Software Engineering 2012, 2(4): 138-146
DOI: 10.5923/j.se.20120204.07

An Improved Model for Component Based Software Development

Asif Irshad Khan[1,*], Noor-ul-Qayyum[2], Usman Ali Khan[2]

[1]Department of Computer Science, Singhania University, Jhunjhunu, Rajasthan, India
[2]Faculty of Computing and Information Technology, King Abdul Aziz University, Jeddah, Saudi Arabia

Abstract Software development costs, time-to-market and quality product are the three most important factors affecting the software industry. Various tools and techniques are invented by researchers and practitioners to improve in delivering quality software systems with lower cost and shorter time to market. One such practice is development of software using Component Based Software Development (CBSD) techniques. CBSD recommended building software systems using existing reusable components, instead of writing from scratch. The main objective of CBSD is to writes once and reuse any number of time with no or minor modification. Some of the advantages that a company may avail by adapting CBSD for the Software development are shorter development time which results in meet tight dead line, Increase productivity and Quality Product. CBSD also, support reusability. The aim of this paper is to describe the characteristics of some selected state of art CBSD models that are widely practiced in software industries. Based on the literature study we proposed a complete model for Component Based Software Development for reuse. This Model will cover both component based software development as well as Component development phases. Further a comparison is being made between the selected state of art CBSD models with our proposed CBSD model to know the strength and weakness.

Keywords Component Model, Component Based Software Development, CBD, Software Process

1. Introduction

Software Industry in the present Information Technology era, has enormous pressure of meeting the product deadlines with minimum development time and minimum development cost. Reusability of software is an important prerequisite for cost and time-optimized software development. More and more software companies are adopting Component-based Development (CBD) methodologies to meet the demands of customers to deliver the product with changing requirement and at lower cost.

Component-based software engineering (CBSE) is used to develop/assemble software from existing components. Some of the advantages that a company may avail by opting CBD for the SW development are reduce development time as less or no coding is involved, Achieve tight deadlines as development time reduced, software productivity is also improved as software are built by integrating already developed components instead of writing them from scratch by using state of art tools.

Further, risk in creating new software is reduced since the use of a component in several other similar domains increases the chance of errors being identified & fixed and strengthens confidence in that component.

CBD technologies comprised of implementing a component into a system through its well defined interfaces [1]. Using well-defined interfaces, a component interact with other components to accomplish a partial function of the system.

The inner structure of the component and the implementation of the interfaces are hidden to the outside. Therefore, CBSE enables a distributed and independent development of components as well as a straightforward replacement of a component by a different component in large-scale systems[2].

Lego, as shown in figure 1, is often taken as an example of a component- based approach.

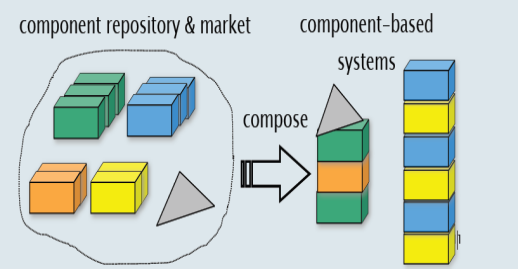

Figure 1. Concept of Component-based software engineering

Lego provides a set of building blocks in a large variety of shapes and colors. Lego is sold in boxes that contain a

* Corresponding author:
alig.asif@gmail.com (Asif Irshad Khan)
Published online at http://journal.sapub.org/se

number of blocks that can be composed to make up toys such as cars, trains and airplanes[3]. System development with components mainly focuses on what are entities that can be easily reusable and relations between them, starting from the system requirements and from the availability of components already existing. Components are built to be used and reused with little or no modification in many applications and should be well specified, easy to understand, easy to adapt, easy to deliver and deploy and easy to replace with newer versions release of the surrounding systems and applications.

In CBSD, Development of components is distinguished from development of systems using components, Component development focused on building reusable units for example Car Company Toyota build component like engine, brakes, etc which can easily be incorporated to a Toyota car while Component based System development focuses on the reuse of components, their evaluation and integration for example Toyota Corolla car is made by integrating different components like engine, brake, tires etc. These two processes are often performed independently of each other. The paper is organized as: section 2 describes Component based software development methodology, section 3 outlines some selected state of art CBD models, section 4 presents our proposed Improved Component based software development model section 5 describes comparison among different model and finally, section 6 describes conclusion and future work.

2. Component based Software Development life-cycle Phases

Component-based software development is a collection of process that deals with the systematic reuse of existing components often know as commercial off-the-shelf (COTS) and assembling them together to develop an application rather than building and coding overall application from scratch, thus the life cycle of component-based software systems is different from that of the traditional software systems. In general, analysis and design phases for component-based software development process models take more time than traditional ones and much less time is spent in development, while testing occurs throughout the process[4].

The life cycle of component based software systems can be summarized as follows:

2.1. Requirements Analysis and Specification

System boundaries should be defined and clearly specified in this activity, in a component-based approach requirement analysis also implies that it is necessary to analyze whether requirements can be fulfilled with available components.

Availability of existing components is also considered in requirements specification. Analysts have to be aware of the components that can possibly be reused. It is very likely that appropriate components fulfils all the similar requirement, in such cases one possibility is to negotiate the requirements and modify them to be able to use the existing components to keep with component-based approach and utilize its advantages. The main motive of this phase is to make the solution as crystal clear as possible.

2.2. Analysis and Designing of the System Architecture

The design phase starts with an architectural design of the system followed by more detailed design of the system. During this phase a decision has to be made about the possible use of the component model as this decision plays a crucial role in the architecture design of the system as well as the quality of the system.

2.3. Implementation

Finding candidate component as per requirement is the main focus of this phase. Theoretically no coding is required during this phase but practically coding is usually required as all functionality is rarely found in a component and some functionality need to be coded which is not provided by component.

2.4. Integration

This is a very important and critical phase of CBSE as at this point components integration (mismatch) most likely occurs because of lack of communication, capability issue, if the component is not made for the architecture. It is like if we want to build a car by integrating different car parts. System quality and functionality need to be validated and verified in this phase. To know the effectiveness of the assembled components a metrics is usually developed.

2.5. Testing

Since components is developed by third party the need for component verification is apparent since the system developers typically have no control over component quality or component functions[15].

2.6. Release and Maintenance

Release phase is similar to any other software released of a traditionally software development methodology. Released is made in such a way that it is suitable for delivery and installation. While maintenance phase is usually involved replacement of old / obsolete component with the new component to the system. Testing and verification have to be done to check the proper integration of component into the system.

3. Literature Review

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:

3.1. The V Model

The V model[5] adopted the traditional software development approach for building a system from reusable software components. The development activities have been shown in figure 1. It consists of several steps and provides the details information at the design phases. The main emphasis of V-Development is component development lifecycle.

Component development lifecycle was considered as different process. The selection phase gets input from the separate system that usually finds and evaluates the suitable components to be composed into the system.

The V Model is an adaptation of the rigid traditional waterfall model for modular system development with little flexibility. 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.

Before development is started, a system test plan is created. The test plan focuses on meeting the functionality specified in requirements gathering.

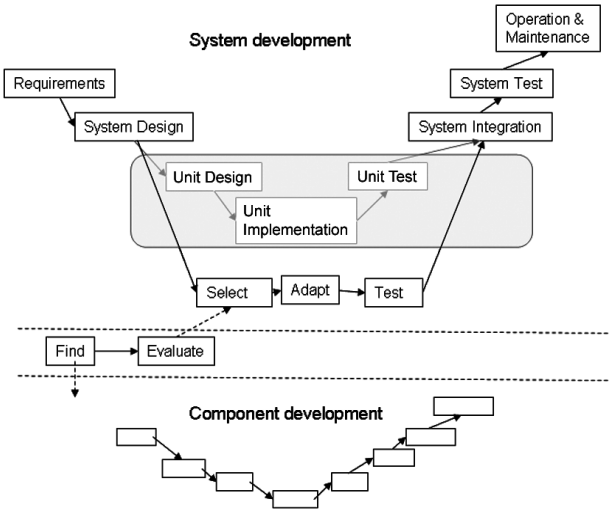

Figure 2. V development process for CBD[5]

3.2. The Y Model

Capretz[6][7] proposed a new life cycle model known as Y model for component-based development. This model described software creation by change and instability therefore the "Y" CBSD life cycle model as shown in Figure 2 and Figure 3 facilitates over lapping and iteration where appropriate.

This model consists of following planned phases; domain engineering, frameworking, assembly, archiving, system analysis, design, implementation, testing, deployment and maintenance.

In this model, the new phases were basically proposed for example domain engineering frameworking, 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.

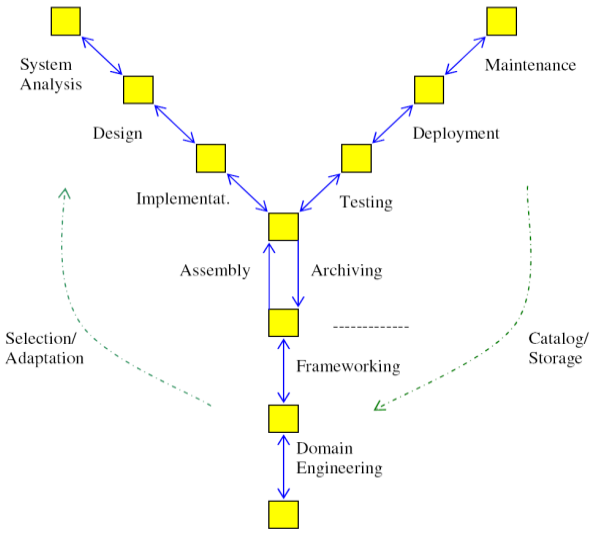

Figure 3. Y Model for CBSD[6]

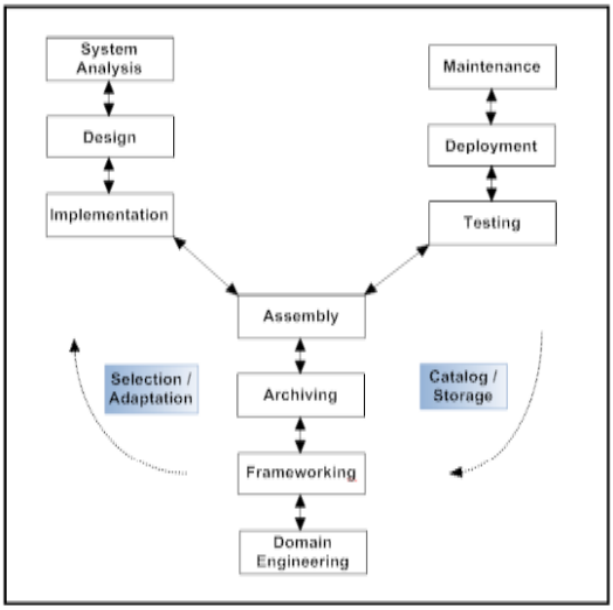

Figure 4. The Y. Model[8]

3.3. The W Model

Two V models have conjoined, one for component life cycle and one for system lifecycle in the W lifecycle model[9]. Component based development process comprises of a component life cycle and a system life cycle and is shown in figure 4. It is the base of W lifecycle model[9]. 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[10] as it fulfills all the requirements of component based development.

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[11], and put into a software components repository. The components that are contained by repository are domain-specific but not system-specific.

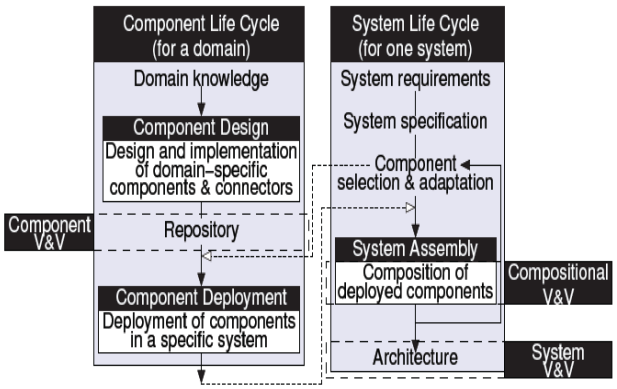

Figure 5. X-MAN CBSD Process with V&V[9]

The X-MAN component based development process with V&V in Figure 4 can be adapted directly as a process with two combined V Models, one for the component life cycle and one for the system lifecycle. These two V Models are combined via the step of component selection, adaptation, and deployment. This 'double V' process is shown in Figure 5. This model is known as W Model.

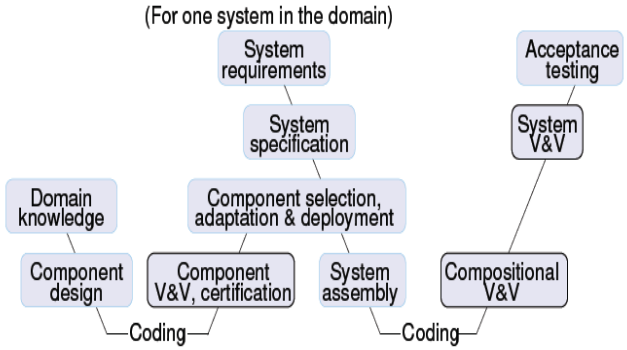

Figure 6. The W Model[9]

V&V activities in the W Model have been highlighted by boxes with black borders. In the context of component based development, the W Model is same as the standard component based development processes that have been discussed in the literature, in those same concepts have been used that is separate life cycles for components and systems.

However, unlike these processes, this component life cycle is the idealized one, which fulfills all the component based development. The W Model accommodates a V model for both component and system life cycles.

3.4. The X Model

Tomar and Gill[12] proposed the X Model in which the processes started in the usual way by requirement engineering and requirement specification as shown in the Figure 7[12]. 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.

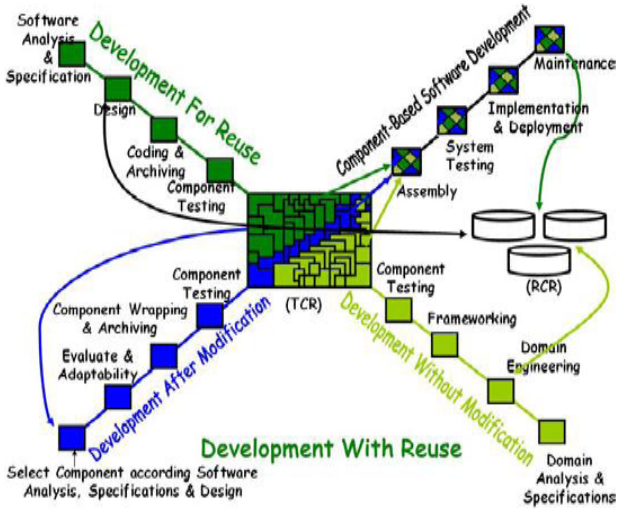

Figure 7. The X Model[12]

The X model lifecycle consists of four sub cycles for component based software development as shown in Figure 5. It basically considers three different cases and one component based software development that normally occurs in component based software development: 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. These cases are described in the sections to follow.

3.5. Elite Life Cycle Model (ELCM)

Lata Nautiyal, Umesh and Sushil[13] proposed Elite Life Cycle Model (ELCM) for the development of new product using component based technology as a viable alternative to address software reusability during component-based software production as shown in figure 8.

ELCM mainly focuses on the reusability during software development, evolution and the production of potentially reusable components that are meant to be useful in future software projects.

Reusability implies the use of composition techniques during software development; this is achieved by initially selecting reusable components and assembling them, or by

adapting the software to a point where it is possible to pick out components from a reusable library.

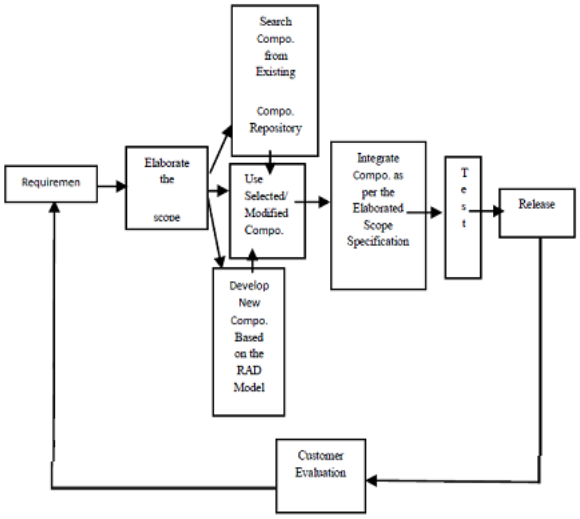

Figure 8. Elite Life Cycle Model (ELCM)[13]

4. Proposed CBSD Model

Some of the popular State of art has been discussed in our literature review section. From the literature review we came to the conclusion that all CBSD lifecycle have some drawbacks and there is a need of a new lifecycle for component based software development. Figure 10 shows details of our proposed improved CBSD Model. Reusing of existing artifacts is the most important concern of the Component Based Software Development. These reusable artifacts can be previously done system requirement, architecture, components and case study. The main phases of our improved CBSD model are 'Project Feasibility Study, System Requirement and Analysis', 'System Design', 'Component Identification and Adaption', 'Component Integration Engineering', 'System Testing and Acceptance' and 'System Release and Deployment' as shown in figure 9.The details about the proposed model are as follows:

4.1. Component Based Software Development Lifecycle

4.1.1. System Requirement, Specification and Decomposition

First and foremost step in developing an application for a client or stakeholders is to study the system requirements by a team of software analyst to elicit the requirements. In CBSD this is done by reviewing old System Requirement documents if any available, interviewing with the stakeholders.

Once the requirements are thoroughly collected, requirement analysis process starts to identify common requirements of the system and subsystems, to identify and find possible reusable software components[18]. The major outcomes of this phase are : detail system requirements, identification of the components that can be reuse on the common requirements as system analysts has knowledge of available components in the in-house repository.

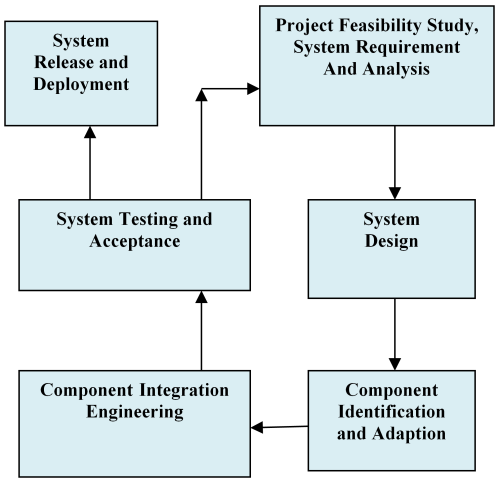

Figure 9. An Improved Model for Component Based Software Development

Main Function and non-functional requirements of the System is also identified in this phase, a system architecture model is drawn as per stakeholder's requirements and needs which have to be validated and verified with the stakeholders for any ambiguity or contradiction while requirements are collected and documented.

System requirements composition are decomposed into sub- requirements and searching for reusable component started to implement the requirement. This process may be repeated several times until there is an agreement between stakeholders and system analyst to go ahead from next stage in the lifecycle.

System engineering management plan must be used to document all methods which define system architecture and software. Evaluation through audits must be conducted frequently and regularly by an independent program organization[16].

4.1.2. Component Requirement and Selection

Once the system requirements are collected a system architecture model is designed based on the matching requirement. The software team determines from the system requirements which of the software requirements can be considered to composition rather than building them from the scratch.

A complete cost benefit analysis is required to know various cost involved is adopting the component like (full life cycle costs, maintenance costs, update requirement cost, licensing and warranty costs). These cost benefits analysis helps in making a decision to reuse a component or to acquire COTS[16].

Following points are considered for choosing the candidate components to implement the requirement. Candidate component is being searched first in the internally maintained repository for the availability of already developed reusable components.

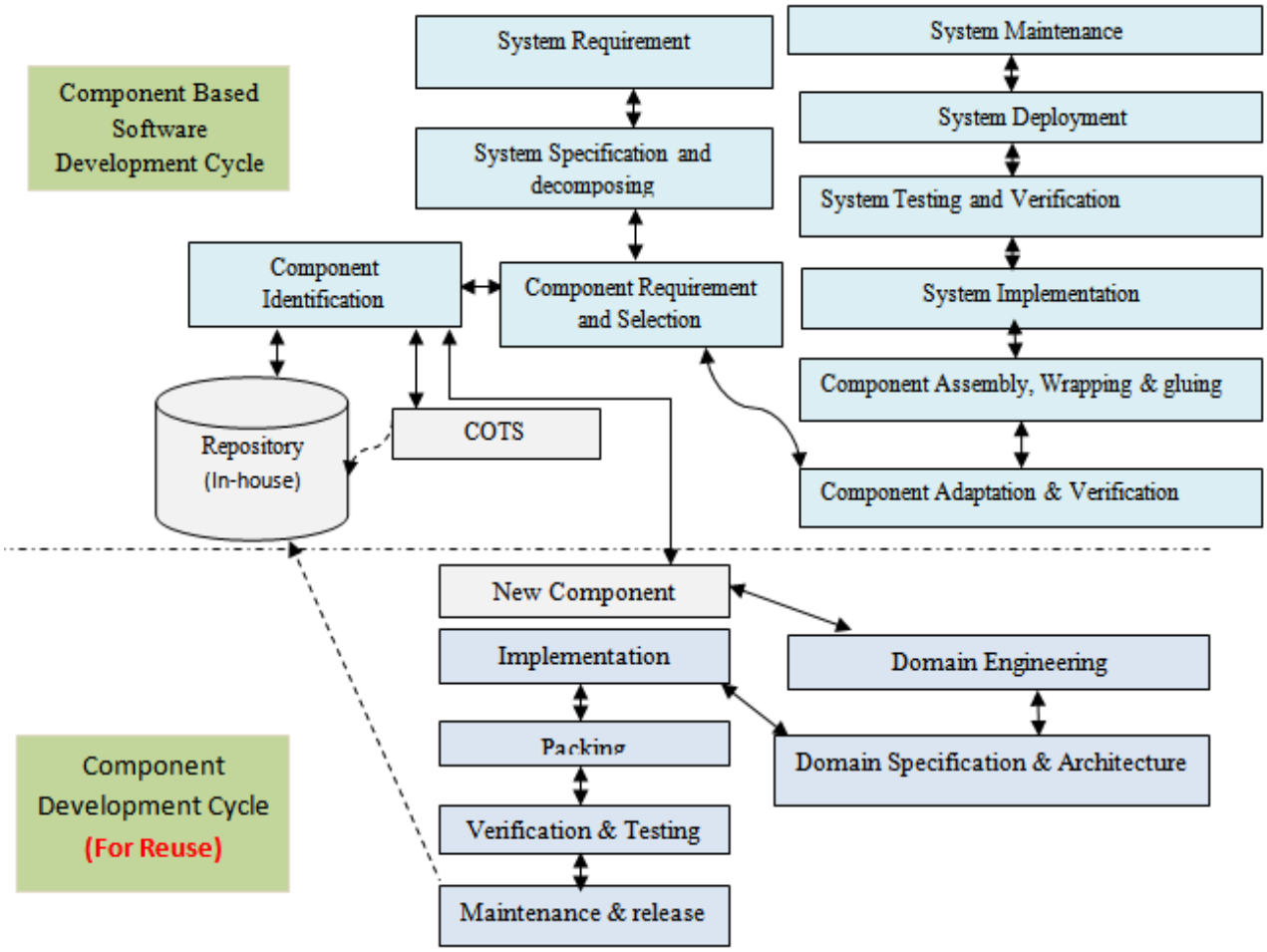

Figure 10. A detail view of An Improved Model for Component Based Software Development

If the candidate component is not available in repository, Commercial off the shelf (COTS) component is being searched in the market, which is a 3rd party tool and need to be purchased. If the candidate component is not available either in the internally maintained repository or in the market, a decision is being made to develop the Component from the scratch as per the requirement as shown in figure 10.

The use of reuse components, COTS, or any other non-developmental items should be treated as a risk and managed through risk management[16].

4.1.3. Component Adaptation and Verification

Once candidate component is selected its capability and fitness in the architecture issues need to be addressed, the selected component need to be fit in the system architecture design, a study usually being done to know how far the selected component is compatible with the system architecture.

In CBSE Requirement negotiation activity is mostly likely come in the picture since it is very difficult to find a component exactly matching with the requirement, component adaptation deals with making decision either to negotiation with the requirement so as to adapt the matching component, or to discard the selected component since it doesn't fit within the system architecture.

Verification of the component is being made through the software metrics and cost benefit analysis techniques[18]. The software architecture must be scalable, so that, components can be easily integrated into the system.

Prior to release for testing all reusable components including COTS need to be individually tested again the requirements[16].

4.1.4. Component Assembly, Wrapping and Gluing

Developing an application by integrating components needs communication among the components through an interface or "glue". Gluing components helps in controlling information flow among the components by interchange of data as well as adding functionality layer.

Component wrappers help in isolating a component from other components of the system. Component wrappers are helpful in mapping data representations, mapping data representations. Some critical functionality which COTS components don't provide internally can be achieved by wrappers.

4.1.5. System Implementation

Theoretically no coding is required during this phase but practically coding is usually required as all functionality is rarely found in a component and some functionality need to

be coded which is not provided by component.

4.1.6. System Testing and Verification

Testing an application investigate is the software going to deliver to a customer is a quality product and it really works as per the given requirement and specification.

In CBSD it lack of detail about component source code and design make it very difficult to track the faults specially occurs while using COTS components, addressing these issues are studied while integrating components in the architecture model of the system.

There is a need to investigate system's dynamic behavior in component integration testing. Some of the challenges of CBSD testing are often it is very difficult to evaluate component suitability for a particular use and framework, Missing functionality, misunderstood of Application Programming Interface (API). Customer's environment must be considered for System integration testing.

4.1.7. System Deployment

System Deployment phase evolved releasing product to a customer in other word software is made available to the customer for use. System deployment must be delivered using some specialized tool to make the deployment easy for the customer.

Different version of system release must be maintain and handle properly. If there is an up gradation in the system it must be maintain in a metadata and repository. Reconfiguration, adaptation, reinstallation of installed system need to be address properly to avoid run time issues usually found in installing system.

4.1.8. System Maintenance

Up-gradation and substitution of components are the main job of the system maintenance. System Up gradation usually occurs when a COTS supplier releases a new version of a component or when a new COTS component is obsolete. Modifications to the code wrappers and glue code are required in system maintenance.

4.2. Component Development Lifecycle

If the candidate component is not available either in the internally maintained repository or in the market, a decision is being made to develop the Component from the scratch, following are the stages of the component development for the reuse

4.2.1. Domain Engineering, Specification and Architecture

Domain engineering main aim is to have a generic system or component that can be used efficiently in different systems with little or no change. Development of reusable software is the main concern of domain engineering. Thus, Domain Engineering has to take into account different sets of customers which also including potential ones and usage contexts[14].

Ernest G. Allen's in his books give the following definition of Domain Engineering "Domain Engineering is the activity of collecting, organizing, and storing past experience in building systems or parts of systems in a particular domain in the form of reusable assets (i.e. reusable work products), as well as providing an adequate means for reusing these assets (i.e. retrieval, qualification, dissemination, adaptation, assembly, etc.) when building new systems"[14].

The component specification describes different properties that are important to realize by the corresponding component implementation. Some of the important properties are[15]:

- **Functionality:** what the main functionality of the component and how a component does.
- **Behavior:** What are the actions of a component and how it does perform those behaviors?
- **Interaction potential:** What are the possible ways in which a component interacts with other software?
- **Quality properties:** what are the Characteristics of a component such as performance, portability and stability?

The interface of a component also plays an important role in execution time. Components communicate with each other through interface. Interfaces are offered by one component in order to be used by other components. The component that offers an interface is responsible for realizing the actions of the interface[15].

4.2.2. Implementation

Table 1. Comparison among different CBD Models

Activity	V Model	Y Model	W Model	X Model	ELCM	Our Improved CBSD Model
Domain analysis	✓	✓	✓	✓	✓	✓
Component search	✗	✗	✓	✓	✓	✓
Component evaluation	✓	✓	✓	✓	✓	✓
Component selection	✓	✓	✓	✓	✓	✓
Component adaptation	✓	✓	✓	✓	✗	✓
Component integration	✓	✓	✓	✓	✓	✓
Component evolution	✗	✗	✗	✗	✓	✓

In This phase Implementation tools and techniques are understood. The main concern of this phase is to develop a generic component which can be easily reusable and adaptable in similar other domain. Study of the framework is required as component usually developed based on an architectural framework and follow come model like CORBA, RMI, DCOM etc

4.2.3. Packing

Before pacing all the reusable components are given a rank based on the percentage of reuse and on the basic of cost benefit analysis[18]. Ranking helps the concerned person to identify and select the components.

In packing stages components are reshaped in such a way that it can be easily understood in the hierarchy. A bundled of single unit is prepared in this stage which contains files that form the component. Implementation manuals are also prepared in this stage[18].

4.2.4. Verification and Testing

This is a very important phase of component development for reuse, in this phase verification and testing of the component is being done to know the capability and accuracy of the component. The new components are designed, developed and tested on unit basis. Integration and system tests of the newly developed and of the reused components are performed. A customer is requested to evaluate and verify software, whether it meets his/her requirements or not during the testing phase. The software is ready to deploy at customer site

4.2.5. Deployment and Release

Unpackaged in a form that can be installed on the target machine [15]

5. Comparison among different CBD Models

Table 1. List comparison among the different CBD Models with our improved CBD model, clearly the table 1 shows our Improved CBSD Model provides improvement in CBSD life cycle activities.

6. Conclusions and Future Work

In this paper we have explained a systematic software process to apply the reused based approaches successfully in software development and software process for reuse based approaches is different from traditional process. The traditional software processes do not consider the software reuse explicitly and cannot support the risks that are attached with the use of reusable software components. Component based software engineering is a systematic way to achieve the software reuse. In this paper we have discussed various CBSD lifecycle models and described different phases of these lifecycles.

The major difference between the traditional software development processes and CBSD processes is a separation of system development from component development. We have surveyed different software lifecycle models for CBSD and all have their own advantages and disadvantages. CBSE is still an emerging field in software engineering and there is much space for research in this field. Although CBSE has advantages but it has also disadvantages such as component maintenance costs, changing requirements (project specific requirements), unsatisfied requirements, repository management and component's version handling etc.

We can overcome these advantages by merging CBSE and traditional software engineering. Project specific requirements should be satisfied by traditional software engineering approach so that standard components can be used as black-box components. It is needed to plan sequence of experiments in which relative costs and benefits of choosing a component based software development can be weighed against the choice of a traditional software development. So in future work we are planning to extend the set of experiments and implementation. Also, we will provide formal justification for the proposed model.

REFERENCES

[1] H. Hansson, M. Åkerholm, I. Crnkovic, M. Törngren, "a Component Model for Safety-Critical Real-Time Systems", in Proceedings of 2004 30th EUROMICRO Conference (EUROMICRO'04), France.

[2] "Research Areas of the Software Engineering Group", Online Available: http://www.cs.uni-paderborn.de/en/research-group/software-engineering/research/research-areas.html

[3] "Basic Concepts of Component-based software", Online Available: http://www.idt.mdh.se/kurser/cdt501/2008/lectures/book%20Basic%20Concepts%20of%20CBSE.pdf

[4] Murat güneştaş , "A study on component based software engineering", a Master's thesis in Computer Engineering, Atılım University, JANUARY 2005

[5] Ivica Crnkovic; Stig Larsson; Michel Chaudron, "Component-based Development Process and Component Lifecycle." Online Available: http://www.mrtc.mdh.se/publications/0953.pdf

[6] 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.

[7] K.Kaur;H Singh, "Candidate process models for component based software development", Journal of Software Engineering 4 (1):16-29, Academic Journal Inc, India 2010.

[8] Syed Ahsan Fahmi; Ho-Jin Choi, "Life Cycles for Component-Based Software Development", IEEE 8th International Conference on Computer and Information Technology Workshops 2008.

[9] The W Model for Component-based Software

Development[online]. Online Available: http://www.cs.man.ac.uk/~kung-kiu/pub/seaa11b.pdf.

[10] Kotonya G; Sommerville I; Hall S, "Towards A Classification Model for Component-Based Software Engineering", in Proceedings of 2003, Euromicro Conference, 29th, Dept. of Computer., Lancaster Univ., UK, 1-6.

[11] Ivica Crnkovic, "Component-based software engineering for embedded systems", ICSE '05 Proceedings of the 2005, 27th international conference on Software engineering, ACM New York, NY, USA.

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

[13] Lata Nautiyal; Umesh, K; Sushil, C, "Elite: A New Component-Based Software Development Model", Int.J.Computer Techology & Applications,Vol 3 (1),119-124.

[14] Ernest G. Allen's, Chapter 3, Domain Engineering, Online Available:http://users.owt.com/eallen/other/domain_engineering.pdf

[15] Online Available:http://centurion2.com/SEHomework/ComponentBasedSE/ComponentBasedSE.php#ProductLineDevelopment.

[16] Critical software practices, Pro-Concepts LLC, Online Available: http://www.spmn.com/www2/16CSP.html#system

[17] John C. Dean, CD Dr. Mark R Vigder, System Implementation Using Commercial Off-The-Shelf Software, Software Engineering Group, NRC Report Number 40173, Ottawa, Ontario, Canada, 97

[18] Sajid Riaz, Moving Towards Component Based Software Engineering in Train Control Applications, Final thesis, Linköpings universitet, sweden, 2012