A Tool incorporating different Techniques for Effective Component Storage and Retrieval

Pankaj Vohra, Ashima Singh

Abstract— In software industry, there is progressive increase in reusability of software components. To increase productivity and to reduce time-to-market organizations are adapting for-reuse and with-reuse approach. In order to make reusable component available in future projects there must be a system to store software components so that they can be efficiently retrieved as per user requirements. So this work is toward, construction of component repository with efficient retrieval of software component that best fits user's requirements, while retrieving software component. Besides, it enhances reusability; reduces efforts of software development from scratch and increases productivity.

Index Terms— Component, Component Based Development, Component Repository, Repository, Retrieval, Exact Retrieval, Approximate Retrieval, Reuse.

1 INTRODUCTION

COMPONENT-BASED Software Engineering is a special case of software engineering for and with reusable assets; reusable assets have a special packing and are referred to as components. It focuses on reusing and adapting existing components [1]. Developing component based system doesn't follow the traditional software development process in which requirements are analyzed and components are identified. Component-based development focuses on development of software systems from already existing reusable components that can be reused in forthcoming projects.it is an approach to find, select, adapt, compose and replace components. A Software library is a set of software assets that are maintained by an organization for possible browsing and/or retrieval [1]. Especially, Component Repository is the tool that store reusable software components and make them available for future. So, for efficient storage components repository must have following features:

1. Component description
2. Storage Structure
3. Efficient storage of components
   a. Manual Storage of Components & description
   b. Component storage using "Tags"
4. Efficient retrieval of components
   a. Exact match
   b. Approximate match

This paper suggests the idea and technique for construction of Component Repository for efficient storage as primary goal and efficient component retrieval as secondary goal. This indirectly increases the reusability, productivity and accelerates development by following CBD process. It also summarizes the various techniques & mechanism for component storage and retrieval.

In the next section, basics of reuse system is briefly described like database management system , storage and retrieval; section 3 defines and suggests the techniques theory for storage of software components in component repository and how component storage is related to component retrieval.. Section 4 defines various component retrieval techniques like exact match and approximate match. Section 5, describes element of component repository. Section 6 describes life cycle of component repository. The goal is to design an appropriate component repository with efficient storage and retrieval of reusable software components.

- *Pankaj Vohra is currently pursuing masters degree program in software engineering in Thapar University, Patiala, India, PH-09463945520. E-mail: pankajvohra.se@gmail.com*
- *Ashima Singh is currently pursuing Ph.D. in Computer Science from faculty of Engineering, UCOE, Punjabi University, Patiala, India. E-mail: ashima@thapar.edu*

2 COMPONENT BASED DEVELOPMENT PROCESS

2.1 Component Based Development & Repository

CBSE (Component Based Software Engineering) is a process that emphasizes the design and construction of computer-based systems using reusable software components [2].

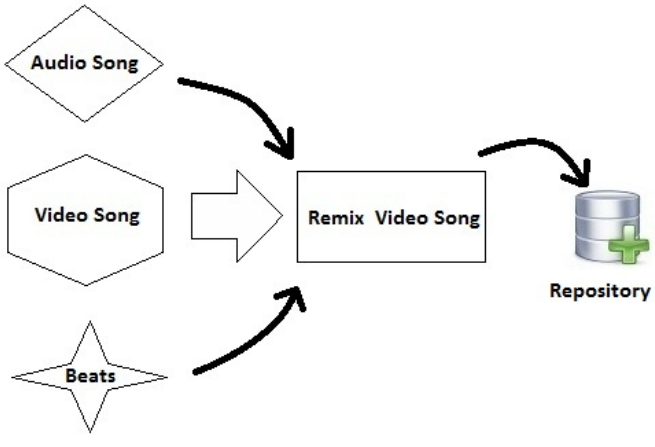

Fig. 1. Component Based Development Process.

Fig 1 describes various reusable components i.e. audio song and video song that are integrated with beats to form Complete System and stored in repository. Component Based Development confirms the increase in productivity and quality of system and enhances reusability.

2.2 Basics for Reuse System

The most common application of software repository is software reuse. Reusing software components must be supported by set of following elements:

1. A database that must be capable of storing software components and its classified information essential for its retrieval.
2. Classification scheme, so that user can quickly find what is required.
3. Automated library system with friendly GUI, for browsing and searching database.
4. Component retrieval system that enables client application to retrieve components.
5. Documentation: essential feature, which provides all the details about particular software component.

3 Techniques For Storage

3.1 Storage Component and its detail manually

Storage of component is an essential task from reuse point of view. Stored component can be further used in other projects only if detail of particular component is available. So a Component Repository provides a system to store components and its detail & description.

1. Store Component: The goal behind storing components is to make them available for future use.
2. Store Component descriptions: Storing description of components is another essential task because only on the basis of its description user can judge whether a component meets their requirement or not.

3.2 Component Storage using "TAGS"

Storing component in efficient way is a difficult task because retrieval and searching of component depends on its storage. Component storage mechanism must be efficient enough so that it provides all the suitable & relevant components against user query.

There is a method for storing components using Tags; Here Tags are keyword or set of keywords that can be stored by admin while storing components. So, Tags are just the keywords that may be user-input for searching component, i.e. if there is no suitable match of keyword in "component name" then same keyword can be explored in "Tags" in order to search relevant components as shown in Fig 4. "Tags" are also discussed in more detail in forthcoming sections.

4 Techniques For Retrieval

4.1 Exact Retrieval

1. Component Retrieval by Exact Keyword matching

In Keyword Retrieval user has to enter the set of keywords and a query string is passed to repository. It matches exact keyword with text that represents the component and display set of possible matches otherwise display no match found.

2. Component Retrieval by predefined Inputs

There must be a facility for user to set there priority to get relevant components from predefined inputs. It'll make intersection of all the inputs and display components that satisfy all the inputs.

4.2 Approximate Retrieval

1. Approximate Component Retrieval by using tags.

If there is no match in Exact Retrieval method, then we can make retrieval of component efficient by using Tags. It'll further explore the query string in Tags of each component. If it'll get any match in tags, then it'll display set of all relevant components and user can select most suitable component as per requirements. Here probability of meeting user criteria is very high.

2. Component Retrieval on the basis of few combinations of FIXED inputs

If there is no match in retrieval of components by predefined inputs using intersection of all the inputs, then repository can display result by considering few combinations with high priority. It'll display all the possible components that satisfy few of the predefined inputs.

5 Features of Component Repository

5.1 Component Specifications

A Software Component is a unit of composition with contractually-specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties [2]

Component specifications describe the functionality, properties and characteristics of a component. Fig 2 describes the metadata of a software component or its various parameters that are essential for classifying software components.

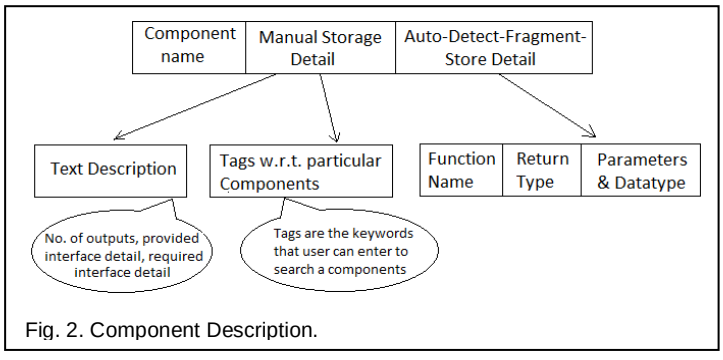

Fig. 2. Component Description.

Above hierarchy defines various parameters of components that can act as fields in component repository. A Component can be defined by Component Name; its manual description that can be entered by user while adding components and Automatic storage module of repository [3]. Manual description of components is the Text description like number of inputs and outputs, provided interface detail, required interface de-

tail etc. as described in Table 1. "Tags" are the collection of keywords that user can enter while storing components in Component Repository, these tags are useful for searching component with approximate keyword based search i.e. if no exact match of keyword found in component name then it will match keywords in Tags and display all the relevant components (including those components that doesn't contain keyword in Component Name). Besides these descriptions, a Source-code component may have their inner coding details as discussed in [3] Table 1 defines various essential parameters & their description of software components.

TABLE 1
COMPONENT SPECIFICATION

Parameters	Description	
Component Name	Name of Component	
Component ID	Unique ID number (Auto)	
Manual Description	**Tags**	**Text Description**
	Collection of Keywords that user may enter while retrieving a component (for optimizing keyword based retrieval)	Description about Component.
		Type (EXE, DLL, DOC etc.)
		Language in which component is developed. (C, C++, java, VB, C# etc.)
		Web-based, Desktop or database application
		Name of Required interface example: Inumber
		Name of Provided interface example: Ifactorial
		Inputs required by components
		Outputs provided by components
Automatic storage of source code components	Function Name Function's Return Type Parameter Arguments Parameter's Data Type Storage Location	

Here tags are just the collection or set of words.

5.2 Classified Storage Structure

Component Storage in a repository is a challenging task because component retrieval and component extraction depends on classification scheme of component storage and its information. Fig 3 describes the various tables that can be used to store components and their relevant parameters. In order to decrease complexity component and its description are stored separately in different tables.

Here, "comp" table contains component name, Component Id, Component type and tags (set of keywords) as described in Table 1, Tags are helpful in component retrieval if there is no exact match found in component name as shown in Fig 4. "Component Id" acts as primary key in comp table and foreign key in other tables. "compdesc" table contains text description of components that can be entered by user. "storagelocation" table contains the location path where component can be stored or retrieved. Component storage location can be further classified on component type for efficient retrieval. Detail of each component can be classified and explored using "Component ID".

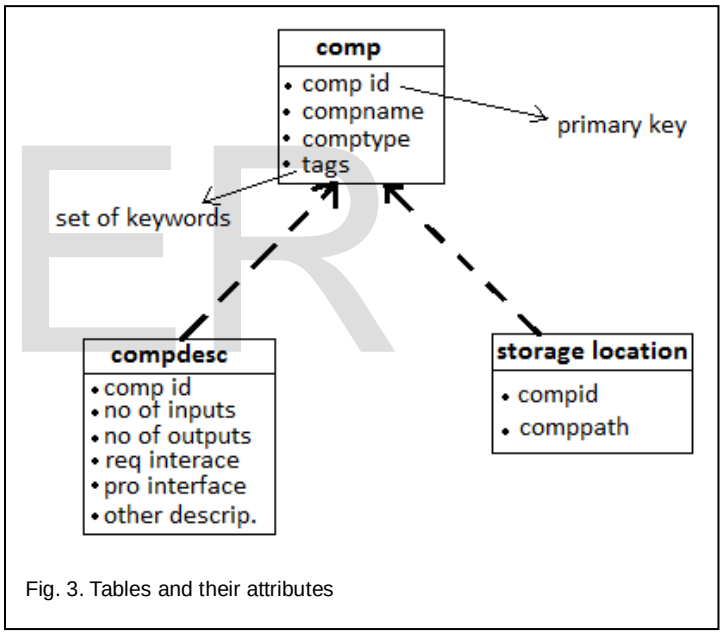

Fig. 3. Tables and their attributes

6 LIFE CYCLE OF COMPONENT REPOSITORY

6.1 Analysis

A Component Repository is developed; it supports storage, browsing and retrieval of software components. The primary goal of Component Repository is to design an appropriate component repository with storage capability to make them available in future. Secondary goal is to provide an efficient retrieval system to browse and retrieve components against user query.

Luqi and Jiang Guo [4], [5] conducted a survey of various repositories and concluded that most of the repositories have common feature of keyword based search. So, by keeping this in mind this commonly used feature is included in Component Repository.

6.2 Design & Flow Mechanism

For developing Compository, we used UML (Unified Modeling Language) for design and documentation [6]. First we wrote Use Cases, Use Case Diagram and Use Case reports. For each module flows, Activity diagram, Sequence diagram, Collaboration, State Chart diagram are written.

In keyword based searching, user first enters the set of keywords for searching component. Component Repository matches these keywords with text in component name and displays the exact possible matches. If it is unable to find any match, then it'll look up in Tags of each component as shown in above mechanism. If it found any match in Tags, then it'll display all the relevant Components otherwise display "No Match Found"

6.3 Implementation

Compository was constructed based on the survey done by Luqi and Jiang Guo [4], [5] and concept of Component-based software engineering [2], [7]. It is a Web-based tool having primary goal to store reusable software components, that can be used in future projects and secondary goal is to provide suitable software component which best meets user requirement

7 Conclusion

The concept of this paper explains the retrieval methods of software components from component repository in section 4, so that components can be efficiently retrieved with high possibility of meeting user requirements. It also explains the concept of reusability in section 2 and section 3 by storing components and to make them available for future reuse (i.e. with reuse and for reuse). Section 5 describes the structure or representation of software component in component repository. Therefore, efforts of developing components from scratch can be saved and productivity can be increased by reusing preexisting components.

References

[1] Mili, Mili, Yacoub, Addy Edward, Reuse-Based Software Engineering, AWiley-Interscience Publication, John Wiley & Sons, INC., 2002. 529 p.

[2] Sommerville Ian, "Software Engineering", 9th edition.

[3] Vohra and Singh, "Automatic Fragmentation and Storage of Code in Component Repository w.r.t their Input and Output Interfaces: A Tool" International Journal of Innovative Technology and Exploring Engineering, Vol. 2, Issue 3, pp-235-238

[4] Luqi and Jiang Guo, "Toward Automated Retrieval for a Software Component Repository", Proceedings of IEEE International Conference and Workshop on the Engineering of Computer Based Systems (IEEE ECBS), Nashville, USA, March 7-12, 1999. Pp. 99-105

[5] Luqi and Jiang Guo, "A Survey of Software Reuse Repositories", Research supported by ARO(38690-MA) and DARPA(99-F759)

[6] Grady Booch, James Rumbaugh, Ivar Jacobson "The Unified Modeling Language User guide", 2005

[7] Roger S. Pressman, Software Engineering – A practitioner's Approach, McGraw- Hill International Edition, 6th Edition 2001