Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 5.258

IJCSMC, Vol. 5, Issue. 8, August 2016, pg.173 – 186

'Z' Maturity Model for Testing in Component Based Development

Dr. Latika Kharb
Associate Professor (IT),
JIMS, Sector-05, Rohini, New Delhi, INDIA
latika.kharb@jimsindia.org

Abstract: Various tools and techniques are invented by researchers and practitioners to improvise software developments like lower software development costs, shorter time-to-market and good quality product leading to increased productivity. One such practice is development of software using component based software development (CBSD) techniques i.e. building software systems using existing reusable components, instead of writing from scratch. The main objective of CBSD is to write once and reuse any number of time with no or minor modification. 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've proposed a Z maturity model for Component Based Software Development. The paper is organized as: section two describes component based software development lifecycle, section three outlines some popular component based development models and their comparisons, section four presents improved component based software development model called as Z maturity model for testing, section five describes conclusion and future work.

Keywords: Component based software development, quality product, reusable components, and software development costs.

1. Introduction

Over the time, researchers and practitioners have expressed their inability to accurately estimate costs associated with software development and it has become even more problematic as costs associated with development continue to increase. Software projects are illustrious for going beyond their deadline, going over budget, or both and the problem lies in the estimation of the amount of effort required for the development of a project. In early days of computing, software costs constituted a small percentage of overall computer-based system costs but today; software

is the most expensive element of virtually all computer based systems [1]. In the present era of information technology, software industry has enormous pressure of meeting the product deadlines with constraints like minimum development time as well as minimum development costs. For this, an important prerequisite for cost and time optimized software development is to follow the software development lifecycle models. Today, more and more software companies are adopting component-based development (CBD) methodologies to meet the demands of customers. Typically, the development of component based systems starts with a collection of existing components [2]. As component-based software paradigm develop software from assembling of existing components, a company may avail advantage by opting CBD for the software development like:

- *reduced development time : as less or no coding is involved;*
- *meet strict deadlines: as development time is reduced;*
- *increased software productivity: as software are built by integrating already developed components instead of writing them from scratch;*
- *reduced risks for creating new software : due to reuse of a component in several other similar domains.*

In CBD, the inner structure of the component and the implementation details are hidden which 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 [3]. Lego (figure 1) is often taken as an example of a component- based approach: Lego is a game of blocks: it contains a number of blocks that can be integrated to make different toys like cars, trains and airplanes.

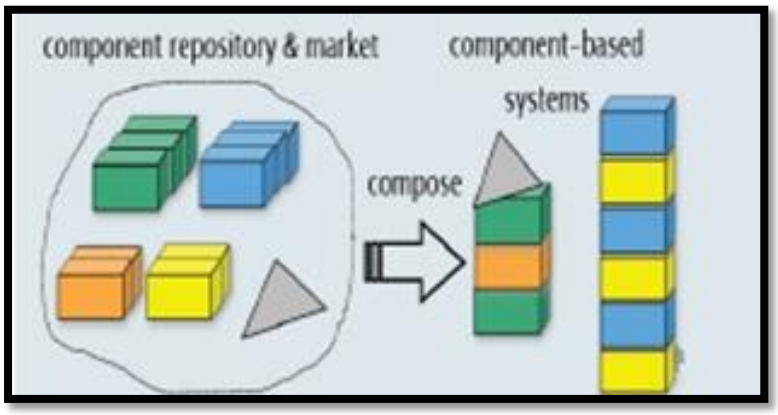

Figure 1: Concept of Component-based Systems

2. Component based Software Development Lifecycle

Delivering a high quality reliable product is the main focus in any software development [4]. Components play a critical role in many software systems. Thus, our ability to reason about the properties of assemblies of components is of great concern to modern system implementers [5]. The life cycle of component-based software systems is different from that of the traditional software systems. In component-based software development process, systematic reuse of existing COTS components: also called commercial off-the-shelf components, is carried out

along with their integration in order to develop an application rather than building and coding overall application from scratch. The life cycle of component based software systems can be summarized in following stages (figure 2):

- *Requirement Analysis and Specification Stage*
- *Design Stage*
- *Implementation Stage*
- *Integration Stage*
- *Testing Stage*
- *Release and Maintenance Stage*

2.1. Requirement Analysis and Specification Stage

In a component-based approach, requirement analysis implies that it is necessary to analyze whether requirements can be fulfilled using available components. Availability of existing components is considered in requirements specification i.e. the analysts have to be aware of the components that can be possibly reused. Keeping in mind that the available components are appropriate and fulfill the entire requirement; negotiate the requirements and modify them to be reusable in order to utilize advantage of component-based approach.

2.2. Design Stage

During this phase, a decision about the possible use of the component model is made since this decision plays a crucial role in the architecture design of the system as well as the quality of the system.

2.3. Implementation Stage

As all functionality is rarely found in a single component and some functionality need to be coded that is not provided by the COTS component. Finding a COTS component as per the requirement is the main focus of this phase.

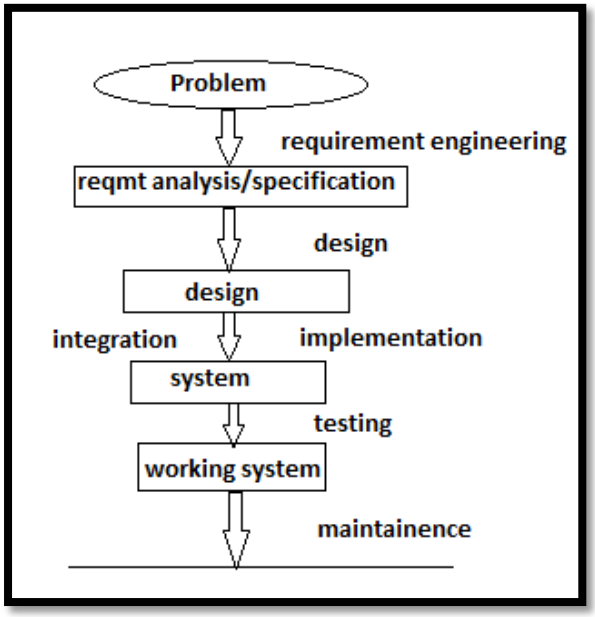

Figure 2: Component based Software Development Lifecycle

2.4. Integration Stage

Integration is the most important and critical phase of CBSE as component integration mismatch most likely occurs because of lack of communication and capability issues. System quality and functionality need to be validated and verified in this phase to know the effectiveness of the assembled components.

2.5. Testing Stage

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 [6].

2.6. Release and Maintenance Stage

Software release is made in such a way that it is suitable for delivery and installation. As maintenance phase usually involves replacement of old component with the new component in the system; so testing has to be done to check the proper integration of component into the system.

3. Literature Review: Some Popular CBD Models

A number of different CBD models appear in industry as well as in academia. Some of the popular state of art CBD models have been discussed in the following section namely; V model, Y model, W model, X model and ELCM model.

3.1. The V Model

The V model [7] adopted the traditional software development approach for building a system from reusable software components.

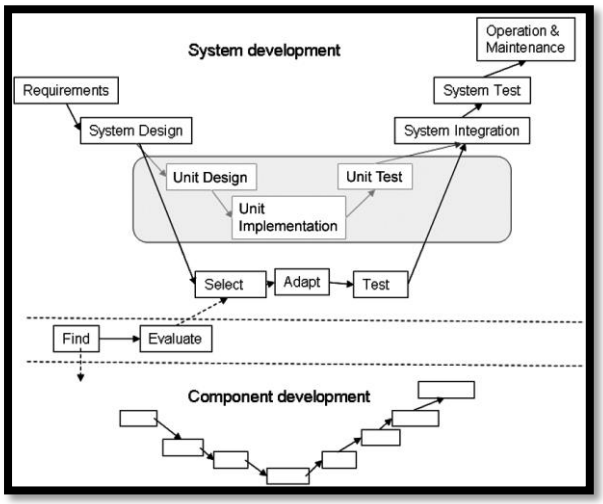

Figure 3: V Model

The V Model (figure 3) is an adaptation of the traditional waterfall model for traditional software development. In V model, each phase must be completed before the next phase begins and testing is emphasized in this model more than the waterfall model.

3.2. The Y Model

Capretz [8] [9] proposed a new life cycle model known as Y model for component-based development. The Y model (figure 4) consists of following planned phases: domain engineering, frameworking, assembly, archiving, system analysis, design, implementation, testing, deployment and maintenance.

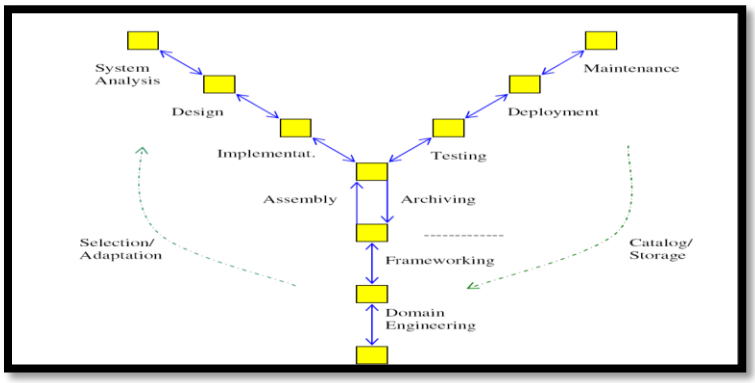

Figure 4: Y Model

In this model, some new phases were basically proposed like domain engineering frameworking, assembly and archiving with the other traditional life cycle phases. 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.

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 [10]. Component lifecycle comprises of two major phases: component design and component deployment(figure 5).

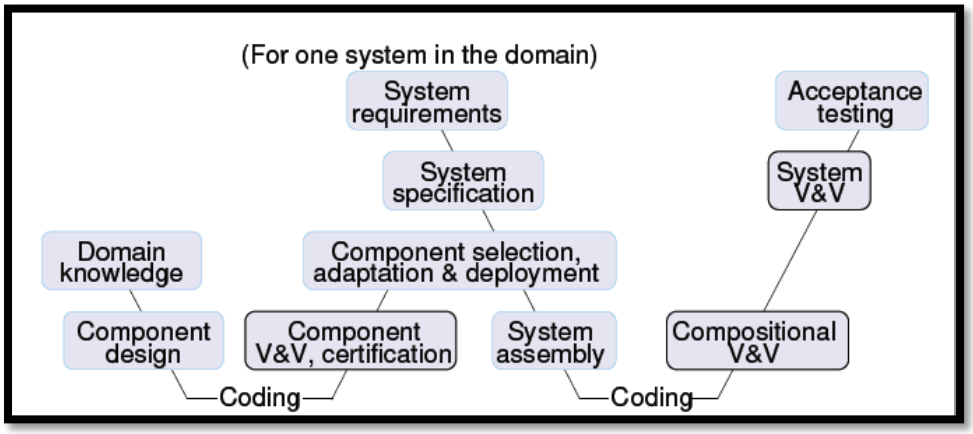

Figure 5: W Model

3.4. The X Model

Tomar and Gill [11] proposed the X Model in which the processes started in the usual way by requirement engineering and requirement specification. This software life cycle model (figure 6) mainly focuses on the reusability where software is built by building reusable components for software development and software development from reusable and testable components.

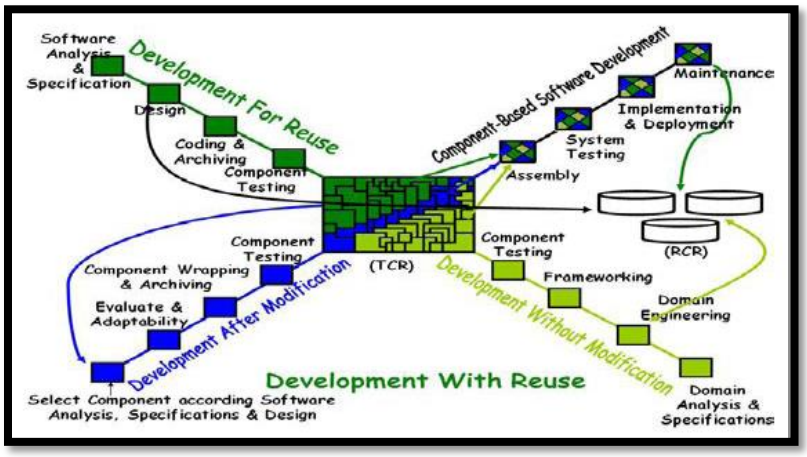

Figure 6: X Model

The X model basically considers the following cases: development for reuse, development after modification, development without modification and component based software development.

3.5. Elite Life Cycle Model (ELCM)

Lata Nautiyal, Umesh and Sushil [12] proposed Elite Life Cycle Model (ELCM) (figure 7) for the development of new product using component based technology as a viable alternative to address software reusability during component-based software production.

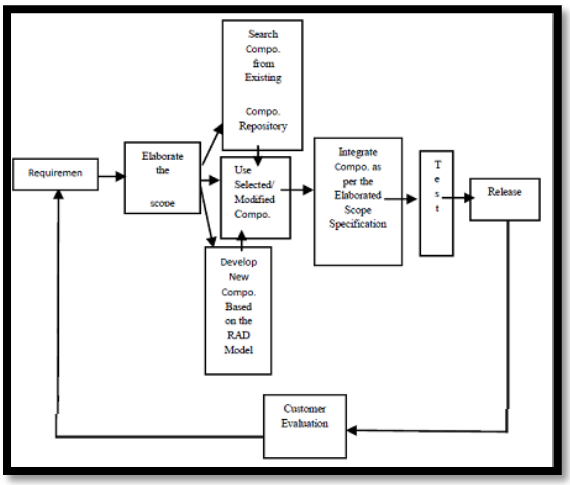

Figure 7: ELCM Model

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.

4. Proposed Z Model for Component Maturity in Testing

Component-based software development (CBSD) has become one of the preferred streams for developing large and complex systems by integrating prefabricated software components that not only facilitates the process of software development but is also changing the ways for software professionals to develop software applications [13]. In this paper, we've discussed some of the popular models and from literature review section, we conclude that all CBSD lifecycle models have some advantages and drawbacks. So, there is a need of a new lifecycle for component based software development (table 1).

ACTIVITY	V MODEL	Y MODEL	W MODEL	X MODEL	ELCM
Domain Analysis	✓	✓	✓	✓	✓
Component Search	✗	✗	✓	✓	✓
Component Evaluation	✓	✓	✓	✓	✓
Component Selection	✓	✓	✓	✓	✓
Component Adaptation	✓	✓	✓	✓	✗
Component Integration	✓	✓	✓	✓	✓
Component Evolution	✗	✗	✗	✗	✓

Table 1: Comparison between Some CBD Models

Figure 8 shows details of our proposed Z Model: reusing of existing artifacts is the most important concern of the Component Based Software Development. The main phases of our improved Z model include: *standard component testing, manageable component testing, certified component testing and systematized component testing*.

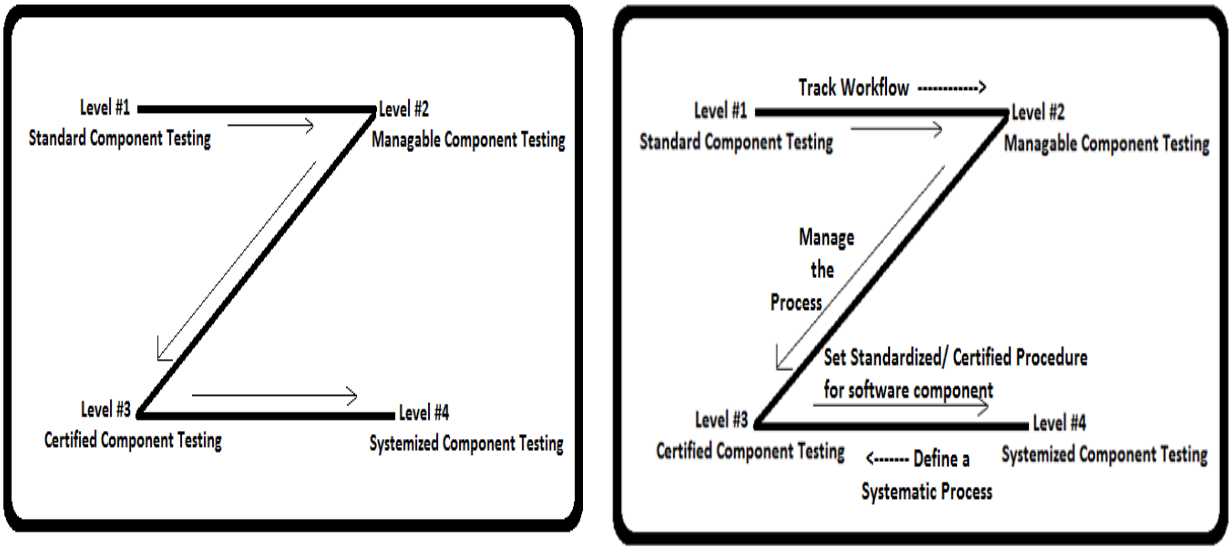

Fig 8: Flow of control in Z model

As component engineering gains a wide acceptance in today's software industry, many software companies have begun to set up component-based software product lines. Building cost-effective products needs an effective product line and for delivering high quality component-based software, we need a well-defined component testing process. In order to understand the status of a component test process, it is important for a manager to have a tool to measure the effectiveness of the process. The proposed Z maturity model could act as a tool that can be useful to measure the maturity level of a process in an organization. In this section, we've described our Z maturity model that could help managers to measure the status of a component test process. It focuses on four levels in the Z model (figure 8) namely:

- *Level 1: Standard Component Testing*
- *Level 2: Manageable Component Testing*
- *Level 3: Certified Component Testing*
- *Level 4: Systematic Component Testing*

1. Level 1: Standard Component Testing:

The first level of Z maturity model consists of four main phases namely, testing standards, pre-defined management procedures, test criteria and well defined mechanisms.

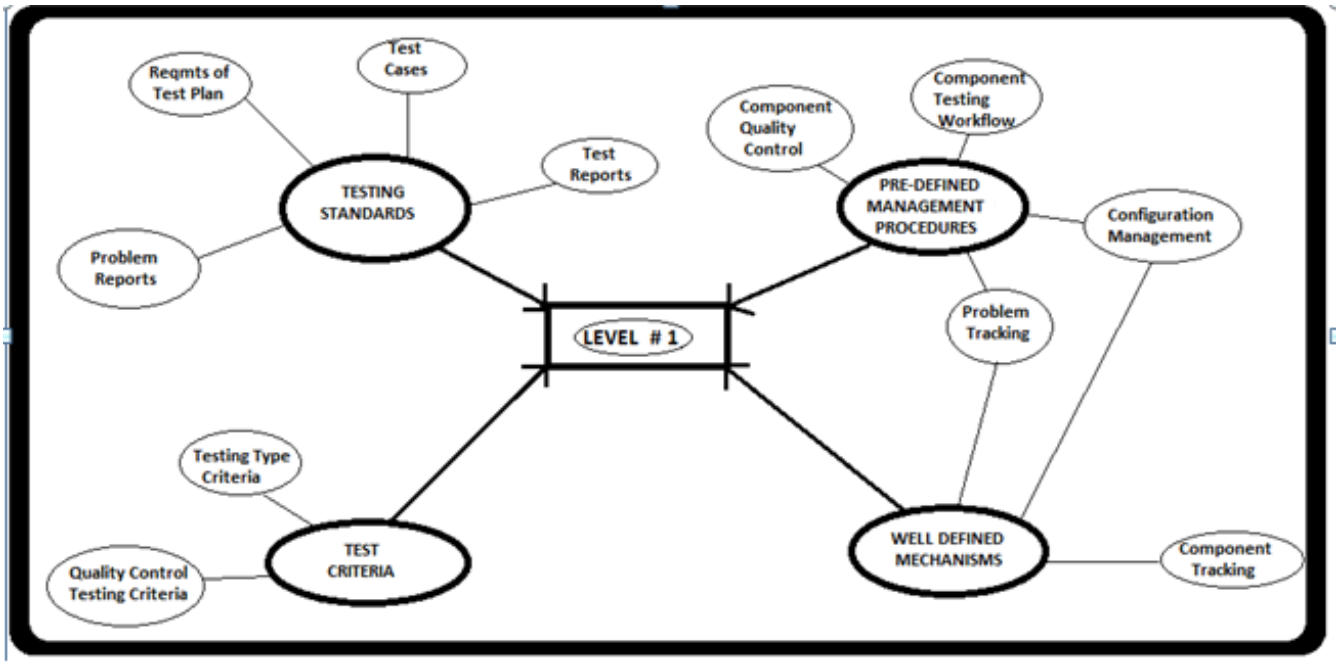

Fig 9: Level 1 of Z Maturity Model

Various mechanisms of each stage (figure 9) have been discussed below:

(i) **Testing Standards:** It comprises of four sub phases: generating problem reports, requirements of test plan, test cases and test reports.

(ii) **Pre defined Management Procedures:** It comprises of four sub phases: problem tracking, component quality control, component testing workflow and configuration management.

(iii) **Test Criteria:** It comprises two sub phases: testing type criteria and quality control testing criteria.

(iv) **Well Defined Mechanisms:** It comprises of two sub phases: problem tracking and component tracking.

In level 1 of Z model, we've characterized a component testing process as a standard process when managerial/ engineering activities are performed based on a set of well-defined criteria viz. test reports and/or management procedures like configuration management. The well-defined criteria include requirements on test plan/ test case and test reports. Management procedures include a component quality control, component tracking etc.

2. Level 2: Managed Component Testing:

The second level (figure 10) of Z maturity model consists of four main phases namely, test cost measurement, quality measurement, test measurement and process management.

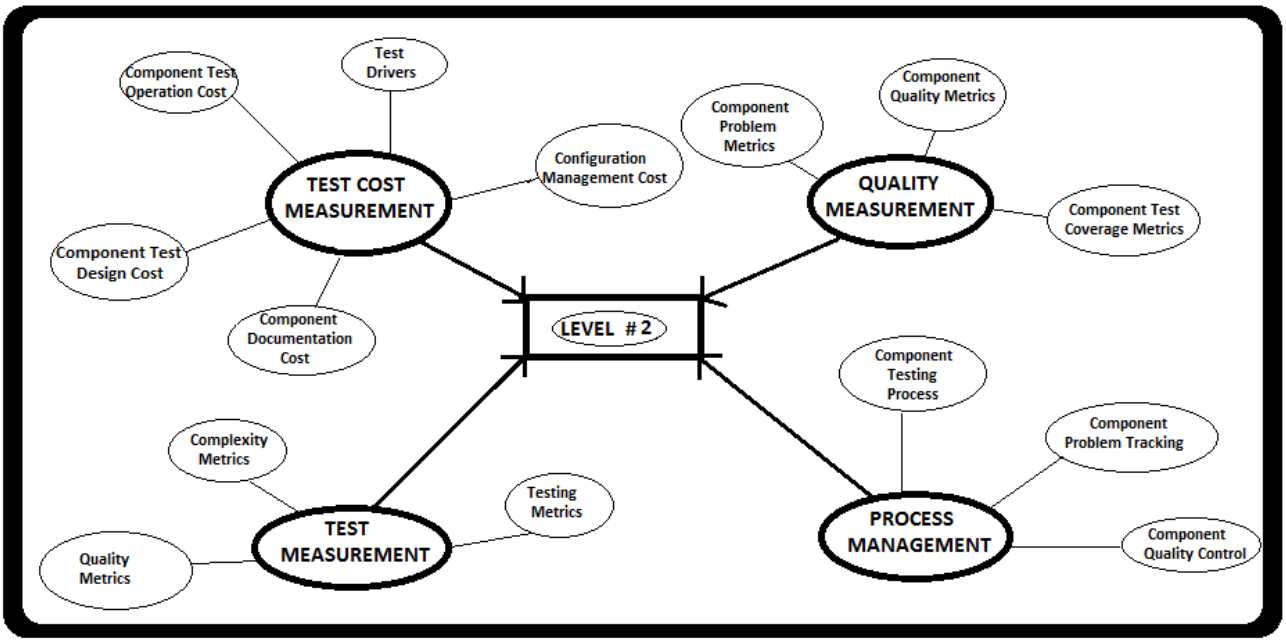

Fig 10: Level 2 of Z Maturity Model

Various mechanisms of each stage have been discussed below:

(i) **Test Cost Measurement:** It comprises of four sub phases: component documentation cost, component test design cost, component test operation cost and test drivers.

(ii) **Quality Measurement:** It comprises of three sub phases: component problem metrics, component quality metrics, and component test coverage metrics.

(iii) **Test Measurement:** It comprises of three sub phases: testing metrics, complexity metrics and quality metrics.

(iv) **Process Management:** It comprises of three sub phases: component testing process, component problem tracking, and component quality control.

In level 2 of Z model, we've characterized a component testing process as a managed process as it collects the detailed measures of the process like component quality, component test cost, component test metrics and component quality metrics.

3. Level 3: Certified Component Testing:

The third level of Z maturity model (figure 11) consists of three main phases namely; component certification criteria, component certification standard, certification process management and managed component testing (level 2).

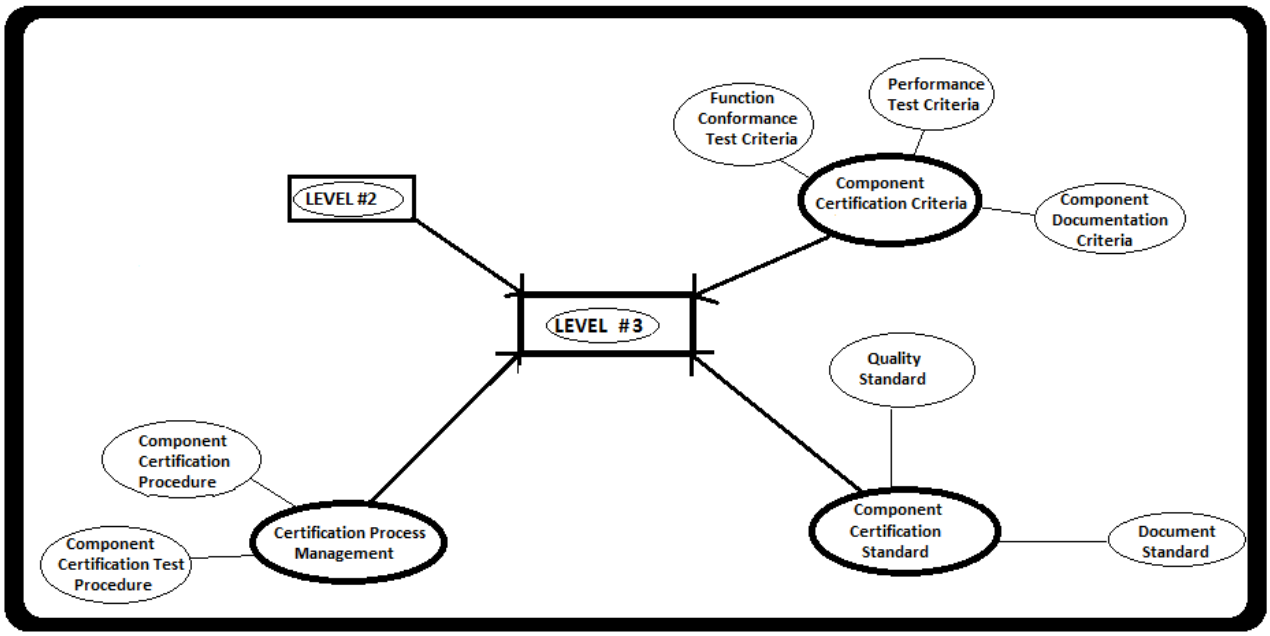

Fig 11: Level 3 of Z Maturity Model

Various mechanisms of each stage have been discussed below:

(i) **Component Certification Criteria:** It comprises of three sub phases: performance test criteria, component documentation criteria and function conformance test criteria.

(ii) **Component Certification Standard:** It comprises of two sub phases: quality standard and document standard.

(iii) **Certification Process Management:** It comprises of two sub phases: component certification procedure and component certification test procedure.

(iv) **Managed Component Testing:** It is level 2 of Z model and is used as extension in this level 3.

In level 3 of Z model, we've characterized a component testing process as a certified process as it defines and implements a certification standard/procedure for software components. The certification process includes certification procedure, certification tests, certificated quality standards, documentation standards etc. For completing the certificate tests for a component, the designated engineer will issue a certificate for a component product on the basis of level 2.

4. Level 4: Systematic Component Testing:

The fourth level of Z maturity model (figure 12) consists of four main phases namely, component test suite, component design for testing, configuration management and component test support. Various mechanisms of each stage have been discussed below:

(i) **Component Test Suite:** It comprises of four sub phases: test selection, test generation, test suite construction and test selection for integration.

(ii) **Component Design for Testing:** It comprises of three sub phases: behavior tracking mechanism, tracking interface, test interface and test mechanism.

(iii) **Configuration Management:** It comprises of five sub phases: component problems, component test suites, component documents, component programs and test mechanism.

(iv) **Component Test Support:** It comprises of six sub phases: problem reporting, test driver generation, test stub, test result checking, component test platform and component integration.

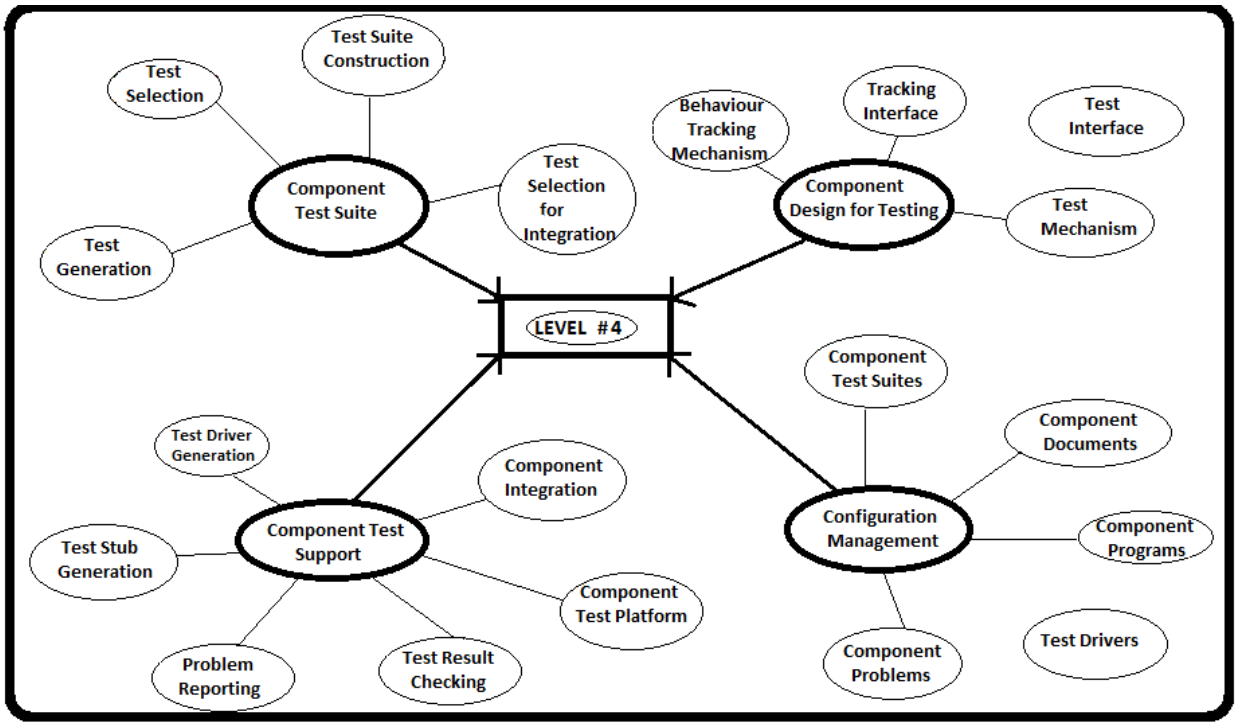

Fig 12: Level 4 of Z Maturity Model

In level 4 of Z model, we've characterized a component testing process as a systematic process as it has defined systematic mechanisms to complete this process. To complete the process, we need well-defined systematic methods to support their operations and activities like component test suite, component test support, component design for testing and configuration management.

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.

User Satisfaction = Compliant Product + Good Quality + Delivery within Schedule [14]

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 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 CBSD and traditional software engineering. We have discussed on the need to plan a 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 and we will provide formal justification for the proposed model.

References:

[1] Latika Kharb: Proposed C.E.M (Cost Estimation Metrics): Estimation of Cost of Quality in Software Testing: International Journal of Computer Science and Telecommunications, Volume 6, Issue 2, February 2015.
[2] Latika Kharb: Assessment of component criticality with proposed metrics, INDIACom-2008: Computing For Nation Development, sponsored by AICTE, IETE, and CSI: INDIACOM-2008
[3] Research Areas of the Software Engineering Group, Retrieved on November 18th 2011 from: http://www.cs.uni-paderborn.de/en/research-group/software-engineering/research/research-areas.html.
[4] Latika Kharb: Proposing a Comprehensive Software Metrics for Process Efficiency: International Journal of Scientific & Engineering Research, Volume 5, Issue 9, September-2014 78 ISSN 2229-5518.
[5] Latika Kharb: CCTF: Component Certification & Trust Framework: International Journal of Scientific Research in Computer Science and Engineering: Vol-1, Issue-6 ISSN: 2320-7639.
[6]Available online:
http://centurion2.com/SEHomework/ComponentBasedSE/ComponentBasedSE.php#ProductLine Development.
[7] Ivica Crnkovic; Stig Larsson; Michel Chaudron, "Component-based Development Process and Component Lifecycle." Online Available:
http://www.mrtc.mdh.se/publications/0953.pdf
[8] 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.
[9] 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.
[10] The W Model for Component-based Software Development[online]. Online Available: http://www.cs.man.ac.uk/~kung-kiu/pub/seaa11b.pdf

[11] 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.
[12] Lata Nautiyal; Umesh, K; Sushil, C, "Elite: A New Component-Based Software Development Model", Int.J.Computer Techology & Applications,Vol 3 (1),119-124.
[13] Latika Kharb: Complexity Metrics for Component-Oriented Software Systems: ACM SIGSOFT Software Engineering Notes Page 1 March 2008 Volume 33 Number 2.
[14] Latika Kharb: An Agent Based Software Approach towards Building Complex Systems, TEM Journal, Volume 4, Number 3, Pg. 287-291, August 2015.

