The following is the steps of calculating the relevancy matrix M_{AB} between two repositories A and B:

$$N = 5, V = 4, N_i = [2, 3, 2, 3, 2], cf_j = [3, 4, 2, 1]$$

$$t_h = [2, 2, 1, 3, 2], icf_j = [0.222, 0.097, 0.398, 0.699]$$

$$tf = \begin{pmatrix} 0 & 0.5 & 0.5 & 0 \\ 0.5 & 0 & 0.5 & 0 \\ 0 & 1 & 0 & 0 \\ 0.33 & 0.33 & 0 & 0.33 \\ 0.5 & 0.5 & 0 & 0 \end{pmatrix}$$

$$w_B = \begin{pmatrix} 0 & 0.0485 & 0.199 & 0 \\ 0.111 & 0 & 0.199 & 0 \\ 0 & 0.097 & 0 & 0 \\ 0.074 & 0.032 & 0 & 0.233 \\ 0.111 & 0.0485 & 0 & 0 \end{pmatrix}$$

$$M_{AB} = \begin{pmatrix} 0.33 & 0.39 & 0.35 & 0 \\ 0.36 & 0.35 & 0.24 & 0.33 \\ 0.54 & 0.13 & 0.35 & 0.5 \\ 0.36 & 0.22 & 0.47 & 0.33 \\ 0.54 & 0.33 & 0 & 0.5 \end{pmatrix}$$

In M_{AB}, the correlation between the terms A_1 and B_2 is 0.5, while the correlation between A_1 and B_4 is 0, etc. A real example is shown in Table 5 with further explanation in Section 4.3. In general, $M_{AB} \neq M_{BA}$.

3.2 Query Transformation

Once the relevancy matrix M_{AB} is set, we can translate a query represented by the terms in A to a query in B with corresponding terms.

A query can be represented by terms using different retrieval models, such as the boolean, the vector, and the probabilistic models [3]. In this paper, the query is based on the boolean model, i.e., a query is composed of index terms linked by three connectives: *not*, *and*, *or*.

Let $q(A_1, A_2, ..., A_l)$ be the query represented by the terms in repository A. Based on the relevancy matrix $\vec{M}_{AB}$, each term A_i in the query $q(A_1, A_2, ..., A_l)$ is mapped to the terms in repository B:

$$B[i] = [\ B_{i1}, B_{i2}, ..., B_{ip_i}\ ]$$

Each $m_{ij} > 0$ indicates that the term A_i is related to the term B_j, i.e., the term B_j should be included into the array $B[i]$ if $m_{ij} > 0$.

Let $q'(B_1, B_2, ..., B_k)$ be the transformed query represented by the terms in repository B. For each A_i in the query q, replace it with $(B_{i1} \vee B_{i2} \vee ... \vee B_{ip_i})$, i.e.,

$$q'(B_1, B_2, ..., B_k) = q(B_{11} \vee B_{12} \vee ... \vee B_{1p_1}, \\ B_{21} \vee B_{22} \vee ... \vee B_{2p_2}, \\ ..., B_{l1} \vee B_{l2} \vee ... \vee B_{lp_l})$$

As a result, the transformed query q' can be used to search components in B.

3.3 Results Ranking

3.3.1 The Ranking Algorithm

Let $C_q = \{c_1, c_2, c_3, ..., c_n\}$ be the set of the retrieved components in B by the transformed query q' from query q. Based on the relevancy matrix $\vec{M}_{AB}$, for each component c_i in the set C_q, we can calculate the weight $w_q(c_i)$.

Since this relevancy matrix is the only information that we can use to rank the retrieved components, the set of terms in B is the sample space for ranking. The Bayesian network model [45] can also be applied to rank the components(see Fig. 4), which is similar as Fig. 3. $P(c_k|A_i)$ can be adopted as the rank of the component c_k with respect to the term A_i in the query q.

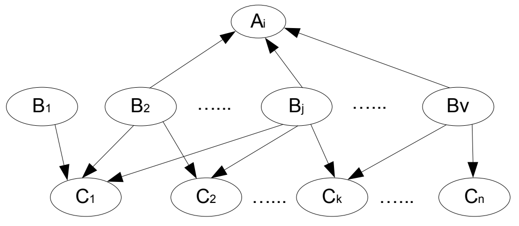

Figure 4: A ranking Bayesian network model.

By the application of Bayes' theorem, we can similarly write:

$$P(c_k|A_i) = P(c_k A_i)/P(A_i)$$

$$P(c_k|A_i) = \frac{\sum_{\forall \vec{B}}(P(c_k|\vec{B}) \times P(A_i|\vec{B}) \times P(\vec{B}))}{P(A_i)}$$

We need to specify the conditional probabilities $P(c_k|\vec{B})$, $P(A_i|\vec{B})$, $P(A_i)$ and $P(\vec{B})$.

Let V be the number of terms in B. Define $g_k(\vec{c})$ as a function that returns the value of c_k in a binary V-dimensional binary vector $\vec{c}$. Let Num_i be the number of terms in B transformed to term A_i in A. Also, let $IN(c_k)$ be the set of the terms in B representing the component c_k.

Let,

$$\begin{aligned}
\vec{B}_j &= \vec{B}|(g_j(\vec{B}) = 1 \wedge \forall_{h \neq j} g_h(\vec{B}) = 0) \\
\vec{c}_k &= \vec{B}|(g_j(\vec{B}) = 1 \Leftrightarrow c_k \text{ is represented by the term } B_j) \\
\vec{A}_i &= \vec{B}|(g_j(\vec{B}) = 1 \Leftrightarrow m_{i,j} > 0)
\end{aligned}$$

Therefore,

$$P(c_k|\vec{B}) = \begin{cases} 1 & \text{if } \vec{B} = \vec{B}_j \wedge g_j(\vec{c}_k) = 1 \\ 0 & \text{otherwise} \end{cases}$$

Further, define

$$\begin{aligned}
P(A_i|\vec{B}) &= \begin{cases} m_{ij} & \text{if } \vec{B} = \vec{B}_j \wedge g_j(\vec{A}_i) = 1 \\ 0 & \text{otherwise} \end{cases} \\
P(A_i) &= \frac{Num_i}{V} \\
P(\vec{B}) &= \begin{cases} \frac{1}{V} & \text{if } \vec{B} = \vec{B}_j \\ 0 & \text{otherwise} \end{cases}
\end{aligned}$$

So, we can get the weight of c_k with respect to the term A_i:

$$\begin{aligned}
w_{A_i}(c_k) &= P(c_k|A_i) = \frac{\sum_{\forall \vec{B}}(P(c_k|\vec{B}) \times P(A_i|\vec{B}) \times P(\vec{B}))}{P(A_i)} \\
&= \frac{\sum_{B_j \in IN(c_k)} m_{ij}}{\frac{Num_i}{V} \times V} = \frac{\sum_{B_j \in IN(c_k)} m_{ij}}{Num_i}
\end{aligned}$$

Based on the probability theory, the algorithm of calculating the weight $w_q(c_i)$ is a recursive algorithm consisting of the following cases:

If $q = A_i$, the weight $w_q(c_k)$ is defined as

$$w_q(c_k) = \frac{\sum_{B_j \in IN(c_k)} m_{ij}}{Num_i}$$

If $q = \neg A_i$, the weight $w_q(c_k)$ is defined as

$$w_q(c_k) = 1 - w_{q'}(c_k)$$

where $q' = A_i$;

If $q = \bigwedge_{j=0}^{r} q_j$, the weight $w_q(c_k)$ is the product of the weights $w_{q_j}(c_k)$, i.e.,

$$w_q(c_k) = \prod_{j=0}^{r} w_{q_j}(c_k)$$

If $q = \bigvee_{j=0}^{r} q_j$, $q = \neg \bigwedge_{j=0}^{r}(\neg q_j)$, thus $w_q(c_k)$ is

$$w_q(c_k) = 1 - (\prod_{j=0}^{r}(1 - w_{q_j}(c_k)))$$

3.3.2 An Example

The query q represented by the terms in repository A is

$$q = A_5 \wedge (A_2 \vee \neg A_1)$$

There are 7 components in repository B:

$$\{c_1, c_2, c_3, c_4, c_5, c_6, c_7\}$$

They are represented by terms in repository B (Table 3).

terms in B	**components in B**						
	c_1	c_2	c_3	c_4	c_5	c_6	c_7
B_1		✓		✓	✓		
B_2	✓		✓	✓	✓	✓	
B_3	✓	✓					✓
B_4				✓		✓	

Table 3: An example: components are represented by the terms in repository B.

The following matrix shows the relevancy matrix M_{AB} calculated in Section 3.1.2 for the two repositories A and B.

$$M_{AB} = \begin{pmatrix} 0.33 & 0.39 & 0.35 & 0 \\ 0.36 & 0.35 & 0.24 & 0.33 \\ 0.54 & 0.13 & 0.35 & 0.5 \\ 0.36 & 0.22 & 0.47 & 0.33 \\ 0.54 & 0.33 & 0 & 0.5 \end{pmatrix}$$

The query q is translated as:

$$q' = (B_1 \vee B_2 \vee B_4) \wedge ((B_1 \vee B_2 \vee B_3 \vee B_4) \vee \neg(B_1 \vee B_2 \vee B_3))$$

As a result, the set of the retrieved components for query q in repository B is $c_1, c_2, c_3, c_4, c_5, c_6$.Table 4 shows the cases of ranking components in the IRMR method. Finally, the retrieved components are sorted as: $c_4, c_5, c_6, c_2, c_3, c_1$.

query	**retrieved components in B**					
	c_1	c_2	c_3	c_4	c_5	c_6
A_5	0.11	0.18	0.11	0.46	0.29	0.28
A_2	0.15	0.15	0.09	0.26	0.18	0.17
A_1	0.25	0.23	0.13	0.24	0.13	0.12
$\neg A_1$	0.75	0.77	0.87	0.76	0.87	0.88
$A_2 \vee \neg A_1$	0.79	0.8	0.88	0.82	0.89	0.9
$A_5 \wedge (A_2 \vee \neg A_1)$	0.09	0.14	0.10	0.38	0.26	0.25

Table 4: An example: results of ranking components based on the improved relevancy matching and ranking method.

3.4 VRSI: A Virtual Repository supporting Semantic Interoperation

The VRSI system is a prototype of a virtual repository supporting semantic interoperation based on the IRMR method discussed in the previous section. VRSI catalogs and refers to software maintained elsewhere. It consists of three functional modules: the *calculator*, the *translator* and the *sorter*, and the *relevancy matrix repository*. The *calculator* module calculates M_{AB} and M_{BA} based on the catalog information in the physical repositories and stores them in the *relevancy matrix repository*. In the *translator* module, users' queries represented by the representation method of their familiar repository are automatically translated to the queries represented by the representation method of other repositories. These queries are used to retrieve components in the physical repositories separately. At last, the *sorter* module ranks the retrieved components.

In summary, the core part (the query space) of the cognitive model of the retrieval process in VRSI has been changed. Fig. 5 shows the query space in changed cognitive model in which (a) is the space in Fig. 2 and (b) is the space in VRSI. The users do not need to formulate the intentions as queries for each repository individually in the changed cognitive model. They only need to formulate the intentions as the query represented by the vocabulary in their familiar repository R_i instead. The *translator* can automatically

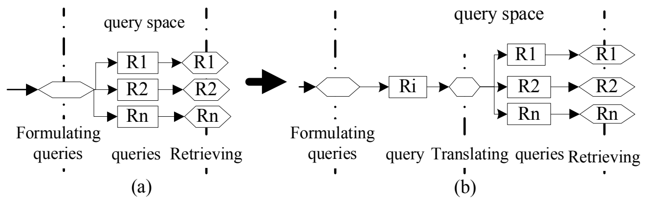

Figure 5: The changed cognitive model:(a) is the query space in Fig. 2, (b)is the query space in VRSI.

translate the query in R_i to queries for each repository (R_1 to R_n) to retrieve the components.

4. EXPERIMENTAL STUDY

4.1 Data Sets

To evaluate our approach, we applied it to two different kinds of real independent repository pairs. The first experiment was performed on two software repositories mostly for software users (not software developers) who wanted to obtain already developed tools or systems, i.e. ComponentSource (CS) [15] and Download.com (DL) [17]. There were 102 terms in vocabulary of CS, 245 terms in DL and 172 mutual components stored in both repositories. The terms in CS were organized as a controlled keyword method. The terms in DL formed an enumerated method, in which each term represented a subject area and an area was broken into mutually exclusive, usually hierarchial classes. Each component in CS can be represented by more than one term. However, each component in DL was represented by only one term that was the leaf node in the classification tree.

The second experiment was performed on two repositories for software developers, i.e. Open-Components (OC) [43] and SourceForge (SF) [50]. There were 30 terms in OC, 184 terms in SF, and 325 mutual components in both repositories. The terms in OC and SF both formed enumerated methods. Each component in OC or SF can be represented by more than one term which was the leaf nodes in the classification trees.

4.2 Experiments

For each repository pair, 2/3 mutual components were used to calculate the relevancy matrices. For CS and DL, 114 mutual components were randomly selected as the training set to calculate the (102×245) relevancy matrix $M_{CS,DL}$ and (245×102) relevancy matrix $M_{DL,CS}$. For OC and SF, 215 components were randomly selected from these 325 components as the training set to calculate the (30×184) relevancy matrix $M_{OC,SF}$ and (184×30) relevancy matrix $M_{SF,OC}$. The remaining components were used for analyzing the search effectiveness, called test set(denoted as C_t).

Each single term was used as a query to retrieve components in C_t. Since the number of terms in each repository was different, the number of generated queries was also different, i.e. 27 queries in CS to retrieve components in DL, 23 queries in DL to retrieve in CS, 20 queries in OC to retrieve in SF, and 57 queries in SF to retrieve in OC. Note that only the terms referred in C_t were used to obtain the results of our experiments.

The results of our method were compared with the standard keyword-based matching method [3] on two types of searching effectiveness: 1) the average precision versus recall, and 2) the average precision at seen relevant components (see [3] pp.73–81). Since each term in the representation methods was a noun phrase (such as "internet communication"), there were only few same terms in different repositories. We adapted the standard keyword-based matching method as follows. In the adapted method, if a term t' in repository B shared some words with term t in repository A, the term t' was considered to be equal to term t in the query (such as "internet communication" and "internet"). This method was referred to as the *word matching* method in this paper. We also used standard pre-processing techniques in information retrieval, such as elimination of stopwords (such as articles and connectives) and use of stemming (which reduced distinct words to their common grammatical root) in this adaption.

4.3 Relevancy Matrix

Table 5 shows some selected rows and columns of the relevancy matrix $M_{CS,DL}$. Since the representation methods in the two repositories are quite different, the same meaning may be represented by entirely different terms, such as the term "training courses" in CS and the term "home & education" in DL, between which the correlation is 1. Some terms that have same words may describe different aspects, such as "internet communication" in CS which usually represents the components for internet communication software development and "internet" in DL which represents the components used on the internet such as web browsers. This behavior partially demonstrates that why the word matching based method cannot achieve good results for retrieving across different repositories. Some other terms are similar, such as "web site" and "internet" (the correlation is 0.235385), "web site" and "software developer" (the correlation is 0.307692).

terms in CS	**terms in DL**		
	home & education	internet	Software developer
training courses	1	0	0
web site	0	0.235385	0.307692
internet communication	0	0.0792308	0.556923

Table 5: Some selected rows and columns of the relevancy matrix $M_{CS,DL}$.

4.4 Experimental Results and Analysis

4.4.1 *Average Precision versus Recall*

Recall is the ratio of the relevant components retrieved by the query q (denoted as $|R_a|$) over the number of relevant components (denoted as $|R_q|$) in the repository, i.e.,

$$Recall_q = \frac{|R_a|}{|R_q|}$$

Precision is the ratio of the relevant components retrieved (denoted as $|R_a|$) over the total number of components retrieved (denoted as $|A_q|$), i.e.,

$$Precision_q = \frac{|R_a|}{|A_q|}$$

The recall versus precision curve is based on 11 standard recall levels, which are 0%, 10%, 20%, ... , and 100%. To evaluate the retrieval performance of a method over all test queries, we average the precision at each recall level r as follows:

$$\overline{P}(r) = \sum_{i=1}^{N_q} \frac{P_i(r)}{N_q}$$

where $\overline{P}(r)$ is the average precision at the recall level r, N_q is the number of queries used, and $P_i(r)$ is the precision at recall level r for the i-th query.

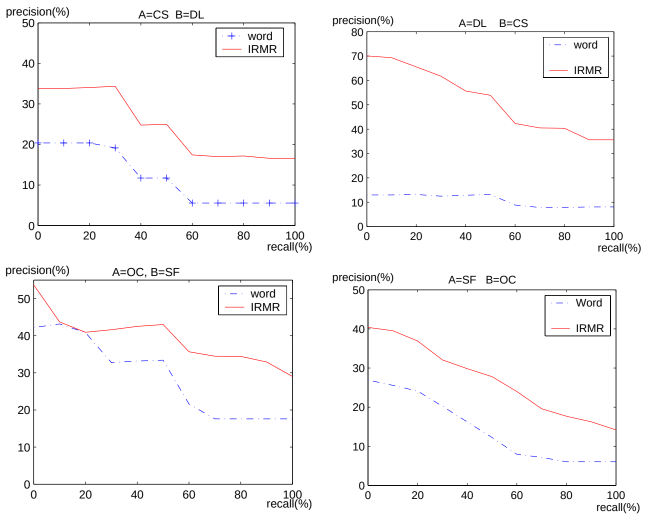

Figure 6: Average precision versus recall curves for the word matching method and our Improved Relevancy Matching and Ranking (IRMR) method.

Fig. 6 shows the average precision versus recall curves for the word matching method and our IRMR method. The curves for our IRMR method are all beyond the curves for the word matching method. This result shows that our IRMR method has higher precision at all recall levels than the word matching method.

It is interesting to note that sometimes both precision and recall increase in the curve (see Fig. 6). This behavior is practically possible [12]. For a specific query q, $|R_q|$ is constant. If $|R_a|$ and $|A_q|$ increase simultaneously, both the precision and recall may increase. For example, in our experiment, the set of components relevant to the given query $q =$ 'databases & networks' in DL and the ranked retrieved components (R) in CS of our IRMR method are:

$$\begin{aligned} R_q &= \{68, 70, 71, 72, 75\} \\ R &= \{68, 60, 37, 70, 71, 72, 75, 82, 87, 93\} \end{aligned}$$

Then, the recall and precision are:

$$\begin{aligned} recall &= 20\%, 20\%, 20\%, 40\%, 60\%, \\ &\quad 80\%, 100\%, 100\%, 100\%, 100\% \\ precision &= 100\%, 50\%, 33.33\%, 50\%, 60\%, \\ &\quad 66.67\%, 71.43\%, 62.5\%, 55.56\%, 50\% \end{aligned}$$

When recall increases from 40% to 100%, precision increases from 50% to 71.43%. It means that the average precision and recall may also increase simultaneously.

Furthermore, according to Buckland and Gey [12], simultaneous increase of recall and precision should be regarded as a merit despite of its rarity. After all, the overall trend of the curve of our IRMR in any case is that precision declines as recall increases, which well conforms to the common sense.

4.4.2 *Average Precision at Seen Relevant Components*

The average precision at seen relevant components is a single value representation of the ranking. It is the mean precision obtained after each new relevant component is observed [3]. For instance, consider the example in Section 4.4.1. The precisions after each new relevant components is observed are $1, 0.5, 0.6, 0.6667, 0.7143$ by our IRMR method. Then the average precision at seen relevant components is $(1 + 0.5 + 0.6 + 0.6667 + 0.7143)/5$ or 69.62%. This measure favors systems that can retrieve relevant components quickly (i.e., relevant components appear early in the ranking).

Table 6 shows the results of the mean precision at seen relevant components for the word matching method and the IRMR method. First, the results show that the IRMR method can yield higher mean precision for any of the four experiments. Second, it also can be shown from the results that the IRMR method is better in most cases of each individual query, which indicates that our method outperforms the word matching method stably. Third, the IRMR method has much fewer zero numbers for all the individual queries (73 for the word matching and 7 for the IRMR method). This demonstrates that IRMR is much more reliable than the word matching method, which is much more likely to retrieve nothing useful at all. In summary, our IRMR method is quicker, more stable, and more reliable than the word matching method based on our experiments.

Furthermore, we analyzed the variance of the average precision at seen relevant components for each method using the Sign Test [27]. The sign test is one kind of the non-parametric alternatives to the t-test, which compares the magnitude of the difference between methods to the variation among the differences. The sign test looks only at the sign of the difference ignoring its magnitude completely.

Let X_i and Y_i be the average precision at seen relevant components of the IRMR method and the word matching method for query i in each experiment, and define $D_i = Y_i - X_i$. These methods assume that the model is additive, i.e. $D_i = \theta + \varepsilon_i$, where the errors ε_i are independent. The null hypothesis is that the difference between the methods does not exist, i.e. $\theta = 0$. Define the statistics n_+ and n_- as the number of queries with $D_i > 0$ and $D_i < 0$ respectively, and let $N = n_+ + n_-$. Any cases where $D_i = 0$ are ignored. Let ν_α be the maximal integer that satisfies

$$\sum_{k=0}^{\nu} C_N^k (\frac{1}{2})^N \leqslant \frac{\alpha}{2}.$$

If $\min(n_+, n_-) < \nu_\alpha$, the testing illustrates that the null hypothesis is rejected at the α level, i.e. there is a significant difference between the methods.

	$A = CS, B = DL$			
Methods	n_+	n_-	N	$\nu_\alpha(\alpha = 0.05)$
Word VS. IRMR	6	20	26	7(reject)
	$A = DL, B = CS$			
Methods	n_+	n_-	N	$\nu_\alpha(\alpha = 0.05)5$
Word VS. IRMR	3	18	21	5(reject)
	$A = OC, B = SF$			
Methods	n_+	n_-	N	$\nu_\alpha(\alpha = 0.05)$
Word VS. IRMR	7	11	18	4(accept)
	$A = SF, B = OC$			
Methods	n_+	n_-	N	$\nu_\alpha(\alpha = 0.05)$
Word VS. IRMR	17	38	55	19(reject)

Table 7: Sign Test analysis of variance for average precision at seen relevant components.

The results of our Statistical Testing are shown in Table 7, in which, they succeed in rejecting the null hypothesis between the IRMR method and the word matching method at the $\alpha = 0.05$ level except the case of $(A = OC, B = SF)$.

query	A=CS, B=DL		A=DL, B=CS		A=OC, B=SF		A=SF, B=OC			
number	**word**	**IRMR**	**word**	**IRMR**	**word**	**IRMR**	**word**	**IRMR**	**word**	**IRMR**
1	0	5.87	22.5	56.97	100	2.8	52.81	65.14	0	28.85
2	0	25	0	100	67	83	100	54.44	75.56	20.92
3	0	15	100	100	2	14	100	100	0	50.74
4	0	11.89	0	62.5	0	100	64.29	44.91	0	33.33
5	0	25	0	100	50	8.3	100	0	50	50
6	0	5.06	0	50	100	17.6	83.39	56.97	0	17.69
7	58.34	34.71	0	100	100	13.3	64.29	15.34	0	27.78
8	100	37.7	0	25	37.4	20.2	0	29.14	0	7.14
9	0	11.03	0	60	0	58.6	0	6.17	0	100
10	0	33.33	0	100	0	29.7	53.33	38.37	0	16.67
11	0	8.59	0	100	100	100	0	60	100	33.33
12	0	43.77	24.29	24.29	5.6	50	0	25	24.31	41.2
13	0	50	33.33	0	50	100	77.84	42.26	36.67	10.1
14	0	100	14.29	0	0	14.1	20	100	88.53	62.42
15	0	21	0	70.16	0	13.3	0	37.12	31.67	39.74
16	0	45.09	0	42.54	20.8	6.4	0	12.5	0	3.48
17	100	0	0	56.86	0	22.2	0	21.36	0	20
18	0	2.22	0	35.01	80.4	6	5.26	5.2	0	5
19	100	100	26.04	69.62	80.4	80.4	0	25	0	16.67
20	0	25	0	33.33	8.5	100	0	7.99	0	3.13
21	0	100	100	0			47.78	54.46	0	3.57
22	0	3.13	0	100			50	8.92	12.5	19.47
23	0	6.62	0	100			63.89	27.81	0	4.35
24	0	4.76					0	35.5	0	100
25	50	0					0	50	8.7	37.5
26	50	9.88					58.33	50	24.29	7.79
27	100	0					0	3.03	0	4.17
28							0	7.05	33.33	14.29
29							0	20.19		
average	20.68	26.84	13.93	60.27	40.11	42.00	25.03	31.28		

Table 6: The average precision at seen relevant components for the word matching method and the IRMR method. (Because there are 57 queries for $\{A = SF, B = OC\}$, we use four columns instead of two.)

Based on the theory in statistics, the statistical testing verified that the IRMR method is significantly better than the word matching method.

5. RELATED WORK

5.1 Repository Interoperation

The impact of reuse repository interoperability upon a software component industry has been identified in the literature [11, 37].

Some previous work of repository interoperation focuses on the description of the assets, especially the common syntax for exchanging the catalog records. BIDM [28] and the supplement [29, 30] specify metadata to interoperate in multiple repositories.

Other work concentrates on building a system or interface to access catalog records in multiple repositories. The NHSE [39] provides a uniform interface to a distributed set of discipline-oriented HPCC repositories. It is a virtual repository, which catalogs and points to software maintained elsewhere [8, 9]. The RIB [44] is a toolkit developed by NHSE for maintainers to create and maintain software catalog records, and exchange these records with other repositories based on BIDM.

DCH [14], Uranus [51] and UDDI[5] work on accessing actual resources in distributed repositories. DCH uses LDAP to operate and manage the distributed repositories while existing a universal representation for components in each distributed node. Uranus uses the degradation functions and a component migration history to tackle the problem. UDDI uses the synchronized replication among the inter-nodes.

Our research differs from the preceding work in tackling the semantic mismatches between different repositories. The only similar work in this area is OML [6], which proposes an architecture using the mediation and ontology technologies to transform queries to retrieve components in multiple repositories. However, the correlations between two terms in two ontology models are provided by the maintainers manually.

5.2 Component Retrieval and Ranking in Reuse Repositories

Another important research issue is to retrieve and rank the components in reuse repositories. Reuse repositories usually apply representation methods [23] to facilitate the users to retrieve and employ components.

In recent years, there are several approaches to assist users to formulate queries or rank retrieval components. Isakowit proposes a method using hypertext to organize retrieval components [32]. Drummond [18] provides an agent to infer users' intentions and advise them. CodeBroker [55] provides an active reuse repository system to formulate query automatically. Component Rank [31] proposes a method to rank components by analyzing actual use relations of components and propagating the significance.

All the preceding work mainly focuses on component retrieval and ranking in one single repository, whereas our research focuses on cross-repository retrieval and ranking. We speculate that the searching effectiveness can be further improved if we incorporate some of these techniques in the preceding work into our ranking algorithm.

5.3 Machine Translation and Cross-Lingual Information Retrieval

Machine translation (MT) attempts to automate the process of translating from one human language to another [2]. In some sense, the central problem addressed in this paper is similar to a MT problem – translating the queries for one repository to queries for another. MT focuses on tackling the syntactic differences between human languages and thus producing correct sentences in the target language with the proper dictionaries.

The cross-lingual information retrieval (CLIR) [42] is closer to our research, which aims at translating queries in one language to queries in another [40]. Unlike MT, CLIR focuses more on word-by-word translation of the queries than on formulation of complex sentences in the target language.

Some previous work [1, 20, 25, 26] proposes approaches to automatically calculating the probability from the parallel texts to translate words between two languages. They can improve the overall retrieval effectiveness by eliminating the ambiguity in choosing the words from the dictionary. Since there is not any available dictionary, these approaches cannot be applied to our approach.

Some other approaches rely on parallel texts for CLIR without any dictionary. Yang et al. [54] investigate several translation models expanded from variants of the traditional vector space model that use parallel texts to calculate two transformation matrices for queries or documents. Some approaches [40, 53] use probabilistic translation models to estimate the probability distribution of a query in the source language over the set of sentences in the target language.

Although these translation models are similar to our relevancy matrix, our technique for calculating the matrix differs from them. First, our technique does not rely on multi-occurrence of a single term that is essential in a document [53] but not the case in reuse repositories. Second, our technique takes both the number of components associated with each term and the number of terms associated with each component into account, whereas Yang et al. [54] only consider the distribution of each term over the documents, and some other approaches [40, 53] only take the number of terms associated with each document into consideration.

5.4 Distributed Information Retrieval

Distributed information retrieval (DIR) [13] is also related to our research. DIR focuses on searching distributed heterogeneous repositories. Typical research topics in this area include common resource description languages [49], schemes for repository selection [41, 48] and merger of ranked lists [52]. Because they mainly focus on the physical rather than semantic differences between repositories, these approaches may be applied to improve our VRSI system.

6. DISCUSSION AND FUTURE WORK

Our IRMR method makes an assumption that there are mutual components stored in two repositories. In most cases, this assumption is realistic, especially noting that if the users want to retrieve components in two repositories, it is expected that these two repositories should have some similarity in contents. In other words, people seldom have the requirement to retrieve components from two repositories whose domains and contents are quite different.

In addition, we are working on applying our IRMR method to more than two repositories to further illustrate its feasibility and effectiveness on reuse. As a result, relevancy information between two repositories could be acquired not only based on their mutual components, but also based on their mutual information with other repositories.

Another problem is how to find the mutual components in repositories, i.e., how to identify similar or identical components. Although the mutual components of two repositories are identified manually in our experiments, there are a number of automatic techniques to tackle this problem. RAIC [34] proposes a technique for identifying similar or identical components via interface relations, functionality relations, domain relations, snapshot relations, security, invocation price, and other aspects. CodeWeb [35] uses name matching and information retrieval-based similarity matching to identify similar components.

We plan to conduct experiments to evaluate our method against the similar approaches for CLIR. As is mentioned above, we also plan to extend our method to repositories based on the vector model and the probabilistic model. Furthermore, we plan to examine whether there are minimum requirements of mutual components to apply our IRMR method.

7. CONCLUSIONS

Although software reuse is an big effort to reduce software crises, it introduces new difficulties for finding and locating components. This paper has proposed a novel method named the *improved relevancy matching and ranking method* for semantic interoperation of reuse repositories. The relevancy matrix of terms in two repositories is calculated automatically based on the mutual components in both repositories to tackle the conception gap between two representation methods. With the relevancy matrix, a query represented by the terms in one repository can be easily transformed to a query in the other repository. Consequently, components distributed in repositories of different types could be retrieved and ranked with a single query interface, which would definitely facilitate component retrieval, management and reuse. We has presented a prototype of a virtual repository supporting semantic interoperation based on the IRMR method. We have also demonstrated the feasibility of our method by applying it to two different types of repository pairs, and the experimental results show that our approach is better than word matching based methods in terms of both accuracy and efficiency.

8. ACKNOWLEDGMENTS

We would like to thank Tao Xie and Professor Shriram Krishnamurthi for their valuable feedback on an earlier version of this paper. This research was sponsored by the State 863 High-Tech Program (SN: 2001AA113070), the National Grand Fundamental Research 973 Program (SN: 2002CB312000), and the National Science Foundation (SN: 60125206 and SN: 60233010) in China.

9. REFERENCES

[1] M. Adriani and C. J. v. Rijsbergen. Term similarity- based query expansion for cross-language information retrieval. In Proceedings of the 3rd European Conference on Research and Advanced Technology for Digital Libraries, pages 311–322, 1999.

[2] D. Arnold, L. Balkan, R. L. Humphreys, S. Meijer, and L. Sadler. Machine Translation: An Introductory Guide. NCC Blackwell, London, 1994.

[3] R. Baeza-Yates and B. Ribeiro-Neto. Modern information retrieval. Addison-Wesley/ACM Press, 1999.

[4] V. Basili, L. Briand, and W. Melo. How reuse influences productivity in object-oriented systems. Comm. of the ACM, 39(10):104–116, 1996.

[5] T. Bellwood, L. Clément, D. Ehnebuske, A. Hately, M. Hondo, Y. Husband, K. Januszewski, S. Lee, B. Mckee, J. Munter, and C. Riegen. UDDI version 3.0 published specification. Technical report, UDDI.org, 2002.

[6] R. M. M. Braga, M. Mattoso, and C. M. L. Werner. The use of mediation and ontology technologies for software component information retrieval. In Proceedings of The Symposium on Software Reusability, pages 19–28, 2001.

[7] F. Brooks. The mythical man-month: essays on softwaer engineering. Addison-Wesley, 1995.

[8] S. Browne, J. Dongarra, S. Green, K. Moore, T. Rowan, R. Wade, G. Fox, K. Hawick, K. Kennedy, J. Pool, R. Stevens, B. Ogson, and T. Disz. The National HPCC software exchange. IEEE Computational Science and Engineering, 2(2):62–69, 1995.

[9] S. Browne, J. Dongarra, K. Hohn, and T. Niesen. Software repository interoperability, UT-CS-96-329. Technical report, Department of Computer Science, University of Tennessee, 1996.

[10] S. Browne, J. Dongarra, J. Horner, P. McMahan, and S. Wells. Technologies for repository interoperation and access control. In Proceedings of the 3rd ACM International Conference on Digital Libraries, pages 40–48, 1998.

[11] S. Browne and J. Moore. Reuse library interoperability and the world wide web. In Proceedings of the 19th International Conference on Software Engineering, pages 684–691, 1997.

[12] M. Buckland and F. Gey. The relationship between recall and precision. Journal of the American Society for Information Science, 45(1):12–19, 1994.

[13] J. Callan. Advances in Information Retrieval, chapter Distributed Information Retrieval, pages 127–150. Kluwer Academic, 2000.

[14] J.X. Ci and W. Tsai. Distributed component hub for reusable software components management. In Proceedings of the 24th International Computer Software and Applications Conference, pages 429–435, 2000.

[15] "http://www.componentsource.com".

[16] P. Devanbu, R. Brachman, P. Selfridge, and B. Ballard. LaSSIE: a knowledge-based software information system. Comm. of the ACM, 34(5):34–49, 1991.

[17] "http://www.download.com".

[18] C. Drummond, D. Ionescu, and R. Holte. A learning agent that assists the browsing of software libraries. IEEE Trans. on Software Engineering, 26(12):1179–1196, 2000.

[19] D. Engelbart. Knowledge-domain interoperability and an open hyperdocument system. In Proceedings of the ACM Conference on Computer-Supported Cooperative Work, pages 143–156, 1990.

[20] M. Federico and N. Bertoldi. Statistical cross-language information retrieval using n-best query translations. In Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 167–174, 2002.

[21] R. Fichman and C. Kemerer. Object technology and reuse: lessons from early adopters. IEEE Software, 14(10):47–59, 1997.

[22] G. Fischer and B. Reeves. Readings in human-computer interaction: toward the Year 2000, chapter Beyond intelligent interfaces: exploring, analyzing and creating success models of cooperative problem solving, pages 822–831. Morgan Kaufmann Publishers: San Francisco, 1995.

[23] W. Frakes and P. Gandel. Representing reusable software. Information and Software Technology, 32(10):653–664, 1990.

[24] W. Frakes and T. Pole. An empirical study of representation methods for reuseable software. IEEE Trans. on Software Engineering, 20:617–630, 1994.

[25] J. Gao, J. Nie, E. Xun, J. Zhang, M. Zhou, and C. Huang. Improving query translation for cross-language information retrieval using statistical models. In Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 96–104, 2001.

[26] J. Gao, M. Zhou, J. Nie, H. He, and W. Chen. Resolving query translation ambiguity using a decaying co-occurrence model and syntactic dependence relations. In Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 183–190, 2002.

[27] D. Hull. Using statistical testing in the evaluation of retrieval experiments. In Proceedings of the 16th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, pages 329–338, 1993.

[28] IEEE. IEEE standard for information technology - software reuse - data model for reusable library interoperability: Basic interoperability data model (BIDM). IEEE std 1420.1, 1995.

[29] IEEE. Supplement to IEEE standard for information technologysoftware reusedata model for reuse library interoperability: Asset certification framework. IEEE std 1420.1a, 1996.

[30] IEEE. IEEE trial-use supplement to IEEE standard for information technologysoftware reusedata model for reuse library interoperability: Intellectual property rights framework. IEEE std 1420.1b, 1999.

[31] K. Inoue, R. Yokomori, H. Fujiwara, T. Yamamoto, M. Matsushita, and S. Kusumoto. Component rank: relative significance rank for software component search. In Proceedings of the 25th International Conference on Software Engineering, pages 14–25, 2003.

[32] T. Isakowitz and R. J. Kauffman. Supporting search for reusable software objects. IEEE Trans. on Software Engineering, 22(6):407–423, 1996.

[33] S. Isoda. Experiences of a software reuse project. Journal of System and Software, 30(3):171–186, 1995.

[34] C. Liu and D. Richardson. Redundant arrays of independent components. Technical report, UCI-ICS-TR-02-09, Information of Computer Science, University of California, USA, 2002.

[35] A. Michail and D. Notkin. Assessing software libraries by browsing similar classes, functions and relationships. In Proceedings of the 1999 International Conference on Software Engineering, pages 463–472, 1999.

[36] H. Mili, F. Mili, and A. Mili. Reusing software: issues and research directions. IEEE Trans. on Software Engineering, 21(6):528–562, 1995.

[37] J. Moore. The impact of reuse library interoperability upon a software component industry. In Proceedings of the 6th Workshop on Institutionalizing Software Reuse, 1993.

[38] "http://www.netlib.org/".

[39] "http://www.netlib.org/NHSE/".

[40] J. Nie, M. Simard, P. Isabelle, and R. Durand. Cross-language information retrieval based on parallel texts and automatic mining of parallel texts from the web. In Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 74–81, 1999.

[41] H. Nottelmann and N. Fuhr. Evaluating different methods of estimating retrieval quality for resource selection. In Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 290–297, 2003.

[42] D. Oard and A. Diekema. Cross-language information retrieval. Annual Review of Information Science, 33:223–256, 1998.

[43] "http://www.open-components.org/".

[44] "http://www.nhse.org/RIB".

[45] B. Ribeiro-Neto and R. Muntz. A belief network model for ir. In Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 253–260, 1996.

[46] S. Rosenbaum and B. DuCastel. Managing software reuse-an experience report. In Proceedings of 17th International Conference on Software Engineering, pages 105–111, 1995.

[47] G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. Information Processing and Management, 24(5):513–523, 1988.

[48] L. Si and J. Callan. Relevant document distribution estimation method for resource selection. In Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 298–305, 2003.

[49] P. Simpson. Query processing in a heterogeneous retrieval network. In Proceedings of the 11th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 359–370, 1988.

[50] "http://sourceforge.net/".

[51] Y. Sun, M. Kao, and C. Lei. A fully distributed approach to repositories of reusable software components. Journal of Information Science and Engineering, 17:147–158, 2000.

[52] J. Xu and J. Callan. Effective retrieval of distributed collections. In Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 112–120, 1998.

[53] J. Xu, R. Weischedel, and C. Nguyen. Evaluating a probabilistic model for cross-lingual information retrieval. In Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 105–110, 2001.

[54] Y. Yang, J. Carbonell, R. Brown, and R. Frederking. Translingual information retrieval: Learning from bilingual corpora. Artificial Intelligence, 103(1-2):323–345, 1998.

[55] Y. Ye and G. Fischer. Supporting reuse by delivering task-relevant and personalized information. In Proceedings of the 22nd International Conference on Software Engineering, pages 513–523, 2002.

[56] Y. Ye, G. Fischer, and B. Reeves. Integrating active information delivery and reuse repository systems. In Proceedings of the 8th ACM SIGSOFT Symposium on Foundations of Software Engineering, pages 60–68, 2000.