International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-2 Issue-3, February 2013

Automatic Fragmentation and Storage of Code in Component Repository w.r.t their Input and Output Interfaces: A Tool

Pankaj Vohra, Ashima Singh

Abstract*— Develop once, Use once is the common approach followed by software developers in software industry. Lot of software development effort, cost as well as time is wasted if the software components can't be reused further. These efforts can be saved by reusing available components. In order to increase the availability of reusable components (with or for reuse), we need an effective component storage structure i.e. Repository and also automatic component addition and extraction from Component Repository. Auto-Detect-Fragment-Store is a staged technique which is used for the automatic storage and retrieval of source-code components. A component repository is constructed which stores automatically, fragmented code on the basis of inputs and outputs. This work is an effort towards automating component storage, its addition and retrieval in a way component can be reused and software reusability is enhanced.

***Index Terms*: Component, CBD (Component Based Development), CBSE (Component Based Software Engineering), Component Repository, Repository.**

I. INTRODUCTION

CBSE (Component Based Software Engineering) is a process that emphasizes the design and construction of computer-based systems using reusable software components [1]. It focuses on reusing and adapting existing components. CBD (Component-based Development) provides idea to develop software systems from pre-existing reusable components, build component that can be reused in future project development. CBD is an approach to find, select, adapt, compose and replace components. Especially, Component Repository is the tool which provides and store reusable software components that are desired by user. So, for efficient storage components repository structure can be classified by:

1) Component Specifications
2) Storage Structure
3) Method/technique of storage
4) Method/technique of retrieval

This paper suggests the idea and technique for construction of Compository (The Component Repository) as implemented product for automatic storage and retrieval of source code components which indirectly increases the reusability by following CBD process. It also summarizes the various fields and Storage structure in component repository that are essential for classifying different software components. In the next section, related works with CBD is briefly described such as repository, their features and section 3 defines and suggests the techniques theory for repository construction to support and realize the CBD process. Section 4 defines and suggests the component description & specifications and storage structure. Section 5, theory for developed Compository (The Component Repository).

The goal of "Compository - The Component Repository" is to design an appropriate component repository with addition and retrieval of components on demand so that they can be reused in future projects. "Compository" finds its suitable application in any software development organization that is interested in reusing its developed components and also wants to store other components by decomposing them automatically into its fundamental units like its function name, input parameters and output parameters. Organizations can make use of its own generated components if they have a suitable storage structure for these components. Appropriate storage structure of components makes reusability effective and easy.

II. COMPONENT BASED DEVELOPMENT PROCESS

A. Component Based Development & Repository:

CBD is the process of identifying & selecting reusable components, qualifying, adapting and compose to form a new reusable component.

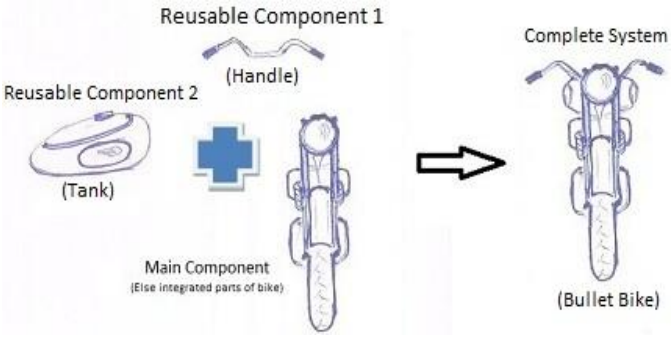

Fig 1 Component Based Development Process

Fig 1 describes various reusable components i.e. Component 1 and component 2 that are integrated with main component to form Complete System.

Manuscript published on 28 February 2013.
***Correspondence Author(s)**
Pankaj Vohra, Computer Science and Engineering Department, Thapar University, Patiala, India.
Ashima Singh, Computer Science & Engineering Department, Thapar University, Patiala, India.

Component Based Development ensures the increase in productivity and quality of system and also improves time-to-market in software development. It also provides maintenance by replacing component with another component.

B. *Essential features of Repository:*

1) Automated library system with friendly GUI, for browsing, searching and retrieval of assets.

2) Standard component framework to include purpose, functional description, certification levels, historical results of usage etc.

3) Classification scheme: so that user can quickly find what is required.

4) Documentation: Most important feature, which helps user in selecting component that best suits requirements.

III. TECHNIQUES FOR STORAGE & RETRIEVAL

A. *Manual Storage of Components*

User can upload and enter the appropriate detail of each component manually, e.g. .exe, .dll components.

1) Upload Component: Components must be uploaded and stored so that they can be extracted and reused later.

2) Store Component descriptions: Repository must be able to record component description so that user can find the suitable component as per his/her requirement.

B. *Auto-Detect-Fragment-Store*

Algorithm is developed for automating storage of source-code components as shown in Fig 5, which first detect the functions used during its developments and fragments their details as follows:

1) Upload Source-code component: uploading of component is necessary so that it can be fragmented.

2) Auto-Detection of interfaces: Auto-Detection of inner details of source-code component.

3) Fragmentation of Source-code: Fragment source-code components by Function inputs:

- Function names
- Function's return type
- Parameter Arguments.
- Parameter's data-type.

4) Store Inner detail of Source-code: All the fragmented details as well as source-code component must be stored in repository so that while retrieval user can easily get & reuse the coding details of source-code components.

C. *Component Retrieval by Keyword Search*

In Keyword Retrieval user has to enter a keyword and a query string is passed to repository. After keyword matching repository provides set of possible matches of components, from where user can select the required one.

D. *Component Retrieval by Fixed Inputs*

Retrieval by Fixed Inputs: Repository can have the functionality to retrieve components through Fixed Inputs as follows:

- Component Type
- Component Language
- Component Domain

IV. INSIDE COMPONENT REPOSITORY

A. *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 [1]

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

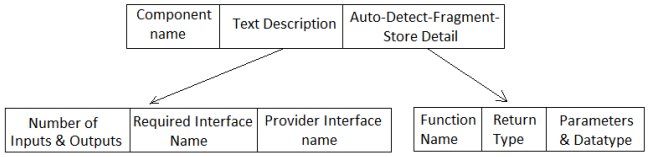

Fig 2 Component Description

Above hierarchy defines various parameters of components that can act as fields in component repository. A Component can be defined by Component Name and its description like number of inputs and outputs, its required interface, provided interface. Besides these descriptions, a Source-code component may have their inner coding details like various Functions, their Return Types, Parameters and Datatype, that are considered while developing source-code component. Table 1 defines various essential parameters & their description of software components. Whereas Table 2 defines attributes that can be entered automatically using Auto-Detect-Fragment-Store technique as discussed in previous section.

Table 1 Manual Entry, Text Description of Software Component

Parameters	Description
Component Name	Name of Component
Component ID	Unique ID number (Auto)
Component Description	Description about Component.
Component Type	Type (EXE, DLL, DOC etc.)
Component Language	Language in which component is developed. (C, C++, java, VB, C# etc.)
Component Domain	Web-based, Desktop or database application
Required Interface Name	Name of Required interface example: Inumber
Provided Interface Name	Name of Provided interface example: Ifactorial
Number of Inputs	Inputs required by components

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

Number of Outputs	Outputs provided by components

Table 2 Auto-Detect-Fragment-Store detail of Source-code Component

Parameters	Description
Function Name	Function used in side Source-code Component
Function's Return Type	Return type of that function
Parameter Arguments	Arguments of that function
Parameter's Data Type	Data type
Storage Location	Path of drive where component stores or from where component can be retrieved.

B. *Storage Structure*

Component Storage in a repository is a challenging task because information retrieval or extracting component all depends on how components and their description are stored in repository. Fig 3 describes the various tables that can be used to store components and their relevant parameters. For easy understanding component and its description are stored separately in different tables.

e.g. "comp" table contains name of component, Component Id (Auto-generated), Component type (exe, dll, source-code, and document), Language and its Domain as described in Table 1, where "Component Id" acts as primary key.

"compdetail" table contains manual details of components that can be entered by user, whereas "compfunction" table contains inner details (as described in Table 2) of source-code components that are stored by Auto-Detect-Fragment-Store technique.

"storedlocation" table contains the path of drive where component will get store or from where component can be retrieved. Component storage location can be further classified on component type for efficient retrieval. Detail of each individual software component can be retrieved by its "Component ID".

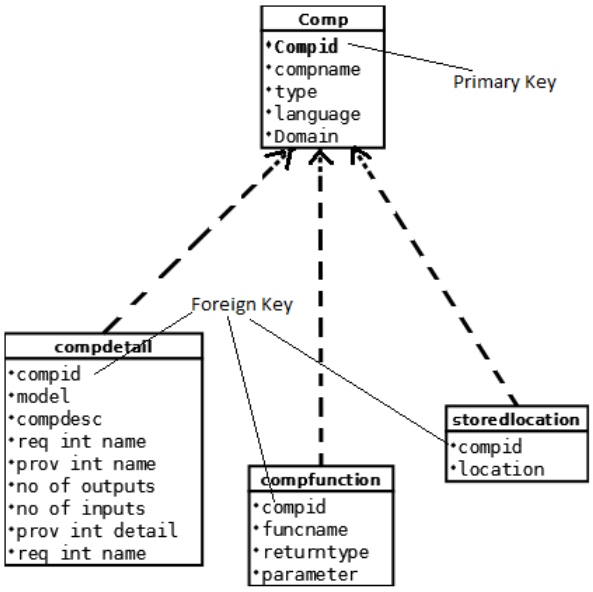

Fig 3 Various tables and fields for storage.

V. System Model For Component Repository

A. *System Analysis*

System model of repository, Compository (The Component Repository) is developed; it supports storage and retrieval of software components. The goal of "Compository - The Component Repository" is to design an appropriate component repository with addition and retrieval of components on demand so that they can be reused in future projects. "Compository" finds its suitable application in any software development organization that is interested in reusing its developed components and also wants to store other components by decomposing them automatically into its fundamental units like its function name, input parameters and output parameters. It is also capable of retrieving components as desired and demanded.

Luqi and Jiang Guo [2], [3] conducted a survey of various repositories that are used by government as well as commercial organizations and concluded that most of the repositories are web-based, have common feature of keyword search. So, by keeping this in mind both these features are included in Compository (The Component Repository).

B. *Design*

For developing Compository, we used UML (Unified Modeling Language) for design and documentation [4]. 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. Fig 4 represents the State Chart diagram Compository (The Component Repository) for Auto-Detect-Fragment-Store technique.

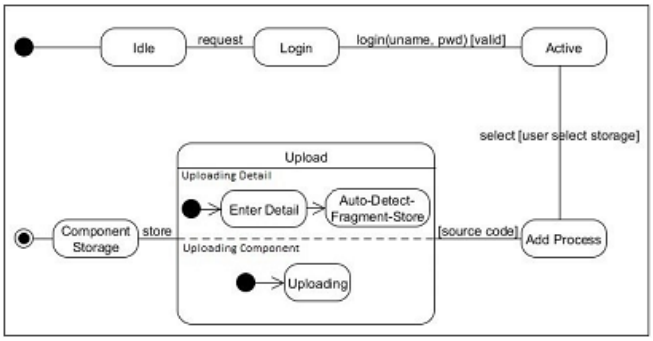

Fig 4 State Chart Diagram for Auto-Detect-Fragment-Store

C. *Execution*

Compository was constructed based on the survey done by Luqi and Jiang Guo [3] and Component based software engineering. It is a Web-based tool to store, add, maintain and retrieve software components like source-code, Executable, dll and documents. Fig 5 describes the manual entry text description of software component. We focused on Automating storage of source-code Components. Its special feature "Auto-Detect-Fragment-Store" efficiently & automatically stores in-house developed as well as off the shelf (COTS) source-code components in component repository as shown in Fig 6.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

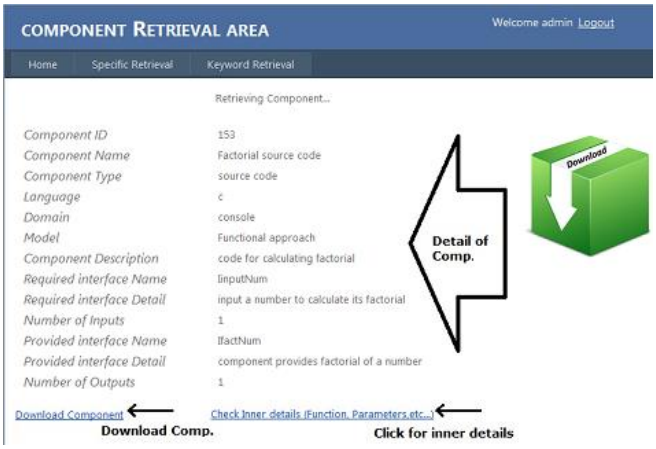

Fig 5 The manual entry text description of software component.

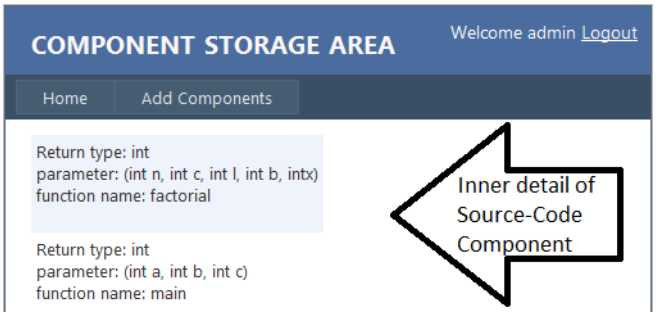

Fig 6 Inner Details of Source-code Component (by Auto-Detect-Fragment-Store technique)

D. Future Scope

Future work would be proceeding on retrieval of Software component and also on efficient classification of different software components.

VI. CONCLUSION

The work undertook in this paper discusses the Auto-Detect-Fragment-Store approach which automatically fragment the code and gives the input output interfaces. These interfaces are stored in component repository. The aspect of reusability is that the Compository generated is explored by developer for function name, input, output interfaces. Further, Software developer can reuse the components by referring to components and its detail information stored in repository. This approach actually makes the use of components much easier. Therefore software development efforts in software industries do not go in vain.

REFERENCES

1. Sommerville Ian, “Software Engineering”, 9th edition.
2. 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.
3. Luqi and Jiang Guo, "A Survey of Software Reuse Repositories", Research supported by ARO(38690-MA) and DARPA(99-F759).
4. Grady Booch, James Rumbaugh, Ivar Jacobson “The Unified Modeling Language User guide”, 2005
5. Roger S. Pressman, Software Engineering – A practitioner’s Approach, McGraw- Hill International Edition, 6th Edition 2001.

Pankaj Vohra is Post Graduate student in Computer Science & Engineering Department, Thapar University, Patiala., India and pursuing Master of Engineering in Software Engineering. He holds Bachelor of Technology (B.Tech) degree in Computer Science & Engineering from Guru Nanak Dev University, Main Campus, Amritsar, Punjab, India (2010). His publication: “A contrast and comparison of modern Software Process Models”, in International Journal of Computer Applications (IJCA) to be published.

Ashima is Assistant Professor in Computer Science and Engineering Department, Thapar University, Patiala, India; and pursuing Ph.D. in Computer Science from Faculty of Engineering, UCOE, Punjabi University, Patiala, India. She holds a Bachelor of Technology (B.Tech.) degree in Computer Science and Engineering from GZSCET, Bathinda, India (2001). She obtained her Master of Technology degree in Computer Science from Department of Computer Science and Engineering, Punjabi University, Patiala India (2005). Her research interests include Software Process Reengineering, Software Engineering, Agile Software Development, Software Quality Improvement in Small Scale Enterprises, Software Reuse, Software Process Customization and Automation, and Software Process Metrics.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)