International Journal on Emerging Technologies (Special Issue NCETST-2017) **8**(1): 441-445(2017)
(Published by Research Trend, Website: www.researchtrend.net*)*

ISSN No. (Print) : 0975-8364
ISSN No. (Online) : 2249-3255

Comparative Study of Different Models in Component Based Software Engineering

Mr. Sandeep Chopra[1], Dr. M.K. Sharma[2] and Dr. Lata Nautiyal[3]
[1]Research Scholar, Uttarakhand Technical University, Dehradun, (U.K.), INDIA
[2]Associate Professor, Amrapali Institute Haldwani, (U.K.), INDIA
[3]Assistant Professor, Graphic Era University, Dehradun, (U.K.), INDIA

ABSTRACT: "Software Engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software". Component-based software development advocates developing software systems by selecting reliable, reusable and robust software components and assembling them within appropriate software architecture. It is assumed that common components in a various software package application solely got to be written once and re-used instead of being re-written when a replacement application is developed. This paper tells the efficiency of various CBSE models which are useful for the software project.

Keyword: Component, CBSD, Risk analysis, Testing, Reliability, Certification

I. INTRODUCTION

A component is a software element that conforms to a component model and can be independently deployed and composed without modification according to a composition standard [1].

By promoting the use of software components that commercial vendors or in-house developers build, the component-based software development approach promises large-scale software reuse. Component-based software engineering offers an attractive alternative for building Web-based enterprise application systems.

A software component can be deployed independently and is subject to composition by third parties [2].

A component [3] is a coherent package of software implementation that:

(a) It can be independently developed and delivered.
(b) It has explicit and well-specified interfaces for the services it provides.
(c) It has explicit and well-specified interfaces for services it expects from others.
(d) It can be composed with other components, perhaps customizing some of their properties, without modifying the components themselves.

So the following details of a component characterize a component [4]:

(ii)A component can be implemented in any language.
(iii) The component interface is described either textually by means of an interface description language (IDL) or visually / interactively by appropriate tools.

Component is defined by Meyer [5] as:

"A component is a software element (modular unit) satisfying the following conditions:

1. It can be used by other software elements, its 'clients'.
2. It possesses an official usage description, which is sufficient for a client author to use it.
3. It is not tied to any fixed set of clients."

In software engineering, this would allow a software system to have as "components" assembly language instructions, sub-routines, procedures, tasks, modules, objects, classes, software packages, processes, sub-systems, etc.

The widely accepted goal of component-based development is to build and maintain software systems by using existing software components. It is understood that the components are required to be reusable components. They must interact with each other in system architecture. This goal of CBSE implies four orthogonal properties for a truly reusable component:

1. Contractually specified interfaces,
2. Fully explicit context dependencies,
3. Independent deployment,
4. Third party composition.

A Component Based Software Development (CBSD)

CBSD approach is based on the idea to develop software systems by selecting appropriate off-the-shelf components and then to assemble them with a well-defined software architecture. The purpose of CBSD is to develop large systems, incorporating previously developed or existing components, thus cutting down on development time and costs.

CBSE can also be used to reduce maintenance associated with the upgrading of large systems. It is assumed that common parts in a software application only need to be written once and reused rather than being rewritten every time a new application is developed.Component primarily based software package development approach relies on the thought to develop software package systems by choosing acceptable ready to wear components and so to assemble them with a well-defined package design.

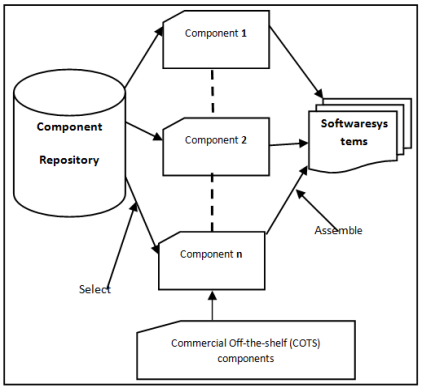


Fig. 1. Architecture of CBSE.

CBSE encompasses two parallel engineering activities, domain engineering and component-based development (CBD). Domain engineering explores the application domain with the specific intent of finding functional, behavioral, and data components that are candidates for reuse and places them in reuse libraries. CBD elicits requirements from the customer and selects an appropriate architectural style to meet the objectives of the system to be built.

This new software development approach is very different from the traditional approach in which software systems can only be implemented from scratch. These commercial off-the shelf (COTS) components can be developed by different developers using different languages and different platforms.

This can be shown in Figure 1, where COTS components can bechecked out from a component repository, and assembled into a target software system.

Component-based software development (CBSD) can significantly reduce development cost and time-to-market, and improve maintainability, reliability and overall quality of software systems [6] [7].

The CBSE generally embodies the following fundamental software development principles [8]:

B. Software Development Independently

Large software systems are necessarily assembled from components developed by different people.

To facilitate independent development, it is essential to decouple developers and users of components through abstract and implementation-neutral interface specifications of behaviour for components.

C. Reusability of Components

While some parts of a large system will necessarily be special-purpose software, it is essential to design and assemble pre-existing components (within or across domains) in developing new components.

D. Software Quality

A component or system needs to be shown to have desired behaviour, either through logical reasoning, tracing, and/or testing. The quality assurance approach must be modular to be scalable.

E. Maintainability

A software system should be understandable, and easy to evolve [9].

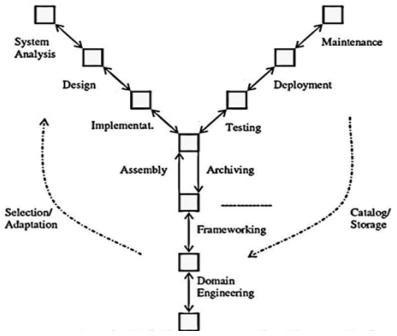
II. SURVEY OF CBSE MODELS

Various process models have been designed by a number of researchers so far for component based software development. Most common among them are studied and described briefly.

There are different CBSD models appear in industry as well as in academia. We referred to some of them; some of the popular state of art has been discussed in the following section:

A. The Y Model

Capretz [10] proposed a new life cycle model known as Y model for component-based development.



Y- Model

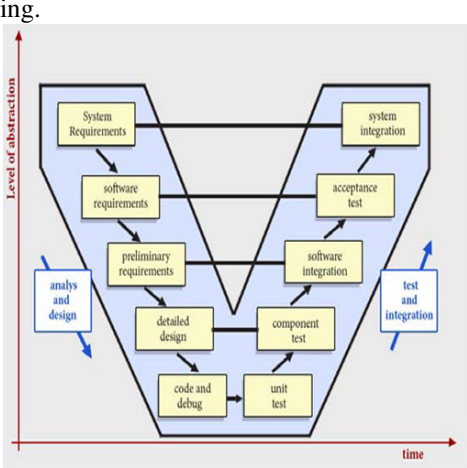
This model described software creation by change and instability therefore the "Y" CBSD life cycle model facilitates over lapping and iteration where appropriate.

This model consists of following planned phases; domain engineering, frame working, assembly, archiving, system analysis, design, implementation, testing, deployment and maintenance. In this model, the new phases were basically proposed for example domain engineering frame working, assembly and archiving with the other traditional life cycle phases stated for the previous models.

This model focuses on software reusability explicitly during CBSD and put more emphasis on reusability during software development, evolution and building of significantly reusable software components that will be built on the assumption to use them in future projects.

B. The V Model

The V-Shaped [11] life cycle is a sequential path of execution of processes. Each phase must be completed before the next phase begins. Testing is emphasized in this model more than the waterfall model. The testing procedures are developed early in the life cycle before any coding is done, during each of the phases preceding implementation. Requirements begin the life cycle model just like the waterfall model. Beforedevelopment is started, a system test plan is created. The test plan focuses on meeting the functionality specified in requirements gathering.Before development is started, a system test plan is created. The test plan focuses on meeting the functionality specified in requirements gathering.

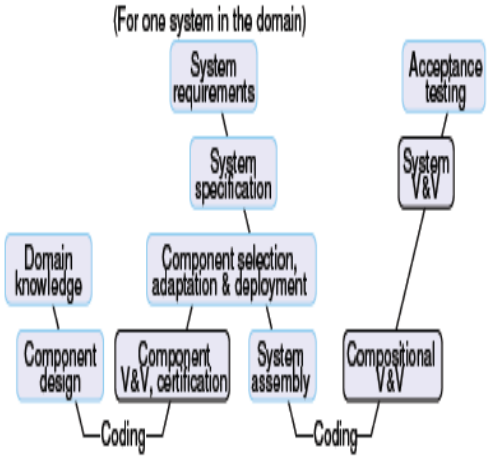


V- Model

III. THE W MODEL

Two V models have conjoined, one for component life cycle and one for system lifecycle in the W lifecycle model [12]. Component based development process comprises of a component life cycle and a system life cycle. It is the base of W lifecycle model.

However, in component based development process component life cycle is slightly different from others because it is a more complete one, namely the idealized one as it fulfills all the requirements of component based development.



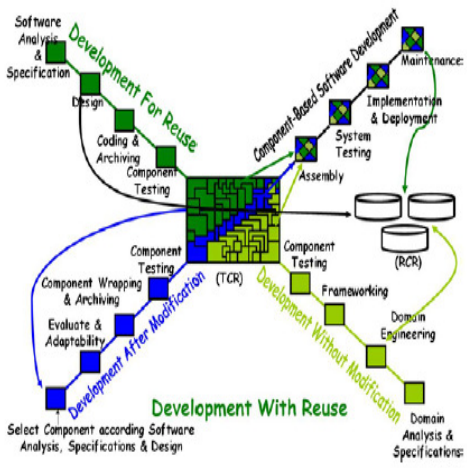
W Model

Component lifecycle comprises of two major phases: component design and component deployment, and is set in the context of a problem domain. In the design phase, software components are identified, designed and constructed according to the domain requirements or knowledge , and put into a software components repository.

The components that are contained by repository are domain-specific but not system-specific.

IV. THE X MODEL

Tomar and Gill [13] proposed the X Model in which the processes started in the usual way by requirement engineering and requirement specification. This software life cycle model mainly focuses on the reusability where software is built by building reusable components for software development and software development from reusable and testable components. In software development, there are two major approaches, build generic software components or develop software component for reuse and software development with or without modification in reusable component. It basically considers three different cases and one component based software development that normally occurs in component based software development



X- Model

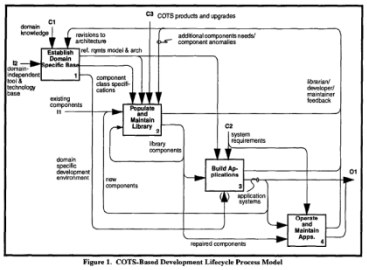
Development for reuse, development after modification, development without modification and component based software development.

It also separates the component development from component-based software development like other component based software development life cycles.

V. THE COTS BASED MODEL

One of the key factors in an exceedingly COTS-based development method is recognizing that there square measure many important changes to the normal system development life-cycle [14].

Some of these might have already been created in a company that has adopted associate degree an architecture-based or Product-line approach to reuse; others are going to be new most organizations. COTS is associate degree adaptation of a high-level method model developed originally for the DARPA Domain Specific Software Architectures Program.



COTS Model

VI. UMBRELLA MODEL

The life cycle of components based software consists of three stages: [15]

(i)The Design phase, when components are chosen by repository or designed, defined and created.

(ii) The Integration phase, when component are integrate with others component.

(iii) The Run-time phase, when component binaries are instantiated and executed in the running system.

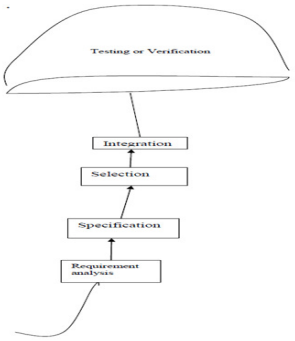
A software component life cycle model should define

(i) What sequence components composition are follows

(ii) Why need of these sequences

(iii) How to compose components.

Over the past three decades, many component based software development methodologies have appeared. Such methodologies indicates some or all phases of the software life cycle ranging from requirements to maintenance. These methodologies have often been developed in response to new concepts regardinga way to address the inherent complexity of software systems. Because of the increasing popularity of software reuse, in the last Fifteen years, research on component based software methodologies has become a growing field of interest.

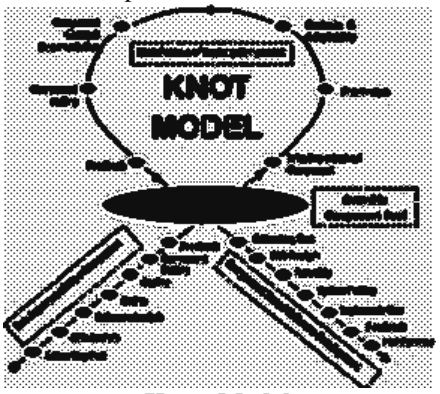


Umbrella- Model

VII. THE KNOT MODEL

Knot Model emphasis on reusability considering risk analysis and feedback in each and every phase. This model may be best suited for medium or larger complex system's development. It is based on three states of the component [16]

i) When a new component is not available in the repository, then it develops a new component for reuse and places it in the pool



Knot- Model

ii) When a component is partially available, then modifies it for reuse and places it in the pool.
iii) When a component is available in the repository then reuses it during the new proposed system. An utmost care should be taken that each component is created for reuse, so the component is not based on particular application's specification but must carry general specification.

In this model risk is resolved in early stages of each phase. This results in the reduction of cost and time and makes the software more reliable and efficient. Moreover feedback at the end of each phase results in further improvement and revised form of component. It also reduces the cost and time by better management as it resolves the conflicts, if any, during that phase.

Table 1: Comparative analysis CBSE models.

Key Factors	Y Model	V Model	W Model	X Model	COTS Model	Umbrella Model	KNOT Model
Component search	✗	✗	✓	✓	✓	✗	✗
Domain analysis	✓	✓	✓	✓	✓	✓	✓
Component selection	✓	✓	✓	✓	✓	✓	✓
Component evaluation	✓	✓	✓	✓	✓	✓	✓
Component selection procedure	✗	✗	✗	✗	✗	✗	✗
Risk analysis of a components	✗	✗	✗	✗	✗	✗	✓
Risk analysis of integrated	✗	✗	✗	✗	✗	✗	✗
Component certification	✗	✓	✗	✗	✗	✗	✗
Component testing	✓	✓	✓	✓	✓	✓	✓
Integrated CBSE software testing	✗	✗	✗	✗	✗	✗	✗
Reliability of components	✓	✓	✓	✓	✓	✓	✓
Reusability	✓	✗	✓	✓	✓	✗	✓

a) No provision of outsourcing of components.
b) Selecting a right component may be difficult.
c) Managing the reservoir may be difficult.

VIII. CONCLUSION

This paper has analyzed seven CBSE models. On the basis of key factors we have made comparative analysis in tabular form . The Key factor was based on the reasoning of the researcher . On the basis of above analyzed Table 1 we can say that Knot Model is the best model for the software project.

REFERENCES

[1] G.T. Heinemanand W.T. Councill, editors. Component-Based Software Engineering: Putting the Pieces Together. Addison-Wesley, 2001.

[2] Szyperski C., (1998). "Component Software, Beyond Object-Oriented Programming", ACM Press, Addison-Wesley, NJ.

[3] D'Souza D. F. and Wills A.C., (1997)." Objects, Components, And Frameworks with UML – the Catalysis Approach", Addison-Wesley, Reading, Mass.

[4] Kung-KiuLau, Mario Ornaghi and Zheng Wang, "A Software Component Model and its Preliminary Formalisation, Springer-Verlag Berlin Heidelberg, 2006

[5] B. Meyer. "The grand challenge of trusted components". In Proc. ICSE 2003, pages 660–667. IEEE, 2003.

[6] G. Pour, "Enterprise JavaBeans, JavaBeans & XML Expanding the Possibilities for Web-Based Enterprise Application Development," *Proceedings Technology of Object-Oriented Languages and Systems, TOOLS* **31**, pp.282-291 1999.

[7] G.Pour, M. Griss, J. Favaro, "Making the Transition to Component-Based Enterprise Software Development: Overcoming the Obstacles – Patterns for Success," Proceedings of Technology of Object-Oriented Languages and systems, 1999, pp.419 – 419.

[8] Iqbaldeep Kaur, Parvinder S. Sandhu, Hardeep Singh, and VandanaSaini, "*World Academy of Science, Engineering and Technology* **50**", 2009.

[9] M. Sitaraman and B. W. Weide , "Special Feature Component-Based Software Using RESOLVE", *ACM SIGSOFT Software Engineering Notes* **19**, No. 4, 21-67, October 1994.

[10] Luiz Fernando Capretz, "Y: A New Component-based software life cycle model", *Journal of Computer Science* **1** (1): 76-82, 2005, ISSN 1549-3636 © Science Publications, 2005.

[11] International Journal of Scientific & Engineering Research, Volume **3**, Issue 2, Februaryy-2012 1 ISSN 2229-5518.

[12] The W Model for Component-based Software Development [online]. OnlineAvailable:http://www.cs.man.ac.uk/~kung-kiu/pub/seaa11b.pdf.

[13] Tomar, P. Gill, N.S., "Verification & Validation of Components with New X Coponent-Based Model", in Proceedings of 2010, Software Technology and Engineering (ICSTE), 2*nd International Conference*, San Juan, PR, 3-5 Oct.

[14] Christine L. Braun, "A lifecycle process for the effective reuse of commercial off-the-shelf (COTS) software" in Proceedings of SSR '99 Proceedings of the 1999 symposium on Software reusability ACM New York, NY, USA, Pp. 29-36.

[15] Anurag Dixit and P.C. Saxena, "Umbrella: A New Component-Based Software Development Model" *International Conference on Computer Engineering and Applications 2009IPCSIT* vol.**2** (2011) (2011) IACSIT Press, Singapore.

[16] Rajender Singh Chhillar, ParveenKajla, "A New Knot Model for Component Based Software Development", *International Journal of Computer Science* Vol: **8** Issue: 3 Pp: 480-484, 2011.