CHARACTERIZATION OF REUSABLE SOFTWARE COMPONENTS FOR BETTER REUSE

Anshul Kalia[1], Sumesh Sood[2]

[1]*Research Scholar, Department of Research, Innovation & Consultancy, Punjab Technical University, Kapurthala – 144601, India*

[1]ansh.kamal007@gmail.com

[2]*Head of Department (Computer Applications), Swami Satyanand College of Management & Technology, Amritsar – 143001, India*

[2]sumesh64@gmail.com

Abstract

The reusable software components can be defined in several ways. The reusable software components possess a distinct functionality that does not affect the functionality of other components. It has also been specified precisely that for what the component reuse stands for and for what the component reuse does not stands for. It is required to characterize the components for better reuse. Characterization describes the features and characteristics of components. Distinct components show distinct characteristics in different domains of their usage and in different operating environments. The components can be classified on the basis of features it have, that facilitates with the better usage, better retrieval, better understanding and better cataloguing. Through component classification one gets the assurance of selecting right component and it suggests various ways in which a component can be reused. The paper explains for the need of characterization which ultimately is reflected from the above stated fact. It also provides with the criteria to characterize the reusable components. The criterion is laid down while keeping in mind the general specifications and formal specifications. These specifications in one way determine the feasibility of a reusable component at the initial level. The general and formal specifications depict the internal and external characteristics which play a significant role in the identification, selection, adoption and implementation of components in a particular application development. Specifications will tell the true character of a reusable component that helps to outline the required components. It also discusses the influence of characterization that it puts on the reuse of reusable components.

Key Words: *Component characterization, Repository, Cataloguing, Specifications.*

----------***----------

1. INTRODUCTION

Before characterizing reusable components it is required to understand what a component and a reusable component is actually. There exist several definitions of software components. Some considers the components as Javabeans, COM, DCOM, OLE, CORBA objects, while others considers the components as a fragment of source code or a functional procedure [11]. If a component is considered from the reuse point of view then it can be defined as an independent part of the module/application that can be replaced easily and it provides a distinct function in such a way that it should not affect the functionality of other modules, that is, they should not have any dependency on the replaced component [15]. The reuse of a component is misconceived several times, so it must be clarified to what a component reuse accounts for and to what a component reuse does not accounts for. The component reuse accounts for [7]:

- ✓ If a software component artifact is acquired or used from another context that otherwise have been written originally.
- ✓ If something is written originally for another project but is incorporated with or without changes to suite the requirements of some other project,
- ✓ If open source software is adapted and integrated into another project then the one it was intended to be used.

The component reuse does not account for [7]:

- ✓ If the software component is used repeatedly for the same project, then it is not regarded as component reuse.
- ✓ The development of new product versions is not regarded as component reuse.
- ✓ If the commercially off – the – shelf (COTS) software is used, then it does not account for component reuse.

Now, it is required to define the characterization. Characterization is a process of describing the qualities or features of a component. These qualities or features represent the true character of the components which is then used to classify the components based on these characteristics. The components of different domains and different operating environments exhibit different characteristics. This complete process certainly helps in managing the components and the component based systems as well. It provides the basis to classify and to categorize the components, in order to locate them easily in component repository qualitatively [12].

2. LITERATURE SURVEY

In 1999 Yacoub et. al taken an initiative to ponder the problem of component characterization. It elaborates the fact that how component characterization can be used to classify the components. It comes up with different techniques to characterize the components. It also explains what impact does the component characterization puts on the systems that are to be developed using reusable software components [15].

Gill in 2006 puts an emphasis on the importance of component characterization. It discusses the need of component characterization that ultimately helps the developer in software reuse. The study raised the point of testing of third party component. In the absence of proper characterization and specifications of components it becomes difficult to test such components [4].

Sandhu et. al in 2007 proposed a new characterization scheme for reusable software components which was based on information retrieval theory. It further pushes for the coding of component characteristics, in order to effectively manage the components. It also proposes a scheme, i.e. uncontrolled vocabulary and automatic indexing of software components for easy retrieval of reusable components from component repository [12].

In 2007 Sharma et. al discusses the issue of managing component – based systems with reusable components. The findings of study include the characterization of component and the basis of characterization. It also includes the benefits of component characterization such as improved cataloguing, improved usage, improved retrieval and understanding of components [13].

In 2013 Alkazemi et. al conducted a study to characterize the reusable software components. The study highlights the key characteristics of reusable components. It also proposes guidelines for deriving the design of reusable software components [6].

3. NEED OF COMPONENT CHARACTERIZATION

Component characterization is needed to optimize the storage, retrieval and management of components and component based systems. The components are classified based on their characteristics. This classification enables the better management of components. This can be understood well with the help of an example. Consider the library of a large university. It contains a number of books belongs to different subjects, periodicals, journals etc. All these entities are required to be stored effectively so that they can be retrieved and managed easily and quickly. It is required to categorize these entities for better use. This categorization is done considering classification scheme for libraries which takes into account the library code of an entity, author name, keywords, other index entries etc. All these classification criteria enable the user to make an optimum use of library services [1]. Similarly, the reusable components need to be characterized in order to make an optimum use of them. On the basis of this characterization these components can be classified accordingly, which may lead to better storage, retrieval and management of reusable components. The effective management of components assists the one in selecting the right component as per the requirements. This classification may also lead to better understanding of components, which certainly proves to be a useful asset while using and implementing the components [4].

4. CRITERIA OF CHARACTERIZATION

The criteria of characterization of reusable software components provide the basis to describe the characteristics or features. There exist reusable components of different nature which lies in different domains and operates in different operating environments. Depending upon the different conditions these components exhibits different characteristics. But in general they can be characterized into two specifications such as formal specifications and informal specifications.

4.1. Formal specifications

It entitles the features of reusable components that are taken into account while considering the technical aspects. The formal specifications specify the internal and external characteristics of a reusable component. It is the formal

specifications that are considered primarily while retrieving the components. The feasibility of a reusable component to be reused in certain project is determined at the very initial level on the basis of formal specifications. Formal specifications can be further divided into two parts depending on the nature of components, i.e. internal specifications and external specifications [15].

4.1.1. Internal specifications

Internal specifications state the internal aspects of a reusable component. It refers to the language used for development, the architecture, design descriptions, system abstraction, encapsulation etc. which are listed below:

- **Encapsulation**

The components developed these days are large and complex in size. It becomes necessary for components of such nature to hide the significant credentials of them to maintain the security aspect and to raise the reliability aspect of the component as well. Encapsulation can be performed on different areas of the components such as design descriptions, execution control descriptions, test plan descriptions etc.

- **Component type**

The component type displays the true nature of component. That is, if a component is specification component, design component, code component, executable component. A specification component gives the expected functionality of it. It helps programmer in implementing the reusable component in different programming languages. Design components can be used as a building block in constructing frameworks. A code component is a piece of code that can be used in building other applications. Executable components are executable piece of application whose source code is generally not accessible [6].

- **Architectural aspect**

The architectural aspect gives the internal structure of the reusable components. It shows different elements involved in the component. The architectural aspect is a representation of blueprint of different elements in a component and an exhibition of collaboration between them [6]. That is, it shows how the different elements involved in a component collaborate with one another.

- **Accessibility to source code**

Accessibility to source code is both beneficial and dangerous as well. It is beneficial because in some situations it is required to modify the reusable components according to the requirements of the application to be developed. In that case, it required to have an access to the source code. Without accessibility to source code it would not be possible modify the components [10]. It is dangerous because access to source code may put an adverse effect on the security. There also remains the possibility that the component may lose the trust that was established initially by the provider of the component. So, it needs to be dealt with due care.

4.1.2. External specifications

External specifications specify the external nature of the reusable components. It reflects how the components will be interacting with the other components or the other modules of applications. Some of the features that characterize the external nature of components are listed below:

- **Function**

A component must be characterized by the function it performs that specifies what certainly it has to offer for the other applications. It is significant in evaluating which component has execution control at any instant of time. The functions that can be performed by a component are of two types in nature, i.e. active and passive [3].

Active – Active functions affects and are affected by other functions. eg: GUI.

Passive – Passive functions are only affected by other functions, they cannot affect the other functions. eg: Databases.

- **Technology**

The technology feature of a reusable component should specify the technological platform on which it is built. Apart from that it should also specify the other possible technological platforms on which it can be reused. The dependency and independency of the component on the technology in both the cases (that is, technology on which it has been developed and possible technological platform on which it can be developed) should also be specified [9].

- **Integration framework**

This feature should specify the framework on which a reusable component runs along with other possible frameworks on which it can run. Framework helps the components to integrate with each other. Sometimes the components integrate with the application directly but sometimes they integrate with the application through the framework that establishes the virtual connection between them [14].

- **Interoperability**

Interoperability feature defines the interoperable character of the components. Interoperability occurs due to the application interfaces, which makes the interaction possible between the components. It is significant from the view that there exist other components as well and a component is required to interact with other components when the time comes. Often it is said that the reusable components should have less coupling. So, interoperability provides a procedure of de – coupling.

- **Portability**

The portability characteristic of a reusable component is necessary. It must specify the current platform of a component on which it runs and also the way it can be ported to other platforms. While developing a component for reuse it cannot be defined on which platform a component will have to run in future.

- **Non – functional characteristics**

The non – functional characteristics specifies the processor requirements, run time requirements, security features, authentication controls, storage space required, execution time etc.

4.2. Informal specifications

It enlists the features of reusable components that are taken into account in general but have no effect while considering the technical aspects. It proves to be a useful criterion when it is required to outline a reusable component or a set of similar reusable components that need to be retrieved without getting into the technical and functional details of the components. Some of the informal specifications are [15]:

4.2.1. Context

It defines the probable situations or the conditions in which the software component can be reused. Rather it is a difficult task to predict the situations either generally or specifically or both where it has the probability for the components to be reused. Probable target domains and applications should be mentioned when considering context as a component characteristic [7].

4.2.2. Level of reuse

It helps in determining at which level or at which phase of development of software lifecycle a component can be reused. By evaluating the level of reuse the component can be inducted easily into the application in which it is selected to be reused.

4.2.3. Agility

The age of a reusable component is considered from the moment when it is initially inducted into the component repository. The component that is induced initially has a more risk of reuse, while the component that has been reused quite a several times has a reduced risk of reuse associated with it. Also the age of the reusable component reflects the maturity and stability of the component.

4.2.4. Intent of reuse

It specifies the objective of reusing a component. It mainly comprise with the problem solving capability of the reusable component [8]. By using this characteristic the components that have similar problem solving ability can also be displayed.

4.2.5. Source of component

It specifies who supplies the reusable components. The reusable component can be either supplied from the internal component repository of the application developing organization or supplied from the repository that lies outside the application developing organization, it can be either an organization who developed the reusable component originally or it can be a component vendor [8].

5. INFLUENCE OF CHARACTERIZATION OF REUSABLE SOFTWARE COMPONENTS

The influence of characterization on reusable software components is stated as under [13]:

- **Improved cataloguing**

With the detailed understanding of the characteristics of the reusable components, the characterization plays a significant role in cataloguing the components in an improved manner.

- **Improved usage**

Characterization of reusable component gives detailed specification of a component. This detailed specification facilitates the better usage of components. It also ensures that a right component is being used in developing the application.

- **Improved retrieval**

Characterizing a component provide for identification, selection and adoption of components. It reduces the time and consequently cost to acquire a component.

- **Improved understanding**

The characterization of a reusable component helps to understand a component in a better way. When all the specifications of a component are present then it becomes easier to identify the required components and to implement the selected components according to the requirements that suites the project under development.

6. CONCLUSION

The study elicits the definition of reusable components. It figures out to what a component reuse accounts for and to what does a component reuse does not accounts for. It also specified the factors that asked for component characterization. That is, the factors that specify the situation in which the requirement of component characterization described. The various characteristics of reusable components are categorized depending upon the specifications of the components. The specifications are divided in the basis of external and internal characteristics of components. The external characteristics refer to the external features of components such as age, context, source of component etc. While the internal characteristics refer to features such as technology, functionality, portability, interoperability, access to source code, architectural aspect, component type etc. The study also comes – up with certain points that show how the component characterization influences the reuse of reusable software components.

REFERENCES

[1] Bakshi A. et.al, "Development of a Software Repository for the Precise Search and Exact Retrieval of the Components", International Journal of Advanced Research in Computer Science and Software Engineering, 3 (8), ISSN: 2277 – 128X, pp. 1116 – 1126, 2013.

[2] Frakes W. B. et. al, "Software Reuse Research: Status and Future", IEEE Software, 31 (7), pp. 529 – 536, 2005.

[3] Garlan D. et. al., "Architecture Mismatch or Why it's Hard to Build Systems out of Existing Parts", Proc. 17th International Conference on Software Engineering, IEEE Computer Society Press, Los Alamitos, ca. pp. 179 – 185, 1995.

[4] Gill N. S., "Importance of Software Component Characterization for better Software Reusability", ACM SIGSOFT Software Engineering Notes, 31 (1), ISSN: 0163 – 5948, pp. 1 – 3, 2006.

[5] Gill N. S., *Software Engineering*, New Delhi: Khanna Book Publishing Company, ISBN: 978 – 81906116 – 3 – 3.

[6] http://www.nvc.cs.vt.edu/ICSRworkshop-DReMeR-13/uploads/1/3/9/9/13998478/characterizingreusable Components-Alkazemi.pdf.

[7] https://earthdata.nasa.gov/esdswg/software-reuse-srwg/reuse-definitions.

[8] Iglesias A. et. al, "Building System Requirements with Specification Components", Proc. of Joint Conference of Information and Computer Science, JICS'98, Vol. 3, pp. 499 – 509, 1998.

[9] Kroeker K., "Component Technology", IEEE Computer, Vol. 31, pp. 132 – 133, 1998.

[10] Mili H. et. al, "Reusing Software: Issues and Research Directions", IEEE Transactions on Software Engineering, 21 (6), pp. 528 – 562, 1995.

[11] Qureshi M. R. et. al, "The Artifacts of Component – Based Development", Science International – Lahore, 19 (3), 187 – 192, 2007.

[12] Sandhu P. S. et. al, "A New Characterization Scheme of Reusable Software Components", International Journal of Computer Science and Network Security, 7 (8), pp. 220 – 225, 2007.

[13] Sharma A. et. al, "Managing Component – Based Systems with Reusable Components", International Journal of Computer Science and Security, 1 (2), pp. 60 – 65, 2007.

[14] Subedha V. et. al, "Design of Conceptual Reference Framework for Reusable Software Components – Based on Context Level", International Journal of Computer Science Issues, 9 (1), ISSN (Online): 1694 – 0814, pp. 26 – 31, 2012.

[15] Yacoub S. et. al, "Characterizing a Software Component", In Proceedings of the 2nd Workshop on Component – Based Software Engineering, in conjunction with ICSE'99, 1999.