International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 4, April - 2013

A Survey For Effective Search And Retrieval Of Components From Software Repositories

Amandeep Bakshi
Computer Science and Engineering Department
Thapar University, India

Seema Bawa
Computer Science and Engineering Department
Thapar University,India

Abstract

Software Repositories for reusing the components have been an active research area for more than a decade, as these software repositories must be designed to meet the evolving and dynamic needs of software reuse development. Software repositories are required to facilitate storing and maintaining reusable software components efficiently. This paper presents a survey about the main research on effective search and retrieval of components and on various software repositories. In the literature, there are several works that explore software reuse repositories. This survey will be the base for an efficiently searchable, user friendly and effective retrieval of components from well-organized repositories

IJERT

1. Introduction

As libraries of reusable software components continue to grow, the issue of retrieving components from software components repositories has captured the attention of the software reuse community [1]. The storage and retrieval of reusable assets in a software repository has been the subject of active research in the past. Reusable assets includes program specifications, source code, object codes, documentations and the UML specifications that are produced in a project and can be later reused.

However, finding a components and reusing the appropriate software components is often very challenging, particularly when faced with a large collection of components and little documentation is there about how they can be reused. This is a particular issue for end users of component-based systems who want to tailor and extend their environment, but have limited understanding of component functionality and implementation [5]. Many software components repositories have been developed, often extending the approaches used for software repositories.

Hence, it is becoming difficult to retrieve relevant components. This motivates researchers to look for retrieval techniques that give efficient guidance for locating and identifying appropriate software components that satisfies user queries [11]. Various retrieval techniques are enumerated classification, facets, frame-based classification; free-text indexing and relational databases have been employed to address the problem of finding relevant components [1]. But issues involving how effective repositories are built, populated and evolved to meet the changing needs of software developers have received considerably a less attention. Thus, this paper describes a research on effective search and retrieval of software components from software repositories.

This paper aims to provide an overview of the state of research in the area of software reuse repositories with the intention of providing future research area by looking at the research gaps and untouched areas. This paper is organized as follows. Section 2 describes the existing repositories. Section 3 describes the various search and retrieval methods and techniques. Section 4 describes the various approaches used for effective search and retrieval of components and Section 5 concludes the work and suggests future research directions.

This review aims at summarizing the current state of the art in software reuse repositories research by proposing answers to the following questions:

(i)What are the requirements of a software reuse repository?

(ii) How the software repository must be designed?

(iii)What are the various approaches followed for constructing effective repository?

(iv)What are the main challenges faced while searching the components?

(v) What are the main challenges faced while retrieving the components?

2. Existing Repositories for the Effective Retrieval of components

2.1 The CodeFinder-PEEL Repository

Scott Henninger outlines an Evolutionary approach [1] in 1996 for constructing effective software libraries which is demonstrated through CodeFinder-PEEL repository (Parse and Extract Emacs Lisp) which extracts components from text files and indexes these components through a combination of automatic extraction. The CodeFinder system is a prototype retrieval tool that uses a combination of retrieval techniques to help users for finding reusable software components. This repository was designed to investigate the cost/benefit tradeoffs. Components are extracted through PEEL and translate these components into a CodeFinder representation. Once, components are translated, seeding a repository is necessary for creating the component representations by indexing them with phrases and key terms. PEEL is a reengineering tool that translates Emacs Lisp files into the individuals reusable components representing into the frame based language named kandor which is used by CodeFinder for indexing the components and create a frame based hierarchy of retrieval concepts. PEEL extracts source code definitions of functions, constants, variables and macros from a source code file. Then this information is extracted and translated into kandor objects. Kandor representation can be viewed as a set of attributes which contain information about the component. This approach basically shows how repositories can be constructed with minimal up-front effort.

2.2 CodeBroker

Yunwen Ye discusses an active and adaptive reuse repository system in 2001 [4] to address the various challenges and demonstrates the feasibility of implementation through a prototype 'Code Broker' system to assist the java developers in reusing various classes and methods. The Code Broker automatically locates the relevant software reusable components and delivers these right components into the development environment. The architecture of Code Broker consist of different software agents i.e. listener, fetcher and presenter. Listener and Presenter are the interface agents which connect the software developers to the backend information agent i.e. fetcher. Listener captures the software developers needs for reusing the components and formulates these queries autonomously. Fetcher retrieves these relevant reusable components based on the concept similarity and constraints. Fetcher is a backend information agent which retrieves the components from the repository. Presenter uses a user profile for each software developers to adapt the components retrieved by the Fetcher to the developer's knowledge level. CodeBroker is currently designed to promote reuse in the phase of coding.

2.3 Aspect-based component repository

John Grundy describes a software component repository [5] in 2001 which uses the concept of 'aspects' to index and query the various software reusable components used for automatically generate a high level indexing based on the systemic characteristics. A key aim of this approach is to make it easier for developers and for the end users to formulate the high level queries for components and have access to high level information about the retrieved components. Thus every component is added into the repository with its aspect details so that component is easily indexed and queried by using the aspect details for efficiently retrieval.

This technique is primarily a facet-based approach where users query for components based on the particular systemic aspects i.e. attributes of components.

2.4 CRECOR Component Repository

Jihyun Lee, Jinsam Kim and Gyu-Sang Shin describes the component repository [6] in 2003 which supports for facilitating EJB as a reusable software component in component based software development CRECOR(Component Repository for facilitating EJB Component Reuse) is developed to store components and support reuse activities which also provides the capability to reuse the pre built EJB software components with GUI. Various Reuse facilitating activities are component analysis, component adaption, component deployment, and component test and assembly are described in this approach so that components can be efficiently reused.

3. Search and Retrieval Methods, Schemes and Techniques

Many software classification and retrieval techniques are available for the effective retrieval of the components from software repository.

3.1 Classification Schemes

Daniel Lucredio presents a survey about the main research on component search and retrieval [7] in 2004 and discusses how efficiently components can be

IJERT

search in order to support future component markets. They specially focus on the classification schemes used to store the software components. The author proposes the utilization of a facet based scheme to classify the software components and the limited numbers of characteristics i.e. facets are retrieved with a set of possible keywords are associated with each facet but problem arises when 2 different people may choose different keywords to describe the components. So, to overcome the problem, maarek [8] purposes the use of automatic indexing and extraction of terms or phrases that describe a best component.

3.2 Component Storage and Retrieval method

A.Mili , R.Mili and R.T.Mittermeir presents a survey on methods of storage and retrieval of software assets in software component libraries in 1998 [3]. They classify storage and retrieval methods on the basis of which software libraries can be characterized. They define the assessment criteria which includes technical, managerial and human factors (precision, recall and coverage ratio factors) for the effective retrieval of components. On the basis of these assessment criteria, they classify the software libraries which include informational retrieval method, descriptive method, operational semantic methods, devotional semantic method, topological method and structure method.

3.3 SOM and GSOM Techniques

Ronaldo C.Veras and Silvio R.L.Meira both discussed and compared the two clustering techniques [9] in 2007 namely Self-Organizing maps (SOM) and Growing Hierarquical SOM (GHSOM) for clustering a repository of classes from a java API for building mobile system to provide a more visualization of the software components and to refine more search by grouping together similar components.

3.4 Structural and behavioral Techniques

Hanen Ben Khalifa and Qualid Khayati purpose structural and behavioural techniques [11] in 2008 for making the more efficient retrieval process by combining formal and semi formal specification for the heterogeneity of the software repositories. They purpose another classification for the retrieval of components that classifies the components into three different dimensional spaces. The first axis describes the interaction by using query or browsing between the user and the retrieval system. The second axis focus on the description model for the software components. The last axis distinguishes between the structural and behavioral method. The behavioural based retrieval approaches are based on the notion of exploiting the executability of software components to classify them. Testing the components with different arguments calling their functions yields dynamic responses, which are collected. This collection is called the component behaviour. An ordering on behaviours is then used to classify components and to search through the library of components. The programs used to produce the components behaviour tries to call a subset or all the functions of the component and recover the results. If the program calls functions that do not exist in the component, the components will ignore the call. The person calling some specific function will enter a specific query and this query will be plugged to all components of the library to test the components behaviour. The components that respond to the searched behavior will be selected and presented to the user [17, 18].

3.5 Hypertext Technique

A hypertext consists of a set of interconnected units of textual information. Such a unit of information is called a node. The connections between the nodes are called links. The starting point of a link is called the anchor or the parent node. The ending point is called the destination or the child node. Usually links are one directional but it is also possible to define two dimensional links. Nodes can be anchors and destinations of more than one link. By means of this set of nodes and links, a structure is imposed on the collection of data. A browsing feature which has been classified as a retrieval technique earlier permits the user to wander through the data, thus providing a natural way of accessing the collection of data.

3.6 Browsing Technique

Browsing is an example of a retrieval technique [16] that requires a well structured document collection. Documents must be represented in the system as a network of inter connected nodes and a hypertext is a good way of defining such a structure. With the aid of the system, the user can now browse through this network to find the information he is looking for. The user does not have to formulate a query at first, to represent his information need and there is no such thing as a formal matching process. This can be a major advantage, especially when the user does not exactly know what he is looking for. In fact, the actual matching takes place in the uses mind while wandering through the network of nodes and links, he decides whether or not he finds the current node interesting. If so, he might want to retrieve the document belonging to the node and we could speak about a match. There are no formal rules for this kind of matching. The user may be changing his mind and decide otherwise or maybe he never really made up his mind about what he is

IJERT

looking for and is not able at all to find any matches [16].

4. Existing Approaches for Effective Search and Retrieval of Components.

4.1 Four Layered Architecture Approach

Wang Haitao and Chen Xing purposes a component library model based on the four-layered architecture in 2011 [12] for organizing and managing the software component library more efficiently so that user can easily retrieve, understand and reuse the component. The four layered architecture include framework layer, component layer, leaf class layer and leaf function layer. This architecture is particular suitably for the flexibility and reusability of the component library system.

4.2 Knowledge Based Approach

The text automatic classification method is based on the content analysis automatically to allocate the text into pre-determined catalogue. The methods of text automatic classification mainly use information retrieval techniques. Traditional information retrieval mainly retrieves relevant documents by using keyword-based or statistic-based techniques [15]. Knowledge-based software retrieval systems make some kind of lexical, syntactic and semantics analysis of natural language specifications of software components without pretending to completely understand the document. They are based on a knowledge base which stores semantics information about the application domain and about natural language itself. These systems are usually more powerful than traditional keyword retrieval systems. However, they usually require enormous human resources: Knowledge bases are created for each application domain and are usually populated manually.

4.3 Automatic Indexing Based Approach

Automatic indexing is the process of analyzing an item to extract the information to be permanently kept in an index. The system extracts lexical, syntactic and semantic information from the natural language description. These approaches automatically extract words or phrases (usually pairs of terms) from documents and from queries in natural language to build their internal representation. Automatic indexing is required for the effective retrieval of information. Indexing is essential to information retrieval because it provides entry points to a collection, without the user having to examine the whole collection. It tells the user where the information can be physically found and allows them to search a collection using keywords or phrases. Indexing has existed for as long as humans have been keeping written records. There are two ways that information can be indexed; manually or automatically. In manual indexing a human indexer compiles the index, while automatic is when the task is done by computer. Systems that provide automatic indexing of software components can be classified in two basic groups: systems that work only at the lexical level and systems that include Syntactic and Semantic analysis of software descriptions [19].

4.4 Automatic Tags Extraction (ATE) Based Approach

Lei Zhang, LichaonChen, Lihu Pan, Yingjun Zhang proposed the novel approach Automatic Tags Extraction (ATE) algorithm in 2012 [14] designed for the component retrieval for improving the performance and effectiveness of large scale repositories. In this approach, component tags are extracted automatically and indexed by ATE algorithm then they use the vector space mode (VSM) similarity algorithm to match the component tags.

5. Conclusion and Future Work

This review paper explores the different repositories, schemes, methods and techniques available for effective search and retrieval of the components from a software repository. The result of the study has been analyzed and classified into several categories. The paper has shown the areas of research that have been done by answering the questions that were defined initially.

More work is needed to improve the similarity matching algorithm to further increase the precision and recall ratio. There is also need to implement semantic based search for the precise search and for the effective and exact retrieval of the components from the software repositories.

6. References

[1] Scott Henninger, "Supporting the Construction and Evolution of Component Repsitories", IEEE, 1996, pp-280-286.

[2] Jiang Guo and Luqi, " A Survey of Software Reuse Repositories", IEEE, 1999.

[3] A.Mili, R.Mili and R.T.mittermeir, "A Survey of Software Reuse Libraries", Annals of Software Engineering 5, 1998, pp-349-414.

IJERT

[4] Yumen Ye , " An Active and Adaptive Reuse Repository System", 34th Hawaii International Conference on System Sciences, IEEE, 2001.

[5] John Grundy, "Storage and Retrieval of Software Components using Aspects", IEEE, 2001.

[6] Jihyun Lee, Jinsam Kim and Gyu-Sang Shin, "Facilitating Reuse of Software Components using Repository Technology, 10th Asia-Pacific Software Engineering Conference, IEEE, 2003.

[7] Daniel Lucredio, "A survey on Software Components Search and Retrieval", 30th EUROMICRO conference, IEEE, 2004

[8] Y.S.Maarek, D.M.Berry and G.E. Kaiser. An information retrieval approach for automatically constructing software libraries, IEEE transactions on software engineering, 1991.

[9] Ronaldo C.Veras and Silvi.R.L.meira, "Comparative Study of Clustering Techniques for the Organisation of Software Repositories", IEEE, 2007.

[10] Vanilson Arruda Buregio, Eduardo Santana Almeida,Daniel Lucredio and Silvio Lemos Meira, " Specification, Design and implementation of a Reuse repository", IEEE, 2007.

[11] Hanen Ben Khalifa, Oualid Khayati, Henda Hajjami Ben Ghezala, "A Behavioral and Structural Components Retrieval Technique for Software Reuse", IEEE Computer Society, 2008, pp-134.

[12] Wang Haitao and Chen Xing, "Study on a Component Library Model Based on the Four-Layer Architecture", IEEE, 2011.

[13] GE Junwei ,TAO Cong, FANG Yiqiu, "Architecture for Component Library Retrieval on the Cloud",IEEE, 2011, pp 536-539

[14] Lei Zhang, LichaonChen, Lihu Pan, Yingjun Zhang, "A Novel Approach of Component Retrieval in Large-Scale Component Repositories", IEEE, 2012.

[15] Zhu Jingbo, Yao Tianshun,"A Knowledge-Based Approach To Text Classification", IEEE, 2001.

[16] Panos Constantopoulus and Martin Dorr, "Component Classification in the software Informationa Base", Prentice Hall, 1995, pp-177-200.

[17] Oualid Khayati, Jean-Pierre Giraudin, "Components Retrieval Systems", 2001, pp-56.

[18] M.R. Girardi and B.Ibrahim, "Automatic Indexing of Software Artifacts", 1994.

[19] Chao-Tsun Chang, William C. Chu, Chung-Shyan Liu, Hongji Yang: " A formal approach to software components classification and retrieval," 264-269 Electronic Edition (IEEE Computer Society DL) Colin J. Hardy, Hlen M. Edwards, J. Barrie Thompson (1997).

[20] P. Chen, R. Hennicker, and M. Jarke," On the retrieval of reusable software components" (August 11-15, 1997).

[21] Pietro Abate, Roberto Di Cosmo, "Learning from the future of Component Repositories", ACM, 2012, pp-51.

IJERT