ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology
Volume 1, Issue 4, June 2012

A Resolved Retrieval Technique For Software Components

Swathy vodithala[1], P.Niranjan Reddy[2] and M.Preethi[3]

***Abstract*___Retrieval is a major area in software repository because reuse is only effective if it is easier to locate than to write from scratch. In this paper we are describing the classical methods for retrieving the reusable components and also proposed a retrieval mechanism which serves the best. The advantages of signature and behavioural algorithms have been unified in order to retrieve the best components.**

.

***Keywords:* Repository, Retrieval technique, Signature matching, Behavioral matching**

I. Introduction

The software engineering has been widely used area nowadays, since it accepts even the other technical areas into it which strengthens this stream. There are three major areas in software engineering which has to be focused when considering the components for software reuse. These are described as a) classifying the components needed .b) describing the components wanted .c) finding the appropriate components. Finding the component includes a major area of search techniques and retrieval techniques. In this paper we have surveyed different types of retrieval mechanisms. However, finding and reusing appropriate software components is often very challenging, particularly when faced with a large collection of components. A classified collection is not useful if it does not provide the search-and-retrieval mechanism to use it. At different times, different software reuse classification and retrieval techniques have been proposed and implemented. . Retrieval should allow users to formulate high-level queries about component capabilities and takes account of the context in which a query is performed to assist query formulation [10].

II. Related work

A. Keyword Search

Keyword search requires assigning to each object a number of relevant keywords or indices [1].

Ex: The following keywords are associated with objects [5].

OBJECT1 -KW1, KW2, KW3.

OBJECT2 –KW3, KW4, KW5.

If we search the objects based on KW3, then the result would be OBJECT1 & OBJECT2.This involves a lot of irrelevant objects. The other disadvantages with the keyword method is the high cost associated with manual indexing, which requires skilled personnel and ambiguous nature of keywords that can lead to substantial disagreement over the choice of keywords

B. Full-text Retrieval

The high cost of manual indexing has made it attractive to automate the indexing process. The simplest kind of automatic indexing is illustrated by fill-text retrieval systems. Such systems work on the basis of a simple mechanism:
"Store the fill text of all documents in the collection in a computer so that every character of every word in every sentence of every object can be located by the machine. Then, when a person wants information from that stored collection, the computer is instructed to search for all documents containing certain specified words and word combinations, which the user has specified "[1].

C. Hypertext Search

Hypertext represents one of the newest forms of computer-based support for reading documents [1]. Rather than being constrained to the linear order of conventional documents, users are able to move through a hypertext document by following links represented on the screen by buttons**.** The basic building blocks in hypertext are nodes and links. Each node is associated with a unit of information, and nodes can be of different types. The node type depends on various criteria, for example, the class of data stored (plain text, graphics, audio or an executable program), or the domain object it represents (diary entry, account, financial statement). Link*s* define an invoice to a detailed customer profile screen. Links are accessed from the source node and can be traversed to access the destination node. Current hypertext systems provide users with sophisticated user interface tools that enable them to inspect node contents, and to navigate through the network by selecting a path to follow.

D. Enumerated classification

Enumerated classification uses a set of mutually exclusive classes, which are all within a hierarchy of a single dimension [8]. A prime illustration of this is the Dewey Decimal system used to classify books in a library. Each subject area, e.g., Biology, Chemistry etc, has its own classifying code. As a sub code of this is a specialist subject area within the main subject. These codes can again be sub coded by author. This classification method has advantages and disadvantages pivoted around the concepts of a unique classification for each item. The classification scheme will allow a user to find more than one item that is classified within the same section/subsection assuming that if more than one exists. For example, there may be more than one book concerning a given subject, each written by a different author. This type of classification schemes is one, and will not allow flexible classification of components into more than one place [9]. As such, enumerated classification by itself does not provide a good classification scheme for reusable software components.

E. Attribute value

The attribute value classification scheme uses a set of attributes to classify a component [8]. For example, a book has many attributes such as the author, the publisher, its ISBN number and it's classification code in the Dewey Decimal system. These are only example of the possible attributes. Depending upon who wants information about a book, the attributes could be concerned with the number of pages, the size of the paper used, the type of print face, the publishing date, etc. Clearly, the attributes relating to a book can be:

· Multidimensional. The book can be classified in different places using different attributes

· Bulky. All possible variations of attributes could run into many tens, which may not be known at the time of classification [9].

F. Faceted

Faceted classification schemes are attracting the most attention within the software reuse community [8, 9]. Like the attribute classification method, various facets classify components, however, there are usually a lot fewer facets than there are potential attributes (at most, 7).proposed a faceted scheme that uses six facets.

· The functional facets are: Function, Objects and Medium

· The environmental facets are: System type, Functional area, setting.

G. Signature matching.

Signature matching compares names, parameters and return types of component methods to the user query.

Consider the signatures presented in Figures 1 and 2 for a stack of integers and a queue of integers, respectively [7].

```
Create: => Stack
Push: Stack x Integer =>Stack
Pop: Stack => Stack
Top: Stack => Integer
Empty: Stack => Boolean
```

Figure 1: Signature of a Stack

Create*:* =>Queue
Enqueue: Queue x Integer:=>Queue
Dequeue: Queue =>Queue
Front: Queue => Integer
Empty: Queue => Boolean

Figure 2: Signature of a Queue

These signatures are isomorphic up to renaming, and thus exemplify what we have come to refer to as the vocabulary problem. Software reusers implicitly associate distinct semantics with particular names, for example, pop and enqueue. Thus, by the choice of names, a component developer can mislead reusers as to the semantics of components, or provide no means of discriminating between components. Figure 3, for example, appears to be equally applicable as a signature for both stack and queue, primarily due to the neutral nature of the names used.

Create: Sequence
Insert: Sequence x Integer =>Sequence
Remove: Sequence =>Sequence
Current: Sequence => Integer
Empty: Sequence => Boolean

Figure 3: Signature of a sequence.

H. Behavioural matching.

Behavioural retrieval works by exploiting the executability of software components. Programs are executed using components, and the responses of components are recorded. Retrieval is achieved by selecting those components whose responses (with respect to the program) are closest to a pre-determined set of desired responses. This idea was originally called ``behavioural matching". A component is represented as a relation between programs and responses. This is because in general, a program execution can yield several responses (due to non-determinism) and a response may be evoked by more than one program. Formally, a component *C* can be declared as [4]:

C: program $\leftrightarrow$ Response.

A program p belongs to program is modeled as a sequence of calls on the component's interface. A response is a sequence of values in correspondence with a program. In effect, each program determines a context in which the behaviour of a component is exhibited. The behaviour of a component *C* is derived from the set of response sequences by removing those responses which are proper extensions of other responses. Thus, the behaviour of a component *C* is the set of guaranteed responses to a program p.

P: Response $\rightarrow$ Behavior

In general, Behavioral matching executes each component with input data in order to retrieve the components that present the expected behavior.

III. Proposed work

The disadvantage of the behavioral algorithm is as follows: when sent a program, the component will respond by executing each call in the program and producing a corresponding output sequence called response. However, here we have a set of predefined responses. If the program's response is matched with the predefined response which is already defined, then the component will be retrieved. But at times a call is rejected where execution stops and there are various reasons for this. The method name called may not be the name of an operation in the component's interface, or the supplied input may not satisfy the operation's preconditions [4].

The disadvantage of the signature matching is as follows: signature match is entirely based on function types, but not on the behaviour. If we consider the functions strcpy ( ) and strcat ( ), they have same signature but opposite in nature.
(Most of the functions have same signature but different behaviours) [2].

With the description above, we can say that alone signature matching and behavioural matching would not yield good results.

The proposed algorithm is the "combination of both the signature and behavioural matching".If we consider the behaviours along with the signatures the disadvantage of signature matching can be minimized. Similarly, if we consider the signatures along with the behaviours the disadvantage of behavioural matching can be minimized that is all the names of the methods are properly filtered before having behavioural match.

EXAMPLE: COMPONENTS TO FIND ADDITION OF TWO NUMBERS

Firstly, we search for a component with signature

int name (int ,int)

i.e... a function which takes two integer arguments and returns a integer value.

This may give a list of functions as

```
int sum (int ,int)
int add ( int ,int)
int sub (int, int)
int mul (int, int)
int div (int ,int)
int avg (int, int)
```

The resulted list has functions with various behaviors. Then we apply the behavioural matching to find the addition of two numbers.
Here the response of a function is checked against the predefined responses and the opposite behaviours are removed. The resulted list would be as

```
int sum (int,int)
int add (int,int)
int avg (int,int)
```

The Expected behaviour is seen in all the above functions .Again here, the sum( ),add( ) comes under the "EXACT MATCH" and avg( ) comes under the "RELAXED MATCH".

IV. Conclusion and future scope

In this paper we explained all the basic retrieval techniques along with their strengths and weakness. We also proposed a matching technique which combines both the behavioural and signature matching .This serves the best when compared with the individual algorithms. The future scope involves the unification of basic techniques with the other technological disciplines which are relevant to optimize the search as well to retrieve the software components.

References

[1] "Supporting search for reusable software objects" Tomas isakowitz and Robert j. Kauffman.Center for Digital Economy Research Stern School of Business Working Paper IS-93-47

[2] "Specification matching of software Components" Amy moormann zaremski and Jeannette m. wing

[3] "Signature matching: A key to reuse"Amy moormann zaremski and Jeannette m. wing.

[4] "Examining behavioral matching"Steven Atkinson.Software Verification Research Centre.Department of Computer Science University of Queensland

[5] "Building Software Reuse Library with Efficient Keyword based Search Mechanism" Rajender Nath and Harish Kumar. technia – international journal of computing science and Communication Technologies, VOL. 2, NO. 1, July 2009. (ISSN 0974-3375)

[6] "Facets of Software Component Repository" Vaneet kaur and shivani goel. / International Journal on Computer Science and Engineering (IJCSE)

[7] "Component Classification and Retrieval Using Data Mining Techniques" Proceedings of National Conference on Challenges & Opportunities in Information Technology (COIT-2007) RIMT-IET, Mandi Gobindgarh. March 23, 2007. Achala Sharma, Daman Deep Kaur

[8] "An Integrated Classification Scheme for Efficient Retrieval of Components" Dr C.v.guru Rao and P.Niranjan. Journal of Computer Science 4 (10): 821-825, 2008 ISSN 1549-3636 © 2008 Science Publications

[9] "A mock-up tool for software component reusesRepository" P.Niranan and Dr.c.v.GuruRao.

[10] "Supporting reuse by task-relevant and personalized information" Yunwen Ye and Gerhard Fischer .ijcse'02.

First Author SWATHY VODITHALA received her B.Tech degree in computer science and engineering in 2005 from KITS, huzurabad (Jawaharlal Nehru Technological University). And M.Tech degree in software engineering from KITS, Warangal (kakatiya university) in 2011.she worked with computer science department for 2 years in KITS, huzurabad as an Assistant Professor. At present, she is working as an Assistant Professor in computer science department at KITS, Warangal.