Volume 2, Issue 2, February 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Smart Environment for Component Reuse

Suresh Chand Gupta
CSE Department
P.I.E.T Samalkha, Panipat

Prof Ashok Kumar
Department of Computer Science & Applications
K.U. Kuurukshetra

Abstract— ***The keyword based approach largely used in descriptive methods for component retrieval procedure is not well accurate. In keyword-based procedure, a candidate asset is selected whenever the keywords that form its depiction match all or most of the keywords of the query. Keyword search regain large number of listed components, it is the responsibility of the user to manually find out the component that is most appropriate to be adapted to reuse that components. To find out the best-fit reusable components is a tedious task for the user. Although keyword based procedure is widely used for descriptive procedure, it yields results that are far too formless. Here we propose a smart environment for the retrieval approach of components to make them reusable. The class diagram contains effective information about the structural description and contents of a class, i.e. class name, attributes, behaviour, relationships, generalization etc. If we search the repository by using this information, the search result would be better and thus providing higher accuracy, as compared to keyword based search procedure.***

Keywords— ***MDL, UML Models, Search Environment***

I. INTRODUCTION

Software reuse is the process of implementing or updating software systems using existing software assets. Software assets, or components, include all software products, from requirements and proposals, to specifications and designs, to user manuals and test suite. Anything that is produced from a software development effort can potentially be reused. Even libraries of Unified Modeling Language diagrams and source codes do exist, one of the challenges that still remain is to locate suitable designs and source codes, and get used to them to meet the requirements of the software designer for the development of new software. A traditional approach to component retrieval is keyword-based; which produced result in the retrieval of less relevant components. A more result producing approach is retrieval based on MDL file format, where the contents of MDL file of the diagrams are matched to retrieve the components. The UML models that used for modeling are stored as MDL file format. These MDL file formats are generally very information rich and contains a large number of important and valuable information about the components. This available information can be structural as well as behavioral in nature. The class diagram MDL file format have valuable information about the structural description and contents of a class, i.e. class name, attributes, behavior, relationships, generalization, cardinality etc. These attributes can be used for specification matching with the contents of the repository. The Use case diagram file contains valuable information about the requirements specification of software. These include use cases and actors. If we search the repository on the basis of attributes of MDL file descriptions, the search result would be better and thus giving much higher precision, as compared to keyword based searching procedure. Hence the role of user to find the best suitable component from the search results would be much easier. In this paper we describe a tool named as a smart environment for retrieving the related components, which assists the software designers in the retrieval of the designs as well as source codes.

II. LITERATURE SURVEY

A. *Existing Retrieval Techniques*

In [27], a hybrid technique based on natural language description and formal specifications using K-nn technique has been discussed. It discusses the reuse and benefits of reuse using formal methods. It also demonstrates the benefits of using natural language along with formal methods, which is supported by the tool demonstrated. It discusses a Reuse WELL System which exploits the benefits of both formal methods and natural language in the retrieval of software components. Reuse WELL tool in this paper has just been implemented for subset of the Z notation. It can further be extended and implemented for every Z markup. Moreover Reuse WELL tool can be implemented for other libraries also. In [25, 26, 28], ant colony algorithms based technique that generates rules to store and then identify the component for possible reuse is discussed and demonstrated. This technique helps users in organization and storage of components and later can help in identifying most appropriate components. In the first stage while searching it makes use of keywords, their synonyms and their interrelationships. Then it makes use of ant colony optimization; initial pheromone of one is assigned to all domain representation terms of components. By updating pheromone for participating terms iteratively and by calculating the quality of each rule generated, it leads to quality rules to represent and retrieve the reusable components. In [30], design and implementation of a storage and retrieval structure for software a component that is based on formal specifications and on the refinement ordering between specifications is discussed. It discusses retrieval algorithm for exact and approximate retrieval, along with their design and implementation.

B. Features for Characterizing Software-Retrieval Method

The features that are being used for characterizing Software-Retrieval methods are as follows [19]:

Nature of Assets – This includes the nature of asset that is stored in the library. Assets include source code, executable code, requirements specification, design description, test data, documentation, and proof.

Scope of Library – This includes the scope in which the library is to be used. That is whether the library is to be used within a project, across a program, across a product line, across multiple product lines, worldwide.

Query representation – This includes the form of the query submitted to the library. Queries can be submitted in various forms as functional specification, signature specification, keyword list, design pattern, behavioral sample.

Asset representation – This includes how the assets are represented in the library. It dictates what form user queries should take. Assets can be represented as functional specification, signature specification, source code, executable code, requirements specification, documentation, and set of keywords.

Storage Structure – This includes how the assets are stored in library. Storage structures can be flat structure, hypertext links, refinement ordering, ordering by generosity.

Navigation Scheme – This includes how the assets are visited or navigated. Various navigation schemes include Exhaustive linear scan, navigation hypertexts links, and navigating refinement relations.

Retrieval Goal – This includes, finding out assets that are correct with respect to a given query. Various retrieval goals include correctness, functional proximity, and structural proximity.

Relevance Criterion – This includes under what conditions a library asset is considered to be relevant for the submitted query with respect to the predefined retrieval goal. Relevance Criterion includes correctness, signature matching, minimizing functional distance, minimizing structural distance.

Matching Condition – This includes the condition that is chosen to check between the submitted query and a candidate library asset. Various factors for matching condition include correctness formula, signature identity, signature refinement, equality and subsumption of keywords, natural language analysis, and pattern recognition.

C. Gaps in Existing Techniques

The major drawbacks of the traditional descriptive classification schemes for software component retrieval are:

- Ambiguity problem in keyword based search procedure; when different words mean different things to different people..
- Both Precision and Recall are not high.
- These techniques are based on a proscribed vocabulary that must be constructed manually for each and every application domain.
- Both classification and retrieval require important human effort because users must select appropriate terms for each facet in the classification scheme from usually list of terms in the controlled vocabulary.

This paper proposes an approach combining main advantages of descriptive classification methods for component retrieval in order to improve retrieval effectiveness and provide a friendlier user interface through the use of queries in MDL format.

III. EXPERIMENT

. The primary goal of this paper is to identify a retrieval approach by using the MDL format. The UML models that are used for modeling are stored as MDL file format. These MDL file formats are practically very information rich and contains lot of meaningful information about the components. This available information can be structural as well as behavioral.

To successfully combine two paradigms software reuse and UML, for Component Retrieval, an automated tool must be designed, named as Component Retrieval Search Engine. The Purpose of Component Retrieval Search Engine is to retrieve best-fit or most reusable Component from the Repository as intended by the (re) user. Moreover the search results to be displayed in descending order of percentage match with the input query. Hence the role of user to find the best-fit component from the search results would be much easier.

A. Proposed System

The system developed is called as a smart environment for component retrieval, which stores MDL files as well as source codes in a Repository. The tool has the retrieving mechanism for retrieving MDL files as well as source codes from the Repository. The System Working is shown in Figure 1.The main components of smart environment for retrieving the components are the input file Reader, The Repository, and retrieving mechanism. First the Repository Manager stores the MDL Files as well as source codes in the Repository. Now when a query is inserted for retrieval of a required component, MDL file Reader reads the MDL file provided as input to the system, and then stores the contents of the MDL file to be used in the search operation in a temporary storage. Based on the contents of MDL File the Search Engine is there to Search for the desired results from the Repository and shows the results back to the developer.

(1) MDL File Reader: Diagrams are the pictorial representation of the system. Here we apply some principles to reveal the details from the diagrams. The diagrams can be easily modelled in some software like Rational Rose or Visualiser. After developing the designs of software in UML, store the file. By specifying and storing all the diagrams in repository, we can read and extract the details. Taking the example of class diagram, in a class diagram, there are different classes, operators, attributes, relationships and documentation or comments provided for explanation and behaviour. There is relationship between the classes like inheritance, association or any other dependency. Then they are related with the Cardinality. These files must be read by applying some algorithm, to extract all the details. But one must keep trace of classes, operations, attributes and their relations. The attributes of one class should not mix with other class. To implement such algorithm, MDL file Reader is used to extract the necessary information. The MDL file can be read by treating it as a simple text file. Start reading the file and whenever a match is found for object class, object attributes and object operation store the string following next in the temporary storage. Likewise all other attributes can be stored by reading the file. The contents are stored in a temporary storage and used for match when the search is to be performed.

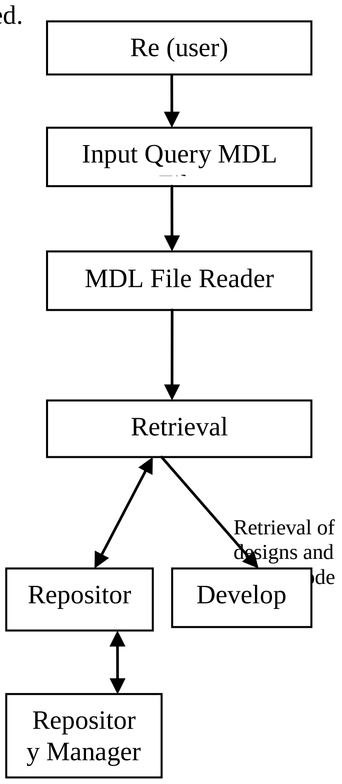

Figure 1. Reusability: component retrieval based on MDL format

2) The Repository Design:
The purpose of Repository is to store designs as well as source codes. Designs are stored in MDL format and source codes consist of programs in Java. The architecture of Repository is prepared for storing both designs as well as source codes.

3) Retrieval Mechanism:
To extract diagrams or components from repository, to facilitate software reuse a search engine is required to fetch data as per requirements of the user. As there are many diagrams in UML the search for each diagram is different and the technique required is different. To search from the repository, different search techniques are utilized to extract relevant data..

MDL File – Class Diagram Search:
Search Technique – In this case, search is made on the MDL file Class Diagram. Now the search is not defined on the keywords or comments of a class. In this case the search is performed on the structure of the whole software i.e. class name, attributes name and operations name. The class diagram of new software is modeled in Rational Rose and the file is saved in MDL file format. Now for the search, this MDL file would be the input to the query. The software would read the class structure modeled and depending upon these structures will search for structure in repository and would result back with preexisting MDL files as well as source codes from the repository. Moreover some weights are assigned to different contents of a class as class name, class attributes and class operations. These weights are computed and stored along with the component path in a temporary storage .The search results are displayed in the decreasing order of their weights i.e. percentage match of the component. The component with highest weight is displayed first and then next and so on. Thus the best- fit component is that which is having highest percentage match. Hence user can easily find the best-fit component from the search results.

MDL File – Use Case Diagram Search:
Search Technique – In this case, search is made on the MDL files Use Case Diagram. Now the search is not defined on the keywords or comments of a Use Case Diagram. In this case the search is performed on the functional requirements of the whole software i.e. use cases and actors. The use case diagram of new software is modelled in Rational Rose and the file is saved in MDL file format. Now for the search, this MDL file would be the input to the query. The software would read the requirements specification modelled and depending upon these requirements specification will search for requirement specifications in repository and would result back with pre-existing MDL files from the repository. Moreover some weights are assigned to different contents of a use case diagram as actors, which is a special kind of class and use case. These weights are computed and stored along with the component path in a temporary storage .The search results are displayed in the decreasing order of their weights i.e.

percentage match of the component. The component with highest weight is displayed first and then next and so on. Thus the best- fit component is which is having highest percentage match. Hence user can easily find the best-fit component from the search results.

Example Cases:
As per the data in the database, required is a search of a data from the repository. The search technique and the search query depend upon the data to be extracted and the data present in the repository. The functionality of the retrieval mechanism is to fetch the data from the repository to the user in the desired output. Here, we are dealing with designs and source codes. Designs itself cannot be reused as it is, as there is no physical component that can be extracted, although source codes can be reused as it is. The output form would be the information about the components or the details provided by the repository manager at the time of insertion of the component in the repository. The form of output of the search query is the file itself. Next level of data extraction is to be provided by the developer as how to use the component. Also the output can provide the information about the components, that component could be found from the Repository and then reuse principles are applied. Here the main motive at the design and coding level is to know about the similar kinds of components as per the search query, present in the Repository.

IV. CONCLUSIONS

Collectively software reusing with UML is an emerging trend in the process of software development. Combining these techniques can help the software development process by finding existing components at the design time only, due to which the total effort of software development can be decreased.

- UML and software reuse working together can be used to retrieve required components that can be best fitted for the development of new software modules.
- The search can be made on class diagrams to search according to structural description of the software.
- The search can be made on use case diagrams to search according to requirements specification of the software.

REFERENCES

[1] Arun Sharma, Rajesh Kumar and P .S. Grover, “A Critical Survey of reusability aspects for component-based systems”, Proceedings of World Academy of Science, Engineering & Technology, Vol. 21, Jan 2007.

[2] Boehm, B “Managing software productivity and reuse”, IEEE Computer 16(9), 111- 113, 1999.

[3] Burton, B. A., Aragon ,R. W. , Bailey ,S .A., Koehler ,K .D., and Mayer ,L .A, “The Reusable software library”, IEEE Software 4, 4, 25-33, 1987

[4] Clifton, C. and W. S. Li, “Classifying software components using design Characteristics”, In proceedings of the 10th Knowledge-Based Software Engineering Conference, KBSE’95, IEEE Computer Society press, Los Alamitos, CA PP 139-146, 1995

[5] Daniel Lucredio, Antonio Francisico do Prado, Eduardo Santana de Almeida, “A Survey on Software Components Search and Retrieval”, euromicro, pp.152-159, 30th EUROMICRO Conference (EUROMICRO’04), 2004

[6] Frakes,W.B and Pole,T, “ An Empirical study of representation methods for reusable Software components”, IEEE Trans. Soft Engg 20, 8,617-630, 1994

[7] Fichman, R.G & Kemerer, C.E, “Object Technology and Reuse: Lessons from early Adopters”, IEEE Software 14(10), 47-59, 1997

[8] Grady Booch, James Rumbaugh, Ivar Jacobson, The Unified Modeling Language, User Guide (Pearson Education, 2005)

[9] Grady Booch, James Rumbaugh, Ivar Jacobson, The Unified Modeling Language, Reference Manual (Pearson Education, 2005)

[10] Hafedh Mili, Fatma Mili and Ali Mili, “Reusing Software: Issues and research Directions,” IEEE Transactions on Software Engineering, Vol. 21, No 6, 1995

[11] Henninger,S “An Evolutionary Approach to constructing effective software reuse Repositories”, ACM Transactions on software engineering and methodology 6(2), 111-140, 1997

[12] Isakowitz,T and R,J Kauffman , “Supporting Search for Reusable Software Objects”, IEEE Transactions on Software Engineering 22, 6, 407-423, 1996

[13] Jiang Guo, Lqui, “A Survey of Software Reuse Repostories”, ecbs, p-92, 7th IEEE International Conference and Workshop on the Engineering of Computer Based Systems, 2000

[14] Jilani L L, R.Mili, M Frappier, J.Desharnais and A.Mili, “Retrieving Software Components that minimize adaptation effort”, In Proceedings of the 12th IEEE International Automated Software Engineering Conference, ASE’97, IEEE Computer Society Press, Los Alamitos, CA pp 255-262, 1997a

[15] Jilani,L.L , R Mili and A Mili, “ Approximate Retrieval: An Academic Exercise or a Practical Concern”, In Proceedings of the 8th Annual workshop on software Reuse (WISR-8), 1997b

[16] Michel Ezran, Maurizio Morisio, Colin Tully, "A Survey of European Reuse Experiences: Initial Results," *euromicro*, p. 875-881, 24 th. EUROMICRO Conference Volume 2 (EUROMICRO’98), 1998

[17] Mili.A, Yacoub.S, Addy.E, Hafedh.M, “Toward an Engineering Discipline of Software Reuse”, IEEE software 16(5), 22-31, 1999

[18] Michail,A. & Notkin,D., “Assessing Software Libraries by Browsing similar classes, functions and relationships” , In Proceedings of 21st International Conference on Software Engineering (ICSE’99), ACM Press, Los Angeles, CA, pp. 463-472, 1999

[19] Mili A, Mili R and Mittermeir R.T, “A Survey of Software Reuse Libraries,” Annals of Software Engineering Vol. 5, 349-414, 1998

[20] Mili R, Mili A and Mittermeir R.T, “Storing and Retrieving Software Components: A Refinement Based System”, In Proceedings of 16th International Conference on Software Engineering, IEEE, pp.91-100, May 1994

[21] Mili and Edward Addy, Reuse Based Software Engineering (A Wiley-Interscience Publication, John Wiley and Sons, Inc.2002)

[22] Peter Eisinga and Jos Trienckens, Software Components for the Industry, From testing of applications to evaluation of components.

[23] Prieto-Diaz, “Implementing Faceted Classification for Software Reuse”, Communication of the ACM 34, 5, 88-97, 1991

[24] Prieto-Diaz, R. and Freeman.,P, “ Classifying Software for Reusability” ,IEEE Software.4, 1, 6-16, 1987

[25] Rajesh K Bhatia, Navneet Kaur, “Information Retrieval from a composite based Repository using Genetics Algorithms” ‘IICAI 2005, page 667-675

[26] Rajesh K Bhatia, Mayank Dave, R.C Joshi, “Retrieval of most relevant reusable Component using genetic algorithms”, Software Engineering Research and Practice 2006, 151-155

[27] Rajesh K Bhatia, Mayank Dave, R.C Joshi, “A Hybrid Technique for Searching a Reusable Component from Software Libraries”, DESIDOC Bulletin of Information Technology, Vol.27, No.5, September 2007, pp. 27-34

[28] Rajesh K Bhatia, Mayank Dave, R.C Joshi, “Ant Colony Based Rule Generation for Reusable Software Component Retrieval”, Proceedings of the 1st Conference on India Software Engineering Conference, pp 129-130, Feb 19-22, 2008, Hyderabad, India

[29] Rajiv D. Banker, Robert J Kauffman and Dani Zweig, "Repository Evaluation of Software reuse", IEEE Transactions on Software Engineering, Vol. 19, No 4, April 1993

[30] Rym Mili, Ali Mili and R.T.Mittermeir, "Storing and Retrieving Software Components: A Refinement Based System", IEEE Transactions on Software Engineering, Vol.23, No 7, July 1997

[31] S. Araban, "A Two level Matching Mechanism for Object-Oriented Class libraries", Ada-Europe 1998: Uppsala, Sweden, pp 188-200, no.1, Jan1993