Indian Journal of Science and Technology, *Vol 9(32), DOI: 10.17485/ijst/2016/v9i32/100187, August 2016* *ISSN (Print) : 0974-6846* *ISSN (Online) : 0974-5645*

Towards The Adoption of Modern Software Development Approach: Component Based Software Engineering

Prateek Jain*

School of Computer Application, Lovely Professional University, Phagwara - 144411, Punjab, India; prateekjain2010@gmail.com

Abstract

Objectives: This paper discusses about the component based software engineering approach along with its classification of various approaches of CBSE and Component Based Development [CBD] and the life cycle development model of Component Based Software Development [CBSD] has been proposed by the component re-usability concept in lieu of reducing the software cost. **Method:** The main focus of the software engineering and component based software engineering is the development of the software in such a manner that it might be used again for the development of some other software. Component Based Software Engineering [CBSE] includes the use of components lying in the component repository while software re-engineering focusses relies with using the concept of forward and reverse engineering. This is done for reducing the software development cost by making use of Component Based Software Engineering [CBSE] principles and s/w re-engineering techniques. **Findings:** Now a-days the development of software from the scratch has become less prioritized and is being used very less. Component based software engineering plays a very vital role in developing of the software not from scratch but instead using the existing components for the development of new software. Hence it can be concluded or finded that the cost of the software development from the re-usability concept can be reduced to a large extent in comparison to the software development from the very scratch being a very costly measure. The proposed model will be helpful in reducing the software development cost by using the component based development strategy. **Applications / Improvements:** The proposed SDLC model for CBSD can be used for the real life applications as well and thereby can be re-used again and again as per the requirements of the industry.

Keywords: CBD, CBSD, CBSE,

1. Introduction

In today's scenario the software plays a multiple role in the same time as it is being considered as a vehicle for the delivery of a product and on the other hand it is also a product and as a product it is responsible for providing the potential of a computer and hardware devices or the networks or other measures or the network of computers being used for various scenarios. The concept of software engineering is based on the layers where it provides the technical aspects of how to do portion of the software build up. While the various methods comprise of a broad network of tasks including the requirement specification, analysis, designing, program construction, testing and finally the support or the maintenance. Finally, the software engineering relies on the set of very basic principles which are responsible for governing every aspect of the technology thereby including modelling activities and other various descriptive techniques. Software Engineering (SE) is the approach of a system, disciplination, quantification for the development, operation, testing and maintenance of software, and the study of these approaches that is, the application of engineering to software. The engineering of the software is also comprised of various principles and practices for developing the software from the very scratch. For this we need to follow the specific software development life cycle models such as Waterfall, Spiral, prototype etc as per the

** Author for correspondence*

different requirements. All these SDLC models helps in developing the software from the very scratch and it consist of Requirements Gathering, analysis of requirements, Design creation, Coding, Testing, implementation and lastly the maintenance activity where the maintenance cost is the higher[1].

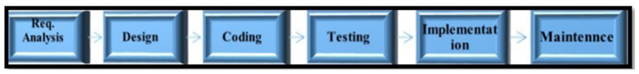


Figure 1. Traditional SDLC process.

As already being discussed above the strength of an industry is meant for solving the problems of the customers and the clients in common and their needs should be fulfilled by the software. Hence there is a problem for the development of a software as per the exact customer requirements. This problem has been shown in [2].

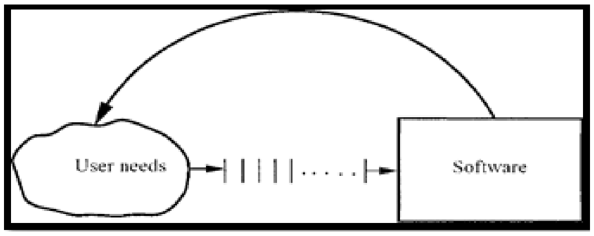


Figure 2. A fundamental problem.

But today is the era of modern technologies and we need to develop the software by reducing the cost of the software especially in maintenance. So we need a technique like component where the software can be developed using the re-usability principles. The organization of the paper is as follows: Section II contains the introduction about CBSE, Section III consist of proposed SDLC model for CBSD, Section IV consist of conclusion and future scope and finally the references are included at the end.

2. Component Based Software Engineering

Now a-days the software industry is having the biggest challenge of developing the software as in today era the software needs to be more reliable, its efficiency should be high, it should have low maintenance cost and it should ensure more effective software development. So in lieu of these challenges can be resolved using the concept of re-usability in order to overcome these challenges thereby having less cost of maintenance. Besides these challenges re-usability may also be helpful in saving the design, coding cost too. Re-usability can be applied for application system reuse, component re-use as well as the function re-use. Figure 3 shows the difference between the cost of maintenance and the savings done by re-use principles (in approximate figures)[3].

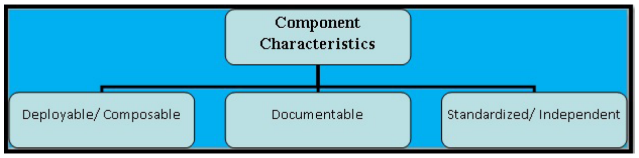


Figure 3. Cost of maintenance vs savings through re-usability.

Now the concept of CBSE emerged from the failure of object-oriented development to support effective reuse principles as in OOD the single object classes are too detailed and specific. Components are found to be more abstract that the objects and classes and hence it can be considered as a stand-alone service provider. The concept of CBSE usually involves a prototyping approach with the components being glued together by means of using a scripting language. In CBSE there is a specific thing called as a "Component - An independent executable entity that can be made up of one or more executable objects." There might be the variations in the sizes of the components starting from any simple application to the complex applications too. The component in the simple terms can be considered like an egg in the form of interface, code and data being the parts of the egg as shown in Figure 4.

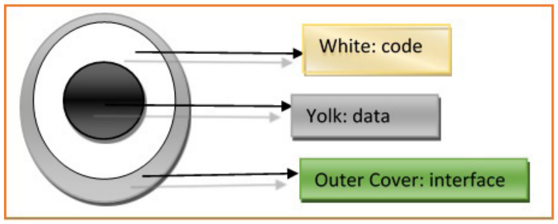
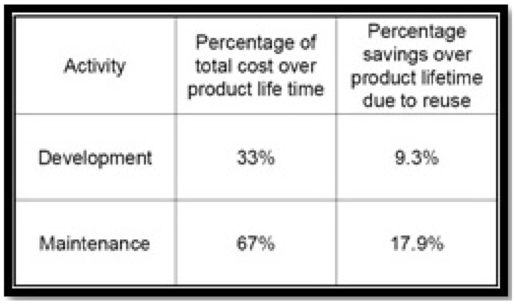


Figure 4. Component as an egg.

An example of component are the classes and the interfaces. Component should have characteristics as standardized, independent, composable, deployable as well as documentable. All of the mentioned characteristics are very crucial before choosing any component for using in development for re-use or the development with re-use. Figure 5 shows the 5 characteristics of a component.

- Deployable: For being deployed a component should be capable of operating as a stand-alone entity. It is being treated as a binary component and need not be compiled before the deployment.

- Composable: All the interactions that are external should be in picture through the publicly defined interfaces and it should also have been able to give its information regarding the components and interfaces to the external entities in order to be composed properly.
- Documentable: The component should also have the characteristics that it should be able to provide the documenting capabilities to the users and users should be able to make the documentation quite comfortable.
- Standardized: The standardization of a component means if it is used in CBSE process so it has to be a part of some standardized component model. This may define the interfaces, component, meta-data of component etc.
- Independent: A component is independent if it is being able to deploy or compose without having to depend on the other components in various situations.



Activity	Percentage of total cost over product life time	Percentage savings over product lifetime due to reuse
Development	33%	9.3%
Maintenance	67%	17.9%

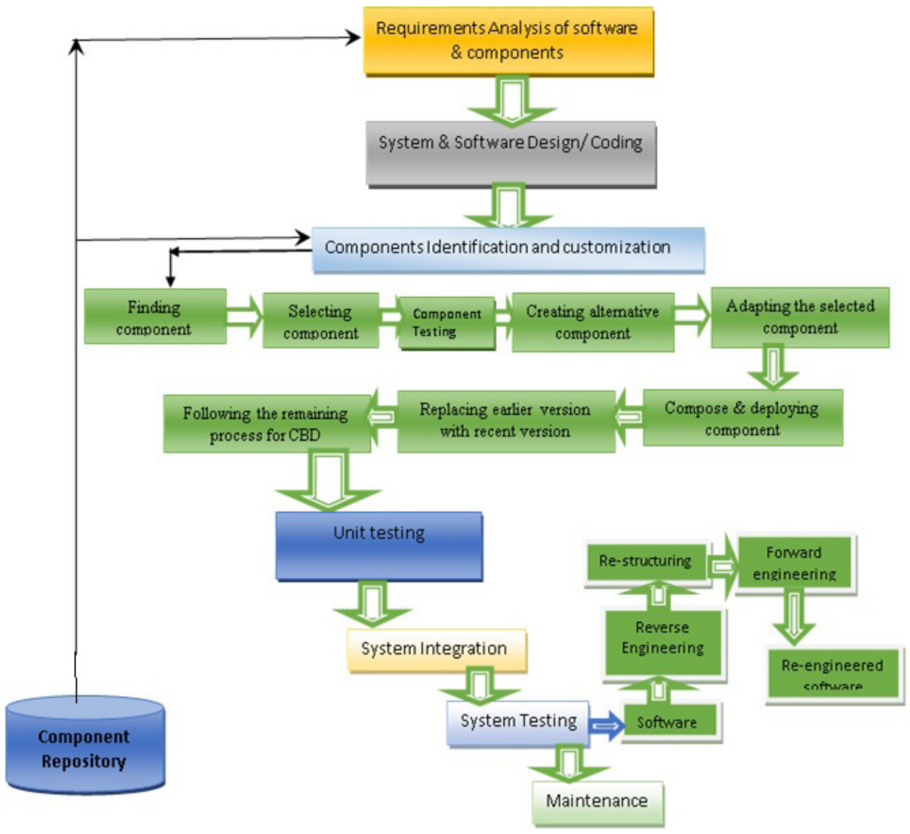
Figure 5. Component characteristics.

The software being developed using the component based approach needs to follow the component development approach named as CBD.

3. ProposedSDLCModeComponent Based Software Development

3.1 Requirements Analysis and Specification

This phase is the starting of the SDLC and consists of gathering the requirements from the clients for the development of the software and correspondingly the SRS



Requirements Analysis of software & components
System & Software Design/ Coding
Components Identification and customization
Finding component
Selecting component
Component Testing
Creating alternative component
Adapting the selected component
Compose & deploying component
Replacing earlier version with recent version
Following the remaining process for CBD
Unit testing
System Integration
System Testing
Software
Reverse Engineering
Re-structuring
Forward engineering
Re-engineered software
Maintenance
Component Repository

Figure 6. Proposed SDLC model for CBSD.

is being prepared. Next thing is that the components should also be checked for the requirements as we need to check whether any component from component repository can be re-used for the development of a new system or not. Hence the SRS along with the checking and analyzing component requirement is also done. This implies that the software engineers must be aware about which components will be used during the entire life cycle in order to develop the efficient software. Since it is not likely that appropriate components can always be found, there is a risk that the new components have to be implemented. During the component re-usability there might be a situation that the component with desired functional is not found so the components need to be modified to make it usable appropriately.

3.2 System and Software Design

As the requirements phase is concerned with the preparation of Software Requirement Specification document and in the similar terms system and software design specification is concerned with the availability of the components. The main components are complying with a particular model of the component. It is very easy for one to assume that it is possible to use the components which are in direct implementation in different component technologies.

3.3 Components Identification and customization

This is for the component to work correctly and efficiently with the whole system, it has to be adjusted in terms of specific system requirements, platform, performance and interfaces. This process is called customization of a component and the resulting product should be ready for integration.

Although the CBSE approach is facing some challenges like un-availability trusted components, certification of component, inappropriate development lifecycle models, configuration techniques of component and the supportability of various tools required for implementing the component. Instead of having these issues till then this approach might be used in the development of software by applying the concept of re-usability components and therefore the cost for the software can be reduced to a great extent.

3.3.1 Finding Component

The first step in the CBSD process is finding the appropriate and the right components according to the requirements of the software to be build on.

3.3.2 Selecting Component

As soon as the component is find out in the first phase so the next step lies in the selection of appropriate component to be used during the software life cycle. The component selection phase is a very crucial phase during this process.

3.3.3 Testing Component

This step comprises of testing a component for its existing functionality and checking whether it is suitable for re-usability or not and if the testing goes fine so we can go with the next phase else we need to reject this component or modify this component in order make it applicable for re-usability.

3.3.4 Creating Alternative Component

If the components are not being selected or not found out in the previous steps so in that case an alternative component needs to be created with the proprietary approach. Proprietary approach signifies that the component will be owned by a specific components and the users need to use this proprietary component as per their requirements.

3.3.5 Adapting Selected Component

After the component is selected or the proprietary component is prepared so we need to adapt that selected component.

3.3.6 Compose and Deploying Component

After adapting the component in the previous step we need to compose the selected component and needs to be combined with the other modules of software and finally it needs to be deployed properly for the usage in component based software development phase.

3.3.7 Replacing Earlier Version with Recent Version of Components

In case we got the updates of the existing version of the component so we can replace that component from existing to the recent version of the component.

Following the remaining process for CBD

This phase comprises of the activities that need to be performed after this phase that includes the unit testing, system integration, system testing and maintenance. In this we can go with either maintenance phase or we can choose the re-engineering approach for modifying the existing system for the new system.

3.4 Implementation and Unit Testing

As fast as the components are being selected from the earlier phases so the next phase is of performing the unit testing for the components and their related modules for checking their current functionality according to the requirements the functionality should be modified or altered. In an ideal case every component themselves are properly built and tested. However, the component tests in isolation are not sufficient. Often design units will be implemented as assemblies of several components and possibly a glue code. These assemblies must be tested separately, since an assembly of correct components may be incorrect although the components themselves are correct[4,5].

3.5 System Integration

The system integration integrating the standard infrastructure components that builds up the component framework as well as the application component framework. So in this the specific component needs to be integrated with entire system which is called as component deployment. Besides the component deployment the rest of the modules are also integrated together in order to get a complete system which should be ready for next phase called as system testing[6].

3.6 System Testing

System testing comprises of testing the whole system at a once in order to check whether the integrated modules are working properly or not. In case the system testing is not ok so in that case the modification in source code needs to be done and if the system testing is ok so we proceed with the further phases consisting of either the software maintenance or the re-engineering of the existing software.

The software maintenance is a very costly process as entire software needs to be modified from the very beginning hence it should be avoided and alternatively we can use the software re-engineering process for the alteration of the existing software as it comprises very lesser cost. It consists of 3 phases i.e.: a) Reverse Engineering b) Re-structuring c) Forward Engineering and finally by following these phases we will get the re-engineered software[7–9].

On a whole CBSE is:

CBSE = COA + COP + COD + COM

COA: Component Oriented Analysis, COP: Component Oriented Programming, COD: Component Oriented Development, COM: Component Oriented Management.

4. Conclusion and Future Scope

It has been observed and concluded that developing the software from the scratch is a very costly process in terms of more cost of money, resources as well as time. But these cost can be reduced to an extent by using the concept of re-usability and thereby using he modern approach called as component based software engineering instead of traditional software engineering approach. CBSE focusses on using the existing components that are being designed in .NET or sometime in Java programming languages. These components are then restructured as per the requirements of a new software and these components can be re-used by the help of component repository that consist of all the components for the re-usability. Hence we can say the future scope for the CBSE is very vast as this approach can be used for distributed as well as the cloud computing environment. So the proposed SDLC model based on CBSD approach will be useful for developing the software by using an existing component rather than doing it from the scratch every time.

5. References

1. Allen R, Douence R, Garlan D. Specifying and analyzing dynamic software architectures. Fundamental Approaches to Software Engineering, Springer Berlin Heidelberg, Indian; 1998 Mar 28. p. 21–37.
2. Garlan D. Pervasive computing and the future of CSCW systems. Proceedings of the Workshop on Architectures for Cooperative; 2000 Dec.
3. Satyanarayanan M. Pervasive computing: Vision and challenges. Personal Communications. 2001 Aug; 8(4):10–7.
4. Crnkovic I, Larsson MP. Building reliable component-based software systems. Artech House; 2002.
5. Schmerl BR, Marlin CD. Versioning and consistency for dynamically composed configurations. Springer Berlin Heidelberg; 1997 May 18.
6. Wang Z, Garlan D. Task-driven computing, Carnegie-mellon University Pittsburgh PA School of Computer Science; 2000 May.
7. Hutchens DH, Basili VR. System structure analysis: Clustering with data bindings. Software Engineering. 1985 Aug; SE-11(8):749–57.
8. Haghpanah N, Moaven S, Habibi J, Kargar M, Yeganeh SH. Approximation algorithms for software component selection problem. 14th Asia-Pacific, Software Engineering Conference, APSEC, IEEE; 2007, Dec 4. p. 159–66.
9. Vescan AN. Dependencies in the component selection problem. Proceedings of the 6th ICAM-International Conference on Applied Mathematics; 2008.