ORIENTAL JOURNAL OF
COMPUTER SCIENCE & TECHNOLOGY
An International Open Free Access, Peer Reviewed Research Journal
Published By: **Oriental Scientific Publishing Co., India.**
www.computerscijournal.org

ISSN: 0974-6471
June 2017,
Vol. 10, No. (2):
Pgs. 467-473

Component Based Software Development Life Cycle Models: A Comparative Review

PREETI GULIA and PALAK*

[1,2] Department of Computer Science and Applications, M.D. University, Haryana, India
*Corresponding author E-mail:palak.aug6@gmail.com

http://dx.doi.org/10.13005/ojcst/10.02.30

(Received: May 09, 2017; Accepted: June 01, 2017)

ABSTRACT

The development of high quality software is the need of current technology driven world. Component Based Software Engineering (CBSE) has provided a cost effective, fast and modular approach for developing complex software. CBSE is mainly based on the concept of reusability. Apart from these CBSE has several advantages as well as challenges which are summarized in this paper. Large and complex software development requires management of reusable components and can be selected from component repository and assembled to obtain a working application. Development of components and their assembly is different from traditional softwares which leads to the need of new development paradigms for Component Based Systems (CBS). Software development life cycle (SDLC) provides planned and systematic arrangement of activities to be carried out to deliver high quality products within time and budget. This paper presents a comparative study of component based software development life cycle models with their strengths and weaknesses.

Keywords: Component; CBSE; SDLC, Software Engineering, Development Lifecycle.

INTRODUCTION

Software development process has evolved a long way from traditional waterfall model to highly manageable component oriented software. Initially softwares were developed from scratch using functional (procedural) approach. It was a top-down approach which breaks functional requirements into sub functions and building a program for functionality. Later a huge change in software industry came with the introduction of object oriented paradigms in early 1990s. Object Oriented Programming System (OOPS) provides a great control over data. Various features like abstraction, encapsulation, polymorphism, inheritance etc. were revolutionary advantages for software industry. They offer a bottom up approach for software development where the main focus was on data and entities rather than on functions. But some problems which are not addressed in OOPS were solved in AOSP (Aspect Oriented Software Programming) such as cross cutting concerns. Later Component Based Software Engineering (CBSE) evolved which focuses on reusability of

the previous effort done to build components. Each component represents a set of services which can be assembled with other components. Thus collection of such interactive components builds the whole software. Later we can add, replace or modify components according to our needs. This helps in reducing software crisis and delivers robust software products with faster delivery and reduced cost. Fig. 1 shows the evolution of SDLC that depicts how practitioners have switched to software modularity and software development is becoming more cohesive and scalable.

Component Based Software Engineering (CBSE) has gained popularity in last few decades because of increasing demand of complex and up to date software. It has provided a cost effective, fast and modular approach for developing complex software with reduced delivery time. Actively reusing designs or code allows taking advantage of the investment made on reusable components.

A component is defined by many researchers in many ways. According to Szyperski- "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 party"[1].

So we can say that a component is a black box, reusable software implementation that can be executed and interacts using well defined interfaces. Components are built to be reusable which makes development of further applications with similar functionalities much easier. Components are heterogeneous in nature in terms of programming languages and platform. Component based architecture provides flexibility to update or modify components and choose the best in class[2].

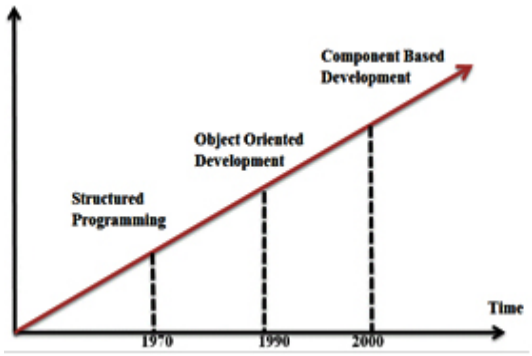

Fig. 1: Evolution of CBSE

Organization of rest of the paper is as follows: Section II summarizes advantages of using CBSE for complex software development. Apart from advantages, CBSE introduces new challenges for the developers which are given in Section III in this paper. In Section IV and V, CBSE development life cycle models are given summarized.

Advantages of CBSE

With the evolution of CBSE, it is easy to manage and update large and complex software. While object oriented paradigms are not sufficient for present day software, CBSE provides a promising solution with following advantages as shown in Fig. 2:

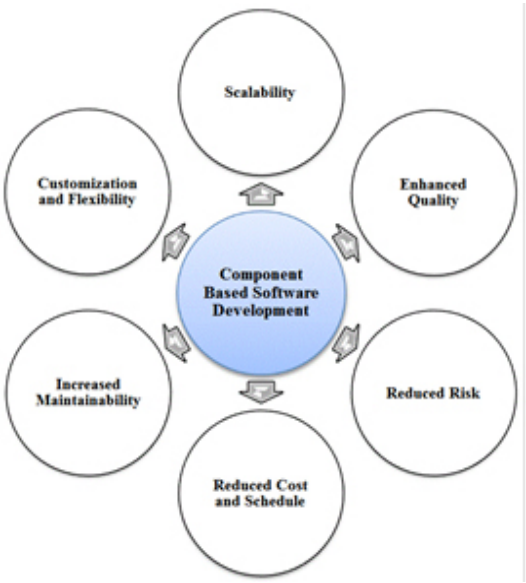

Fig. 2: Advantages of CBSE

Scalability

omponents can be easily added, removed or updated. CBSE systems are highly scalable as more and more components with new functionality can be added easily.

Enhanced Quality

As certification process is already completed for the developed components so the

final software is readily predicted to be of good quality. Several models have been proposed to evaluate quality in such systems[3,4].

Reduced Cost & Schedule

As the components are reused, the cost and the time needed to develop new components are saved. The cost of component development is recovered after five successive reuses.

Customization and Flexibility

CBSE provides a set of adaptable components with predefined architecture. Application developers can purchase them from third party and customize and assemble them according to their specific requirements.

Increased Maintainability

CBS are more maintainable as it is easy to replace faulty components with their alternatives[5].

Reduced Risk

Risk of software failure is reduced because of availability of various alternatives for a component with similar functionality. The advantages from CBSE are not limited to those described in above section. Several other inherent advantages are there for the developer as well as the user which makes CBSE a right choice for the future software products.

CBSE Challenges

Apart from advantages, CBSE introduces new challenges for the developers. Component based software products are completely dependent on efficient reusability and interaction between the heterogeneous components. Although the field of CBSE is heavily researched over last two decades but still there are some challenges[6,7] which the practitioners have to face. Some of them are listed below:

Heterogeneity of Components

Components are heterogeneous in terms of programming language, platform, data structure, naming conventions etc. They are developed by third party at different platforms under different project plans. Any internal error in the component can lead to its failure leading to overall system failure and cannot be easily corrected due to the heterogeneous nature of components.

Optimal Component Selection

A promising and optimal set of components should be selected from component pool after system analysis and requirement engineering. Component selection is NP (Non – Deterministic Polynomial) hard problem which requires a lot of effort and soft computing based approaches. This has been an attractive research area for many researchers[8,9].

Expensive and Inadequate Component Testing

Component testing is an expansive process as it involves construction of test drivers and stubs. Building and configuring separate stub and drivers for each component is a cumbersome task for testers. All possible combinations of available components are very difficult to test.

Complex Interface Specification

Each component has some interfaces through which it interacts with other components. Interface specifications are the entry points for defects. They need to be tested thoroughly. But the problem arises when these specifications are complex.

Continuous Versioning

Co-existence of different versions of a component with slight modifications creates challenge for testers to test all possible versions.

Improper Working due to Application Level Changes

Changes on the application or system level may affect overall working of CBS as the lifecycle of components and the application in which they are used are separate. There is a risk that this change introduced will cause system failure[6].

Component Configuration and Certification

A component must have standard configuration and must have undergone well established certification policy. This develops faith in stakeholders of CBS. But there is a lack of procedures and standards for the same[6].

Component Based Software Development Life Cycle Models

As stated earlier, CBSE is an approach of developing complex software applications by assembling reusable components from a variety of sources into a well-defined architecture. Traditional waterfall and iterative life cycle development models are not sufficient for CBS. So many researchers proposed various Component Based Software Development Life Cycle models over years. This section briefly summarizes some of these models.

The Y Model

Y model was proposed in 2005 by Luiz Fernando Capretz[10]. Considering the concept of reusability, Y model separate the development of components. This model allows iteration and overlapping of stages. The model resembles the letter Y in English from where the name Y model came in existence. The model has three branches showing the main phases of development. Various phases are shown in Fig. 3. At intersection of three branches is the assembly phase. Assembly of reusable components can be done after domain engineering and frame working where reusable components and their interrelationships are identified in terms of application vocabulary. Parallel to domain engineering, system analysis and design phases are carried out. The results of system analysis and designing phase are useful for adapting the selected components according to the system design requirements. Next step is to assemble and implement the system that is composed of various reusable components glued together in a framework. Component testing and system testing is also an important phase to assure quality of final product.

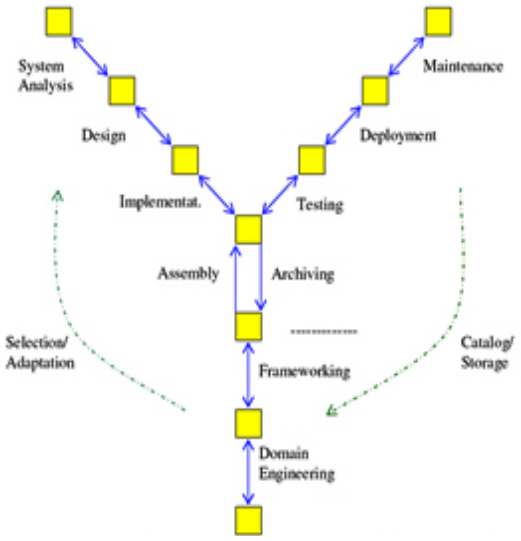

Fig. 3: Y Model [10]

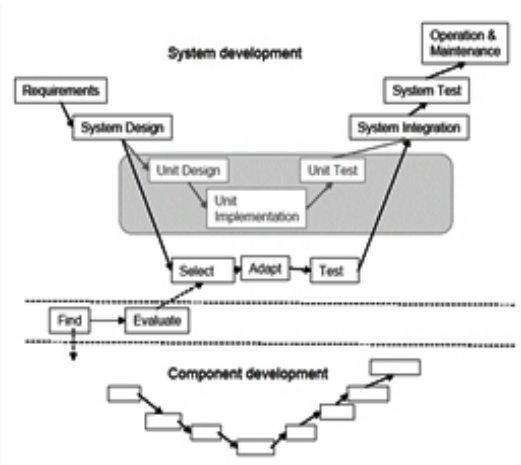

Fig. 4: V Model for CBD [11]

The V Model

Ivica Crnkovic (2005) et. al. in[11] distinguished evolutionary development models from that of traditional sequential models. They proposed a modified V model considering two aspects i.e. "component development" and "developing system form components". They identified that component based development process is different from non-component based development and found the new problems that arise due to component selection and assembly. Fig. 4 shows the modified V model in which the unit design phase is replaced by component selection phase. Component development phase is independent of developing system from components and can be carried out in parallel. The developed reusable components are stored in component pool from where they can be selected after analysis and design phase is over.

The X Model

Fig. 4 shows X model that was proposed in 2008[12,13]. The model contains four arms arranged in the shape of letter X of English alphabet and each arm represents different perspective of development.

At the intersection of these four arms is the Testable Component Repository (TCR). The upper left arm specifies the stages for development of components for reuse. Here the components are developed from scratch and stored in component repository. The lower left and lower right arms specify stages for development after modification and development without modifications respectively. Both arms select components from component repository and finally assemble them for Component based software development that is represented by upper right arm of the X model. X model provides scope for system development with component modification as well as without component modification with is shown in lower left and lower right arm of X model in Fig. 5.

CBSD Dual Life Cycle Model

Jason et. al. in[14] proposed a dual life cycle model for CBD. This model coarsely divides the whole process into two parts i.e. component development and system development. The individual phases of each part are shown in Fig. 6. They also provided tenets of design science followed in successive phases of development. The component development is often carried out by third party commercial developers and they are developed in such a way that their architecture is well defined in terms of required inputs and outputs for proper functionality with other components. Component fabrication deals with testing the component in external environment to check its reusability. System development is divided into sub-phases namely: Requirement analysis, system and sub-system architecture, Component selection, cataloging and retrieval and finally assembly of components in a defined architecture is carried out. Component selection phase of system development is directly linked to component development part.

Fig. 5: X Model [12]

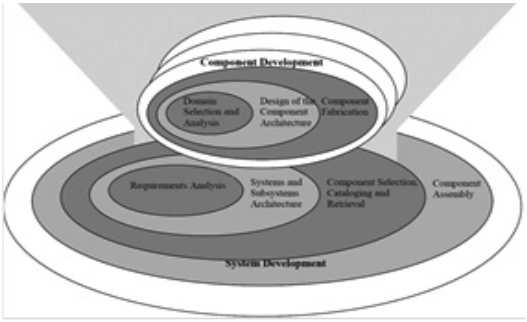

Fig. 6: CBSD Dual Life Cycle Model [14]

Table 1: Various Cbs Development Models And Their Focus Area

CBD Model	Year	Focus Area
Y Model [10]	2005	Reusability and Parallel Development
V Model [11]	2005	Component Development and Component Selection
X Model [12]	2008	Component Development, System Development With And Without Component Modification
CBSD Dual Life Cycle Model [14]	2009	Separation Between Component Development And System Development
Knot Model [15]	2011	Reusability, Modularity, Risk Analysis
W Model [16]	2011	Verification And Validation, Separate Component And System Development.

This model promises the separation between the two types of development but lacks the concern for component modification and overall system verification and validation.

The Knot Model

In 2011, Knot model[15] came into existence which was focused towards reusability, modularity and risk analysis. Fig. 7 shows the main phases and sub-phases of knot model. The main phases are: Development of New Component, Modification of Existing Component and Component Based Software Development. Out of these phases, modification of existing components is an iterative phase which deals with selecting the component from component pool, adapting and testing it according to system architecture and receiving feedback. The phase is repeated unless the selected component is fit for being assembled in the defined component framework. The efficiency of this model depends on the powerful implementation of Reusable Component Pool which is the pillar for all the phases. This model is easy to understand but selecting the appropriate component from reusable component pool is the most critical task on which the overall quality of the final product depends.

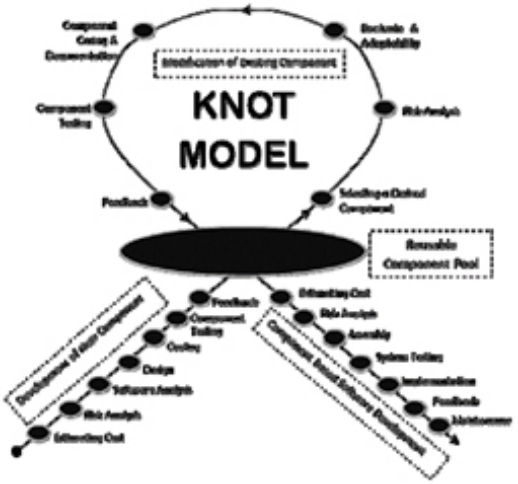

Fig. 7: Knot Model [15]

W Model

W model is a combination of two V models as shown in Fig. 8[16]. The left hand side represents component life cycle and the right hand side represents system lifecycle. It was proposed in 2011 and mainly focuses on Verification and Validation (V & V). The authors followed the standard CBD process and separated the life cycle of component development from system development. Component selection and adaptation step is the connecting link between two V models. V & V is considered at three levels in this model i.e. Component level, Compositional level and finally System level.

Summary of Various CBD Models

This section summarizes of all the models proposed for CBD on the basis of their focus area given in Table 1. This table clearly depicts that the aspect of reusability is of prime concern as depicted

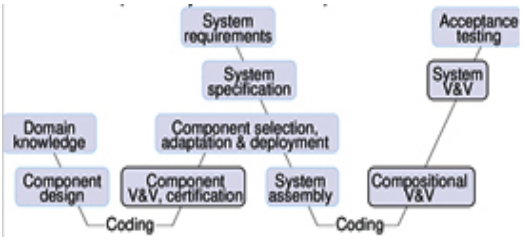

Fig. 8: W Model [16]

by Y and Knot Model. The second aspect that has got more attention is separation of development process of components from development of final system. Selection of components is also focused upon by many development models.

CONCLUSION

Complexity of software is increasing day by day in this era of technology. Fast and responsive systems rely on their underlying software which in turn relies on the robustness of development phases. So SDLC is the pillar for system performance and efficiency. With the evolution of component based software development paradigm a number of SDLC models has been proposed over time. This paper summarizes some important CBS development models such as V Model, Y Model, X Model, Dual Model, Knot Model and W model. Their main focus areas are discussed. One common aspect that is concluded from this study is that each model separates the development of component from the development of system. Moreover, selection of appropriate components from component pool is still an open issue in all the models.

REFERENCES

1. Szyperski C., Component Software-Beyond Object -Oriented Programming, Addison-Wesley, 1998.
2. Crnkovic Ivica, and Magnus Larsson. "Component-Based Software Engineering-New Paradigm of Software Development." Invited talk and report, MIPRO, pp- 523-524, 2001.
3. Prasenjit Banerjee, Anirban Sarkar, "Quality Evaluation Framework for Component Based Software" In *Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies* (ICTCS '16). ACM, New York, NY, USA, Article 17, 6 pages. 2016. DOI: http://dx.doi.org/10.1145/2905055.2905223
4. Gaurav Kumar and Pradeep Kumar Bhatia, "Neuro-Fuzzy Model to Estimate & Optimize Quality and Performance of Component Based Software Engineering" SIGSOFT Software Eng. Notes 40, 2 (April 2015), pp. 1-6.
5. Tassio Vale, Ivica Crnkovic, Eduardo Santanade Almeida, Paulo Anselmo da Mota Silveira Neto, Yguaratã Cerqueira Cavalcantic, Silvio Romero de Lemos Meira, "Twenty-Eight Years of Component Based Software Engineering", The Journal of Systems and Software, 111, pp. 128–148, 2016.
6. Ivica Crnkovic, "Component-Based Software Engineering — New Challenges in Software Development", Journal of Computing and Information Technology - CIT 11, 3, pp. 151–161, 2003.
7. Deepti Negi, Yashwant Singh Chauhan, Priti Dimri, Aditya Harbola. "An Analytical Study of Component-Based Life Cycle Models: A Survey", In Proceedings of International Conference on Computational Intelligence and Communication Networks (CICN), 2015.
8. Vinay, M. K., Johri, P., "W-Shaped Framework for Component Selection and Product Development Process", World Applied Sciences Journal, 31(4), pp. 606-614, 2014.
9. Rakesh Garg, R. K. Sharma, Kapil Sharma, "Ranking and selection of commercial off-the-shelf using fuzzy distance based Approach", Decision Science Letters, pp. 201–210, 2016.
10. Luiz Fernando Capretz, "Y: A New Component-Based Software Life Cycle Model", Journal of Computer Science (1), pp:76-82, 2005.
11. Ivica Crnkovic, Stig Larsson, Michel Chaudron, "Component-based Development Process and Component Lifecycle", Journal of Computing and Information Technology - CIT 13, 2005, 4, 321-327.
12. Gill N. S., Tomar P, "X Model: A New Component-Based Model", MR International Journal of Engineering and Technology, Vol. 1, No. 1 & 2, 2008.
13. Nasib Singh Gill and Pradeep Tomar, "Modified Development Process of Component-Based Software Engineering", ACM SIGSOFT Software Engineering Notes, March 2010, Volume 35 Number 2.
14. Jason H. Sharp, Sherry D. Ryan, "Component-Based Software Development: Life Cycles and Design Science-Based Recommendations", Proceedings of the Conference on Information Systems Applied Research, Washington, DC, 2009.
15. Rajender Singh Chhillar, Parveen Kajla, "A New - Knot Model for Component Based Software Development", International Journal of Computer Science Issues, Vol. 8, Issue 3, No. 2, pp. 480-484, May 2011.
16. Kung-Kiu Lau, Faris M. Taweel and Cuong M. Tran, "The W Model for Component-based Software Development", 37th EUROMICRO Conference on Software Engineering and Advanced Applications, IEEE, pp.47 – 50, 2011.