ISSN : 0976-8491 (Online) | ISSN : 2229-4333 (Print)

Aspect of Reusability in CBSE

[1]Vaneet Kaur Bhatia, [2]Dr. D.S Dhaliwal, [3]Dr. Sawtantar Singh
[1]Dept. of CSE, PTU Campus Bathinda, Punjab, India
[2]Dept. of CSE, BGC Sardulgarh, Punjab, India
[3]Dept. of CSE, BMSCE Muktsar, Punjab, India

Abstract
The Component-Based Software Engineering (CBSE) or Component-Based Development (CBD) emphasizes the development of applications based on components so that the applications are easy to maintain, and extend. CBSE is a process that aims to design and construct software systems using reusable software components. Component-Based Software Engineering (CBSE) refers to the construction of large software systems from existing parts. The main idea of using component-based development is reusability. The biggest challenge facing component based software reuse is managing a large number of stored reusable components efficiently to allow fast allocating and retrieving. A software repository is a storage location from which software packages may be retrieved and installed on a computer. Repositories built with reuse in mind can be considered as special-purpose information systems, required to support powerful semantic modeling, flexible retrieval of varied software descriptions and efficiency optimization directed towards a large variety of classes rather than large populations per class.

Keywords
Repository, Retrieval, Component, Reusability, Query

I. Introduction

The goal of component-based software engineering is to increase the flexibility, productivity, quality, and time-to market in software development. CBSE uses software engineering principles to apply the same idea as object oriented programming to the whole process of designing and constructing software systems. It focuses on reusing the existing components, by just coding in a particular style.This approach to code reuse reduces the production costs and enhances the maintainability of the software system [9].
CBSE is different as compared to other development strategies in a sense that when a component is developed it is intended that it can be integrated with other systems. The main idea of using component-based development is reusability. The biggest challenge facing component based software reuse is managing a large number of stored reusable components efficiently to allow fast allocating and retrieving [5]. CBSE encourages the composition of software systems, as opposed to programming them [14]. The software systems built using CBSE are not only simple and cheaper but usually turn out to be more robust, adaptable and updateable. Component-based systems emphasise appropriate reuse and composition of software components as a key concept. However, finding and reusing appropriate software components is often very difficult, particularly when faced with a large collection of components and little documentation about how they can and should be used [12]. Software engineering data contains different type of documents like requirement analysis, design, testing, test runs, source code and bugs details. All these data sets are available in the projects own repository and developers of the same organization can use or refer them in future[1].

II. Component as an Effective Software Reuse

Brown and Wallnau describe a Software component as “a unit of composition with contractually specified and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties” [11].
The Component-Based Development, also called development “with reuse”, deals with the construction of application. These applications are composed by reusing existing components. If needed, new components can be developed, or even acquired from third party COTS (Commercial Off-The-Shelf) .COTS comes in a variety of types and levels of software, e.g., components that provide specific functionality such as subroutines, classes, frameworks, and even complete applications or tools used to generate code such as domain-oriented language processors and application generators. Many companies are providing reusable off-the-shelf components for specific domains, and those components can be purchased by developers from the market. As this trend continues, programmers may be able to create their own systems in the future by integrating components from different component vendors.
Software Reuse is the process of implementing or updating software systems using existing software assets [10]. Anything that is produced from a software development effort can potentially be reused. Designs requirements specifications, development processes, and decision making are applicant for reuse [13]. Reusable software components have both short-term and long-term benefits for the development of software systems. Short-term benefits are the instant benefits that a programmer achieves during the implementation of a programming task. In long-term benefits the programmer does not reuse the components, but they extend to the whole life cycle of the software system and to later programming activities of the programmer. The benefits of software reuse are increased productivity, quality and shorter time to market [2].
Software reuse is the use of existing software to construct new software [14]. Reusable assets can be either reusable software or software knowledge. Reusability is a property of a software asset that indicates its probability of reuse. The main purpose of software reuse is to improve software quality and productivity.

A. Software Repository and its Importance

Repository commonly refers to a location for storage, often for safety or preservation. In the simplest term, a repository is nothing more than a collection of knowledgeable assets. A repository is a place where the data is stored. It is often stored in relational databases but can be stored in the place that can hold the data: hierarchical databases, Excel spreadsheets or even text files.A relational database is a distinct type of database where data is stored in related tables. Not all repositories are stored in relational databases, but many are, for example an XML repository is a system for storing and retrieving XML data.
Repositories may be exclusively for particular programs, such as CPAN for the PERL programming language, or for an entire operating system. Operators of these kind of repositories typically provide a package management system, tools for searching, installing and otherwise manipulate software packages from the

repositories [4]. Software repositories contain a wealth of valuable information about the evolution of a software project.
Repositories play a pivotal role in an integrated reuse- based application development environment. Reusing software components implies the storage and maintenance, and the ability to efficiently find these components.
This methodology is used for organizing the reusable components in the repository. It is commonly agreed that software reuse does not only concern code, it also implies the simultaneous reuse of objects associated with it, plus information about the objects. Therefore, designs requirements, development processes, and decision experiences are also applicant for reuse [6].

B. Software Component Storing and Retrieval

Reusability is a process of utilizing and applying already developed components. So there are many work products that can be reused, for example designs, specifications, source code, architectures and documentation. Successful reuse requires having a wide variety of high quality, stable components, proper classification and retrieval mechanisms. Users of the system have access to appropriate components if it requires effective software reuse. 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 [8]. The user must access these components precisely and rapidly, and if necessary, be able to modify them. When the users classify software it allows reusers to organize collections of components into structures so that they can be searched easily. Many retrieval methods require some kind of classification of the components. Four different categorization techniques had been previously engaged to construct reuse repository namely Attribute Value, Free Text, Enumerated, and Faceted classifications.

C. Software Component Querying, Browsing and Searching

Querying and browsing are the two major information access mechanisms for most users. Querying is direct that users formulate a query and the system returns information matching the query. However to formulate a query is a quiet difficult task because users have to overcome the gap from the situational model to the system model. In browsing, users determine the usefulness or relevance of the information currently being displayed in terms of their task and traverse its associated links Mili et al. [9] claims that browsing is the most predominant pattern of component repository usage because most programmers often cannot formulate clearly-defined requirements for reusable components so they rely on browsing to get familiar with available reusable components in the repository.
Searching is direct and fast. Software developers formulate a query, and the repository system returns components that match the query. Formulating queries is a cognitively challenging task because software developers have to overcome the gap from the situational model to the system model (i.e., the description of the components in the repository) [15].

D. Effective Retrieval Mechanism in Retrieving

Storage and representing reusable software components in software repositories to facilitate convenient identification and retrieval has been always a concern for software reuse researchers.
A retrieval technique is concerned with the comparison of the representation of a query and the representations of documents for the purpose of identifying and or retrieving those documents that are relevant to that query. A retrieval technique is not directly concerned with the way of actually representing the documents and queries, but different ways of representation may imply using different retrieval techniques [14].
An effective retrieval mechanism including a representation schema for indexing and a matching criterion between a query and a component is essential [7]. Retrieval mechanisms play an important role in locating reusable components that match reuse queries. In the retrieval process, the Information Retrieval system extracts the documents or more generally, the pieces of information, which will presumably answer the information need formulated by the user. This retrieval process is usually separated into a preliminary step of indexing the documents, followed by an operational step of retrieval. The retrieval process can be iterative, if the user provides some feedback on the retrieval results.

III. Conclusion and Future Scope

The paper presented the study on the ways to make the reusable components retrieved with more ease to the users. The basic step in reusing already developed software artifacts is to build a library of such components. Such library is not just a collection of software artifacts but it is built with the objective in mind that the components in such a library will be stored and retrieved for the purpose of reuse. Components should be stored are developed in such a way that these become more and more reusable. To make such a reuse library requires some different method for storage and retrieval of software components. There is a need of user friendly interface which provides the best solution for organization of the components so that it can be reused by the way it is required. Software components are one of the major factors that provide the software reusability. For effectively reusing the components from the repository, the selection of proper retrieval technique is essential. Classifications schemes in different repositories have different impact on retrieval. Also various ways of optimizing system speed and efficiency must be explored in order to keep the repository as an effective design tool.

References

[1] Nida Yasir, Bushra Jamil, Javed Ferzund,"PDCML: A Model for Enhancing Software Reusability", IJSEA Vol. 7, No 1, pp. 123-136, 2013.
[2] S. Singh, S. Goel,"CBSE versus COTS Based Software Development", IJARCSSE, Vol 2, Issue 10, pp. 29-32, October 2012.
[3] P.Niranjan, C.V.Guru Rao,"A mock- up tool for software component reuse repository", International journal of software engineering and applications (IJSEA), Vol. 1, No. 2, April 2010.
[4] M. Hopfner,"Source Code Analysis Management and Visualization for PROLOG", Dissertation for science doctoral degree, Julius, Maximilians, Universitat, Wurzburg, 2008.
[5] S.Shiva, L.Shala,"Using Semantic Wikis to Support Software Reuse", Journal of software, Vol. 3, No. 4, pp. 1–8, April 2008.
[6] W.B.Frakes, K.Kang,"Software reuse research: status and future", IEEE Transactions on Software Eng 31, Vol. 7, pp. 529–536, 2005
[7] N.Kaur,"Retrieving Best Component from Reusable Repository", pp. 23, 2005.
[8] J.C.Grundy,"Storage and retrieval of Software Components using Aspects", In Proc. of the 2000 Australasian Computer Science Conference, Canberra, Australia, IEEE CS Press, pp.

95-103.

[9] Mili, S.Yacoub, E.Addy, M.Hafedh,"Toward an Engineering Discipline of Software Reuse", IEEE Software Vol. 16, No. 5, pp. 22–31, 1999.

[10] F.Church,"Software Reuse Executive Primer", Department of Defense, April 1996.

[11] P.C.Clements,"From Subroutines to Subsystems: Component-Based Software Development", American Programmer, Vol. 8, No. 11, November 1995.

[12] W.Frakes, T.Pole,"An empirical study of representation methods for reusable software components", IEEE Trans. Softw. Eng.Vol. 20, No. 8, pp. 617–630, 1994.

[13] P.Constantopoulos, M.Doerr, Y.Vassiliou,"Repositories for Software Reuse: The Software Information Base", In Proceedings of the IFIP Conference in the Software Development Process, Como, Italy, 1993.

[14] T.R.G.Green,"Programming Languages as Information Structures, in Psychology of Programming", J.-M. Hoc, et al., (eds.) Academic Press: New York, pp.118-137, 1990.

[15] B.Curtis, T.J.Biggerstaff, A.J.Perlis,"Cognitive Issues in Reusing Software Artifacts", Software Reusability, Vol. 2, pp. 269-287, 1989.

Vaneet Kaur received her B.Tech degree from MIMIT,Malout in 2008, M.Tech degree in Software Engineering from Thapar University Patiala, India in 2011.Currently pursuing the Ph.D. degree in Software Engineering from PTU Jalandhar. Presently, teaching at Giani Zail Singh PTU Campus, Bathinda, Punjab as Assistant Professor in the Computer Science Department from last two years. Her research interests include Software Engginering and Testing.

Dr. Dalvinder Singh Dhaliwal received his B.Tech[CSE] from Punjab Technical University,Jalandhar, India and M.Tech[CSE] from Punjabi University Patiala, India. He received his PhD in Computer Science & Engineering from Punjab Technical University, Jalandhar. He is currently working as a Director at Bharat Group of Colleges, Sardulgarh, Punjab, India. He has more than 12 years of teaching and research experience. He has published more than 30 research papers in various referred International journals and conferences. His current research interests are data mining, machine learning, cloud computing, mobile computing, ad-hoc networks, software engineering and digital image processing. Currently 16 research scholars are pursuing their PhD under his guidance.