CodeBroker: An Active Reuse Repository System

Yunwen Ye
Mar 15, 2004
Software reuse

- **Definition**
  - Creating new software systems with existing artifacts

- **Reusable artifacts**
  - **Code artifacts**
    - macros, functions, *methods, classes*, subsystems, systems
  - **Non-code artifacts**
    - analyses, designs, test plans and cases, domain models
  - **Knowledge**
    - program idioms, program plans, design patterns, software architecture styles, domain knowledge

- **Reuse repository systems**
  - Supporting reuse activities
Why reuse?

- Increased productivity
  - Reduced development time
  - Reduced cognitive load
  - Reduced testing time

- Increased quality
  - Fewer bugs

- Enhanced evolvability and maintainability
Reuse process (sLCMS)
Research problems

- No attempt to reuse (*Location*)
  - Information islands
    - Not aware of the existence of reusable components
  - Perceived low reuse utility (benefits/cost)
    - High cost of locating components
- Unable to locate the component (*Location*)
  - Situation model vs. system model
- Unable to use the component (*Comprehension*)
User’s knowledge about a reuse repository

L4: System Model

L3: Belief

L2: Vaguely Known

L1: Well Known
Reinventing the wheel

- **Have you ever found that you have accidentally implemented a function that is in the library already?**
- Countless times! (tomo)
- Yes this happens often while learning a new language. (prabhu)
- Yes, I have done this a number of times. (mandalia)
- Yup, I wrote a parser in Java that would have been much easier with a StringTokenizer. I'm sure I've done this other times, but that one really gets me (minick).
- Yes. When I was trying to convert a string of numbers into integer, I wrote a function to do it. Later on I found out there is function atio in C library to the exactly the same thing (jing).
- Probably many times, but how would I know? (Jon Marbach).
Reinventing the wheel

- Have you ever found that you have accidentally implemented a function that is in the library already?
- Not yet (jackson)
- I cannot remember ever implementing a function that was already in the library. (deriggi)
- No, but I have never really checked this out (Serina Croll).
Reinventing the wheel

“Conversations with developers revealed several cases in which programmers, unaware of a virtual machine primitive for an operation, repeatedly reimplemented the same operation--in one case, ten times.” [Devanbu, 1991]

Reusable objects demand proper advertisement [Walton, 1992]

“We have discovered that ‘marketing’ the components in the CSL is just as important as providing the correct technologies for users in Schlumbeger Oilfield Services products. [Rosenbaum 1995]

It happens that we develop functions when they exist and we do not realise it. [Coulange 1997]

“I could be creating a method that does exactly the same thing somebody else’s does ... even though we have access to each other’s code. We might call them different names and we might have a bit different way of doing it, but we’re still doing the same thing.” [Fichman, Kemerer, 1997]
Proposed solution

- Active component repository systems
  - Overcoming the limits of browsing and searching
  - Supporting information delivery

- Benefits
  - Reusing unknown components
  - Reduced locating cost
  - Seamless integration with programming environment
Challenges in active reuse repository systems

- L3: Belief
- L2: Vaguely Known
- L1: Well Known
- L4: System Model

Task-relevant information
/** This class simulates the process of card dealing. Each card is represented with a number from 0 to 51. The program should produce a list of 52 cards, as results from a human card dealer */
public class CardDealer {
    static int [] cards=new int[52];
    static {
        for (int i=0; i<52; i++) cards[i]=i;
    }
    /** Create a random number between two limits */
    public static int getRandomNumber (int from, int to) {

---

/** CardDealer.java 10-05 02:08 PM 0.97 (JDE)--L10--All----------------------*/
/* An example for getInt written by yunwen "Fri Oct  5 14:00:58 2001"*/
import com.objectspace.jgl.util.*;
/** Roll a die and print the probability of each number's occurrence */
public class DiceRoller {
    final static int times=10000;
    public static void main(String args[]) {
        int[] distribution=new int[6];
        int p;
        for (int i=0; i<times; i++) {
            p = Randomizer.nextInt(1, 6);
            distribution[p-1]++;
        }
        System.out.println("(Number, Occurrences, Probability)\n");
        *CB-Example*/(/home/yunwen/java/examples/DiceRoller.java) (JDE)--L10--Top--
1 0.89 getInt Generate a random number using the default generator
2 0.78 getLong Generate a random number using the default generator
3 0.78 nextInt Generates a random integer between
4 0.77 nextLong Generates a random integer between
-1;**: **RCL--display* 10-05 02:08 PM 0.97 (ReusableCode)--L1--Top
com.objectspace.jgl.util.Randomizer::int getInt(int lo, int hi)
Inferring the task

- Plan recognition
  - Actions → Inferred goal → Suggested actions or information

- Similarity analysis
Similarity analysis in CodeBroker

Create a random number between two limits

```
int <- int x int
```

Signature Matching
Information Retrieval

Fetcher

Listener

Situation A

Create a random number using the default generator

```
int <- int x int
```

```java
/** This class simulates the process of card dealing. Each card is represented with a number from 0 to 51. The program should produce a list of 52 cards, as results from a human card dealer. */
public class CardDealer {
    static int[] cards = new int[52];
    static {
        for (int i = 0; i < 52; i++) cards[i] = i;
    }
    /** Create a random number between two limits */
    public static int getRandomNumber(int from, int to) {
        return (int) (Math.random() * (to - from + 1)) + from;
    }
}
```

```java
/* An example for getInt written by yunwen “Fri Oct 5 14:15:58 2001” */
import com.objectspace.util.*;
/* Roll a die and print the probability of each number’s occurrence */
public class DiceRoller {
    final static int times = 10000;
    public static void main(String[] args) {
        int[] distribution = new int[6];
        int p;
        for (int i = 0; i < times; i++) {
            p = Randomizer.getInt(1, 6);
            distribution[p-1]++;
        }
        System.out.print("(Number, Occurrence, Probability)");
        for (int i = 0; i < 6; i++) {
            System.out.print("");
            System.out.print(distribution[i]);
            System.out.print("");
        }
    }
}
```
The rationale

- Three aspects of a program
  - Concept
    - The functionality of the program
    - Semantic information
    - Revealed in *comments*, identifiers, ...
  - Constraint
    - Execution environment
    - Syntactic information
    - Revealed in *signatures*, protocols, ...
  - Code
    - The implementation

- The assumption
  - Similar concept + compatible signature $\rightarrow$ reusable code
Basic information retrieval (IR) techniques

- Information retrieval: Finding similar documents based on the commonality of terms
  - Documents and queries are represented by term vectors
    \[ D_j = (f_{1,j}, f_{2,j}, \ldots, f_{N,j}) \]
  - Similarity is the distance between two vectors
    \[
    \text{Similarity} (Q, D) = \frac{\sum_{i=1}^{n} Q[i] \times D[i]}{\sqrt{\sum_{i=1}^{n} Q[i]^2 \times \sum_{i=1}^{n} D[i]^2}}
    \]

Term space: (factor information help human operation retrieval system)

<table>
<thead>
<tr>
<th>Contents</th>
<th>Vector</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q human factors in information retrieval system</td>
<td>(1 1 0 1 0 1 1)</td>
<td></td>
</tr>
<tr>
<td>D1 factor factor factor human human retrieval system</td>
<td>(3 0 0 2 0 1 1)</td>
<td>7/75^{0.5}=0.80</td>
</tr>
<tr>
<td>D2 information operation retrieval retrieval</td>
<td>(0 1 0 0 1 2 0)</td>
<td>0.55</td>
</tr>
<tr>
<td>D3 factor help help retrieval</td>
<td>(1 0 2 0 0 1 0)</td>
<td>0.37</td>
</tr>
</tbody>
</table>
LSA: Improved IR

- Latent semantic analysis
  - Addressing the vocabulary mismatch problem (people use different names to refer to the same concept)
  - Creating a semantic space with a large amount of documents

\[
\begin{pmatrix}
    w_{1,1} & w_{1,2} & \cdots & w_{1,M} \\
    w_{2,1} & w_{2,2} & \cdots & w_{2,M} \\
    \vdots & \vdots & \ddots & \vdots \\
    w_{N,1} & w_{N,2} & \cdots & w_{N,M}
\end{pmatrix}
\begin{pmatrix}
    t_{1,1}^{(0)} & t_{1,2}^{(0)} & \cdots & t_{1,r}^{(0)} \\
    t_{2,1}^{(0)} & t_{2,2}^{(0)} & \cdots & t_{2,r}^{(0)} \\
    \vdots & \vdots & \ddots & \vdots \\
    t_{N,1}^{(0)} & t_{N,2}^{(0)} & \cdots & t_{N,r}^{(0)}
\end{pmatrix}
\begin{pmatrix}
    s_{1,1} & 0 & \cdots & 0 \\
    0 & s_{2,2} & \cdots & 0 \\
    \vdots & \vdots & \ddots & \vdots \\
    0 & 0 & \cdots & s_{r,r}
\end{pmatrix}
\begin{pmatrix}
    d_{1,1}^{(0)} & d_{1,2}^{(0)} & \cdots & d_{1,M}^{(0)} \\
    d_{2,1}^{(0)} & d_{2,2}^{(0)} & \cdots & d_{2,M}^{(0)} \\
    \vdots & \vdots & \ddots & \vdots \\
    d_{r,1}^{(0)} & d_{r,2}^{(0)} & \cdots & d_{r,M}^{(0)}
\end{pmatrix}
\]

Reducing the singular vectors

\[
\hat{X} =
\begin{pmatrix}
    t_{1,1} & t_{1,2} & \cdots & t_{1,k} \\
    t_{2,1} & t_{2,2} & \cdots & t_{2,k} \\
    \vdots & \vdots & \ddots & \vdots \\
    t_{N,1} & t_{N,2} & \cdots & t_{N,k}
\end{pmatrix}
\begin{pmatrix}
    s_{1,1} & 0 & \cdots & 0 \\
    0 & s_{2,2} & \cdots & 0 \\
    \vdots & \vdots & \ddots & \vdots \\
    0 & 0 & \cdots & s_{k,k}
\end{pmatrix}
\begin{pmatrix}
    d_{1,1} & d_{1,2} & \cdots & d_{1,M} \\
    d_{2,1} & d_{2,2} & \cdots & d_{2,M} \\
    \vdots & \vdots & \ddots & \vdots \\
    d_{k,1} & d_{k,2} & \cdots & d_{k,M}
\end{pmatrix}
\]
Probabilistic IR model

- Adding weights to each term
  \[ D_j = (t_{1,j}, t_{2,j}, ..., t_{N,j}) \]
  \[ t_{i,j} = TRW_i \times f_{i,j} \]

- Term Relevance Weight
  \[ TRW_i = \log \left( \frac{p_i \times (1-q_i)}{q_i \times (1-p_i)} \right) \]
  - \( p_i \) Probability of the term appearing in relevant documents
  - \( q_i \) Probability of the term appearing in irrelevant documents
Weighting schema in CodeBroker

\[
\text{sim}(Q, D_j) = \sum_{i=1}^{T} \left( \log \frac{N - n_i + 0.5}{n_i + 0.5} \right) \frac{(k_1 + 1)tf_{i,j}}{K + tf_{i,j}} \frac{(k_3 + 1)qtf_i}{k_3 + qt}.
\]

- \(N\) is the number of components
- \(n_i\) is the number of components whose documents contain the term \(t_i\)
- \(T\) is the number of terms in the component collection
- \(tf_{i,j}\) is the frequency of term \(t_i\) in the document of the component \(D_j\)
- \(qtf_i\) is the frequency of term \(t_i\) in the query \(Q\)

\[
K = k_1 ((1 - b) + b \cdot dl_j / \text{avdl})
\]

- \(k_1, k_3, b\) are empirically determined parameters depending on the nature of the document collection. In CodeBroker, \(k_1\) is set to 1.2, \(k_3\) to 1.0, and \(b\) to 0.75.

- \(dl_j\) is the length of document \(D_j\)
- \(\text{avdl}\) is the average length of all documents in the collection
Signature matching determines the constraint compatibility

- Reusable components must be compatible in signature
  - Signature is the syntactic interface of a module (method and class)
  - Improving the precision of retrieval

- Method level match
  - Exact match
    - $\text{Type}_1 \times \text{Type}_2 \rightarrow \text{Type}_3$
    - $\text{Type}_A \times \text{Type}_B \rightarrow \text{Type}_C$
    - $\iff \text{Type}_1=\text{Type}_A \ \text{AND} \ \text{Type}_2=\text{Type}_B \ \text{AND} \ \text{Type}_3=\text{Type}_C$
  - Relaxed match
    - Generalization / Specialization / Reorder
      - $\text{string} \times \text{int} \rightarrow \text{int} \ \text{matches (relaxed)} \ \text{long} \times \text{string} \rightarrow \text{long}$
Signature matching for classes

```java
public class AutomaticReception extends Vector {
    public boolean initialize();
    public void delete();
    public insert(string person);
    public int length();
}
```

```
public class Queue extends Vector {
    public boolean empty();
    public dequeue();
    public enqueue(object item);
    public int size();
}
```

void -> boolean
type string -> void
type object -> void
void -> int
Presenter: tailoring the delivery to larger context and user

**User model**: list of components known to the user

**Discourse model**: list of components from uninterested domains
Discourse models: Improving task-relevance

Discourse models capture the larger context of programming activities

- Representing the interaction history between programmers and CodeBroker
- Removing irrelevant components
- Negative discourse models: specifying what is not of interest to programmers

Example:

```plaintext
(("java.util.zip") ;; a package
 ("java.awt" ("CardLayout"))) ;; a class
```
User models: User-specific delivery

- User models represent programmers’ knowledge on the component repository
  - A list of known components
  - Example:
    ```
    ("java.applet" ("Applet" ("getParameterInfo"))
    ("java.io" ("File" ("exists"
        "11/02/00" "11/10/00"
        "11/11/00")
    ("isAbsolute"
        "11/01/00" "11/10/00"
        "11/11/00")))
    ```
  - Components contained in user models are **not** delivered
Incremental discourse modeling and user modeling

- **Initial** user models
  - Created by analyzing existing user programs

- **Adaptive** user models
  - CodeBroker updates user models automatically when it detects the use of a component in the editor

- **Adaptable** user models and discourse models
  - Using the Skip Components Menu associated with each delivered component

![Skip Components Menu]

- Added to discourse model
- Added to user models
Models in CodeBroker
Retrieval-by-reformulation

- A process for software developers to incrementally develop reuse queries
- Delivered components help developers become familiar with the vocabulary and structure of the repository
  - Change the way of writing the query
  - Limit the search scope by specifying (un)interested packages and classes
The cycle of delivery-browsing-searching

- Delivered components are results of information reconnaissance.
- Possible actions after the delivery:
  - The needed component is delivered
    → Choose the needed one through browsing
  - Too many components are delivered
    → Filter the delivered components
  - The needed one is not delivered
    → Search again through retrieval-by-reformulation
Supporting comprehension and use

Example

Delivery buffer

Mouse click

Search an example

Illustrator

Get further help from the author

Java documentation

```
/** An example for getInt written by yunwen "Fri Oct 5 14:00:58 2001" */
import com.objectspace.jgl.util.*;
/** Roll a die and print the probability of each number's occurrence */
public class DiceRoller {
    final static int times=10000;
    public static void main(String args[]) {
        int[] distribution=new int[6];
        int p;
        for (int i=0; i<times; i++) {
            p = Randomizer.getInt(1, 6);
            distribution[p-1]++;
        }
        System.out.println("(Number, Occurrences, Probability)");
    }
}
```
Evaluating retrieval effectiveness

- **Recall** = \[
\frac{\text{No. of relevant doc. retrieved}}{\text{No. of relevant doc.}}
\]

- **Precision** = \[
\frac{\text{No. of relevant doc. retrieved}}{\text{No. of doc. retrieved}}
\]

- **Results of 19 queries**
  - One-third is relevant

<table>
<thead>
<tr>
<th>Recall</th>
<th>Prob. Precision</th>
<th>LSA Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>45.82</td>
<td>35.77</td>
</tr>
<tr>
<td>10</td>
<td>45.82</td>
<td>31.86</td>
</tr>
<tr>
<td>20</td>
<td>45.82</td>
<td>30.89</td>
</tr>
<tr>
<td>30</td>
<td>41.20</td>
<td>25.62</td>
</tr>
<tr>
<td>40</td>
<td>41.01</td>
<td>20.62</td>
</tr>
<tr>
<td>50</td>
<td>40.74</td>
<td>20.44</td>
</tr>
<tr>
<td>60</td>
<td>37.46</td>
<td>13.86</td>
</tr>
<tr>
<td>70</td>
<td>37.46</td>
<td>13.82</td>
</tr>
<tr>
<td>80</td>
<td>32.71</td>
<td>13.82</td>
</tr>
<tr>
<td>90</td>
<td>32.19</td>
<td>12.32</td>
</tr>
<tr>
<td>100</td>
<td>29.43</td>
<td>12.32</td>
</tr>
</tbody>
</table>
Evaluation experiments

- **Experiment goals:**
  - Observe the effectiveness of CodeBroker in encouraging programmers to reuse
  - Analyze the effectiveness of task inference, discourse models, and user models

- **12 experiments with 5 subjects**
  - Implementing an assigned task with CodeBroker

<table>
<thead>
<tr>
<th>Subjects</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Years of prog. in general</td>
<td>3-4</td>
<td>5-6</td>
<td>8</td>
<td>10+</td>
<td>10+</td>
</tr>
<tr>
<td>Java skill (self-evaluation)</td>
<td>4</td>
<td>7</td>
<td>7-8</td>
<td>10</td>
<td>7</td>
</tr>
</tbody>
</table>
## System assessment

<table>
<thead>
<tr>
<th>Sub</th>
<th>No</th>
<th>total</th>
<th>delivered</th>
<th>breakdown of deliveries</th>
<th>triggered</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>unanticipated (L4-L3)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>anticipated but unknown (L3)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>vaguely known (L2)</td>
<td></td>
</tr>
<tr>
<td>S1</td>
<td>1</td>
<td>10</td>
<td>4</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>S2</td>
<td>3</td>
<td>7</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>S3</td>
<td>6</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S4</td>
<td>9</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S5</td>
<td>11</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Sum</td>
<td>57</td>
<td></td>
<td>20</td>
<td>9</td>
<td>10</td>
</tr>
</tbody>
</table>

**Total**

<table>
<thead>
<tr>
<th>triggered</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
</tr>
</tbody>
</table>
## Role of discourse models

<table>
<thead>
<tr>
<th>Subject</th>
<th>Task</th>
<th>Retrieved#</th>
<th>Added to DM#</th>
<th>Removed by DM#</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>T1</td>
<td>168</td>
<td>1 pkg., 1 class</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>T2</td>
<td>28</td>
<td>1 pkg., 1 class</td>
<td>10</td>
</tr>
<tr>
<td>S2</td>
<td>T3</td>
<td>140</td>
<td>4 methods</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>T3</td>
<td>80</td>
<td>1 pkg.</td>
<td>7</td>
</tr>
<tr>
<td>S4</td>
<td>T5</td>
<td>140</td>
<td>2 pkgs.</td>
<td>68</td>
</tr>
<tr>
<td>Other 7 experiments</td>
<td>872</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- Discourse models removed irrelevant components
- Larger tasks may make programmers add more components to discourse models
Role of user models

- User models removed few components
  - Incomplete user models
  - Most of the delivered components were unknown
- Removed components not reusable
- User models too simple
  - Unforgiving
  - No decaying mechanism

<table>
<thead>
<tr>
<th>Retrieved</th>
<th>Removed</th>
<th>User added</th>
<th>System added</th>
</tr>
</thead>
<tbody>
<tr>
<td>168</td>
<td>15</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>140</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>52</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>160</td>
<td>14</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>60</td>
<td>0</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>60</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>80</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>140</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>100</td>
<td>1</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>420</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Problems found

- Irrelevant components
  - More sophisticated task modeling techniques
- Abstraction mismatch from queries to components
  - Indexing based on usage
- Lack of guidance in refining queries
  - Guidance on the choice of terms
- Lack of configurability
  - More user-friendly interface
- Lack of examples
  - The development of the Illustrator agent
Future research

- **Long-term** user models and their evaluation in natural settings
- **Distributed** CodeBroker supporting software development communities
  - Make programmers aware of reusable components
  - Bring together programmers working on similar programs

![Diagram of CodeBroker and programmers interacting](image)
General lesson: Designing information repository systems

- Two modes of designing and using information repository
  - Filtering the input vs. Filtering the output

[Diagram showing input and output filters for designer, active user, and passive user]
Summary

- Better understanding of cognitive difficulties of component reuse
  - Unknown components
  - Low reuse utility
- A new type of component repository systems
  - Active component repository systems
- Contributions to the design of information repository systems in general
  - Similarity analysis-based task modeling
  - Focusing output filter instead of input filter