Overview of Component Search System SPARS-J

Tetsuo Yamamoto*,Makoto Matsushita**,
Katsuro Inoue**

*Japan Science and Technology Agency
**Osaka University

Outline

- Motivation and research aim
- SPARS-J
  - Outline
  - System architecture
  - Ranking method
  - Each part
    - Analysis part
    - Retrieval part
    - User Interface
- Experiment
- Conclusion and Future work

Motivation

- Reuse of Software Components
  - is a technique of developing new software components by using the components developed in the past.
    - Example of reusable components: source code, document .....
  - improves productivity and quality, and cuts down development cost as a result.
- However, reuse of components is not utilized effectively.
  - A developer doesn't know existence of desirable components.
  - Although there are a lot of components, these components are not organized.

- In order to take advantage of reuse, it is required to manage components and search suitable component easily

Research aim

- We have built the system which have functions as follows
  - Collects software components eagerly without preserving their inherent structures
  - Manages the component information automatically
  - Provides component be suitable for User's request
- Targets
  - Intranet
    - closed software development inside a company
  - Internet
    - Large open source software development web site
      - – SourceForge, Jakarta Project. etc.

Outline

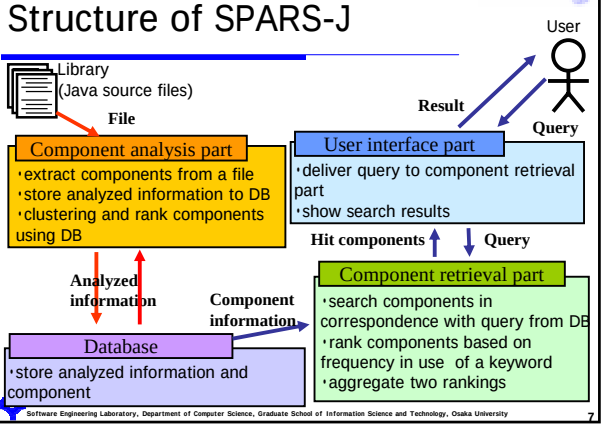
- Motivation and research aim
- SPARS-J
  - Outline
  - System architecture
  - Ranking method
  - Each part
    - Analysis part
    - Retrieval part
    - User Interface
- Experiment
- Conclusion and Future work

SPARS-J

(Software Product Archive, analysis and Retrieval System for Java)

- Java Software Product Archiving, analyzing and Retrieving System
  - Many components are analyzed automatically.
  - A search engine is built based on the analysis information.
  - Component: a source code of class or interface
- Features
  - Keyword search
  - Two ranking methods
    - Frequency in use of a word
    - Use relation
  - Analyzed information
    - Components using/used by a component
    - Package hierarchy

Structure of SPARS-J



Ranking search results

- Ranking method
  1. Component suited to a user request
     - Ranking based on frequency in use of a word
       - ➡ **Keyword Rank (KR)**
  2. Component used mostly
     - Ranking based on component use relation
       - ➡ **Component Rank (CR)**
- We make it high ranking that the component both 1 and 2 are high
  - Search results are shown to aggregate two ranks

Outline

- Motivation and research aim
- SPARS-J
  - Outline
  - System architecture
  - Ranking method
  - Each part
    - Analysis part
    - Retrieval part
    - User Interface
- Experiment
- Conclusion and Future work

Component analysis part

- Extract component and its information from a Java source file
- The process
  - Extract a component
  - Index the component
  - Extract use relations
  - Clustering similar components
  - Rank components based on use relations (CR method)

Extract and index a component

- Extracting component
  - Find class or interface block in a java source file
    - Location information in the file (start line number, end line number)
- Indexing
  - Extract *index key* from the component
    - *Index key*: a word and the kind of it
    - No reserved words are extracted
  - Count frequency in use of the word

```
public final class Sort {
  /* quicksort */
  private static void quicksort(…) {
    int pivot;
      :
    quicksort(…);
    quicksort(…);
  }
}
```

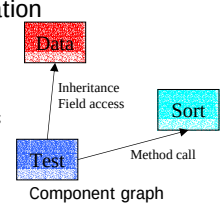
word	kind	
Sort	Class name	1
quicksort	Comment	1
quicksort	Method name	1
pivot	Variable name	1
quicksort	Method call	2
:	:	:

Index key　frequency

Extract use relations

- Extract use relations among components using semantic analysis
- Make component graph from use relations
  - Node: component
  - Edge: use relation

```
public class Test extend Data{
    :
  public static void main(…) {
    :
    Sort.quicksort(super.array);
    :
  }
}
```



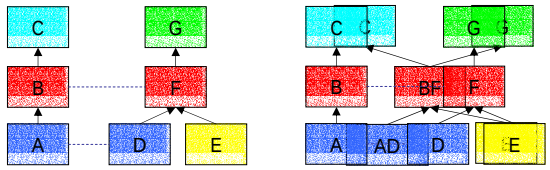
Component graph

Inheritance
Interface implementation
Variable type
Instance creation
Field access
Method call

The kind of use relation

Similar component

- Similar component is copied component or minor modified component
- We merge similar components into single component
- Merged component have use relations that all component before merging have



Component graph　　Clustered component graph

Clustering components

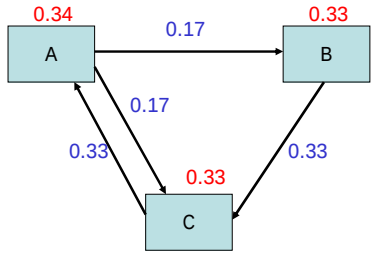
- We measure characteristics metrics to merge components
- The difference ratio of each component metrics
  - Metrics
    - complexity
      - The number of methods, cyclomatic, etc.
      - represent a structural characteristic
    - Token-composition
      - The number of appearances of each token
      - represent a surface characteristic

Ranking based on use relation

- Component Rank (CR)
  - Reusable component have many use relation
    - The example of use is much
    - General purpose component
    - Sophisticated component
  - We measure use relation quantitatively, and rank components
    - The component used by many components is important
    - The component used by important component is also important

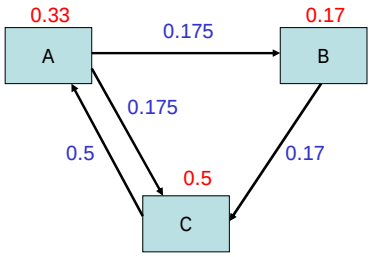
Katsuro Inoue, Reishi Yokomori, Hikaru Fujiwara, Tetsuo Yamamoto, Makoto Matsushita, Shinji Kusumoto: **"Component Rank: Relative Significance Rank for Software Component Search"**, ICSE, Portland, OR, May 6, 2003.

Propagating weights



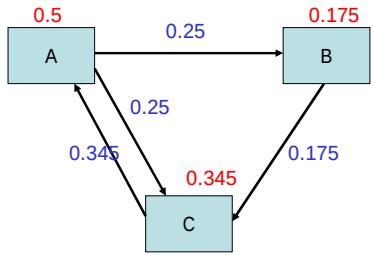
Ad-hoc weights are assigned to each node

Propagating weights



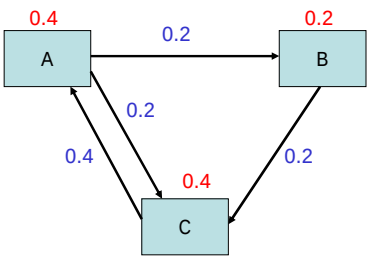
The node weights are re-defined by the incoming edge weights

Propagating weights



We get new node weights

Propagating weights



- We get stable weight assignment
  next-step weights are the same as previous ones
- Component Rank : order of nodes sorted by the weight

Outline

- Motivation and research aim
- SPARS-J
  - Outline
  - System architecture
  - Ranking method
  - Each part
    - Analysis part
    - Retrieval part
    - User Interface
- Experiment
- Conclusion and Future work

Component retrieval part

- Search components from database, rank components
- The process
  - Search components
  - Ranking suited to a user request
  - Aggregate two ranks (CR and KR)

Search components

- Search query
  - Words a user input
  - The kind of an index word, package name
- Components contain given query are searched from Database

Ranking suited to a user request

- Keyword Rank (KR)
  - Components which contain words given by a user are searched
  - Rank components using the value calculated from index word weight
  - Index word weight
    - Many frequency in use of a component
    - A word contained particular components
    - A word represent the component function such as Class name
  - Sort the sum of all given word weight
  - *TF-IDF weighting* using full-text search engine

Calculation of KR value

- Calculate weight W_{ct} with component c word t
  - TF_i: The frequency with which a kind i of word t occurs in component c
  - IDF: the total number of components / the number of components containing word t
  - kw_i: Weight of a kind i

$$w_{ct} = \left(\sum^{allkind} kw_i \bullet TF_i \right) \bullet IDF$$

- KR value is the sum of all word W_{ct}

the kind of a word	weight
Class name	200
Interface name	50
Method name	200
Package name	50
Import	30
Method call	10
Field access	10
Variable type	10
Instance creation	10
Local var access	1
Comment	30
Doc comment	50
Line comment	10
String	1

Aggregate two ranks

- Aggregate two ranks KR and CR
- Aggregation method
  - Borda Count method known a voting system
    - Use for single or multiple-seat elections
    - This form of voting is extremely popular in determining awards
  - SPARS-J
    - Rank components both KR and CR
    - Using KR and CR, the component that be suitable user's request, reusable and sophisticated

Borda Count method

- There are 10 voters and 5 candidates (from A to E)
- Each voter rank candidates
- 1 point for last place, 2 points for second from last place ..., and N points for first place
- 1st=5points, 2nd=4points, ...
  - A：15+3+6+4=28points
  - B：38points
  - C：38points
  - D：22points
  - E：26points

	1st	2nd	3rd	4th	5th
3	A	B	C	D	E
3	E	B	C	D	A
2	C	B	A	E	D
2	C	D	B	A	E

Aggregation

1st	1st	3rd	4th	5th
B	C	A	D	E

Outline

- Motivation and research aim
- SPARS-J
  - Outline
  - System architecture
  - Ranking method
  - Each part
    - Analysis part
    - Retrieval part
    - User Interface
- Experiment
- Conclusion and Future work

User interface

- Receive a user's query and provide the search results through Web browser
  - Microsoft Internet Explore, Mozilla, etc.
- The process
  - Parse query word and the search condition
  - Show rank ordered results
  - Show analyzed information of the component
    - Used by/Using the component
    - Metrics

Analyzed information

A component information are as follows

- Metrics
  - The number of method, variable
  - LOC, cyclomatic
  - Etc. (measurable metrics in the component itself)
- Components used by/using the component
  - Show lists of nodes followed use relation
- Components that are similar to the component
  - Show lists of similar components

Package browsing

- The naming structure for Java packages is hierarchical
  - A user can search lists of components in same package of a component easily

Screenshot (top page)

Screenshot (search results)

Screenshot (source code)

Screenshot (similar components)

Screenshot (using the component)

Screenshot (used by the component)

Screenshot (package browsing)

Outline

- Motivation and research aim
- SPARS-J
  - Outline
  - System architecture
  - Ranking method
  - Each part
    - Analysis part
    - Retrieval part
    - User Interface
- Experiment
- Conclusion and Future work

Experiment(1/2)

- Comparison with Google
  - Register about 130,000 components get from Internet
  - Query words ‘calculator applet’ and ‘chat server client’
    - Calculate relevance ratio of 10 rank higher
    - Relevance: The component is reusable source code
- Google is a web search engine...
  - Add ‘java source’ term to the query words
  - Follow one link from the result web page

Experiment(2/2)

- Example 1:
  - ”calculator applet”
  - SPARS-J
    - 9 hits
    - 7 suited components
- Example 2:
  - ”chat server client”
  - SPARS-J
    - 69 hits
    - 57 suited components
- Using SPARS-J, suited component is high order

$$\text{relevance ratio} = \frac{\text{The number of relevant component}}{\text{rank order}}$$

	Example1				Example2			
	SAPRS-J		Google		SPARS-J		Google	
order	Relevance	Ratio	Relevance	Ratio	Relevance	Ratio	Relevance	ratio
1	○	1	○	1	○	1	×	0
2	○	1	×	0.5	○	1	×	0
3	○	1	○	0.67	○	1	×	0
4	○	1	×	0.5	○	1	×	0
5	○	1	○	0.6	○	1	×	0
6	×	0.83	○	0.67	○	1	×	0
7	○	0.86	×	0.57	○	1	○	0.14
8	×	0.75	○	0.63	○	1	×	0.13
9	○	0.78	×	0.56	○	1	○	0.22
10	-	-	×	0.5	○	1	○	0.3

Conclusion and Future work

- We developed component search engine SPARS-J
  - Using SPARS-J, retrieval of components used well is enabled easily.
- Future work
  - Morphological analysis of Index keyword
  - Collaborative filtering
  - Investigate best ranking method
    - The value of weight
    - Aggregation ranks
  - Evaluation of SPARS-J
    - Usability