International Journal of Scientific Engineering and Science
Volume 2, Issue 1, pp. 26-32, 2018. ISSN (Online): 2456-7361

Component Based Software Development – "An Efficient Approach"

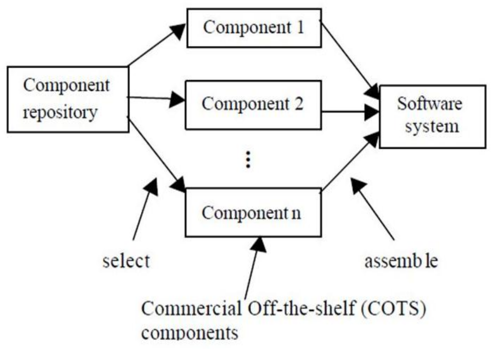
Dr. Parul Gandhi[1], Kritika Vashisht[2], Kunal Dawra[3], Shanu Jaitly[4], Prasenjit Banerjee[5]
[1]Associate Professor, Faculty of Computer Applications, Manav Rachna International Institute of Research & Studies, Faridabad
[2, 3, 4]Students, Faculty of Computer Applications, Manav Rachna International Institute of Research & Studies, Faridabad
[5]Research Mentor, Accendere KMS Pvt.Ltd

***Abstract*—*Elemental role of component based software engineering is to develop the systems of an assembly of parts, the development of parts in the form of reusable institutions, and maintenance and upgrading of systems by customizing and altering such parts. Component-based software Development (CBSD) is used to develop / collect Software from existing components. Component based software development follows some principles. Paper shows a study of various software reusable concepts. The aim is to gather useful information on software components and the component on which component reusability is highly dependent, as a result of the review of the software reusability that is very dependent on reusability, cost, optimization, and complexity of the interface, documentation quality and portability. In this paper, the main description is about component based software development where the components can be reused and combined together to form a software system that meets the user's actual requirements. This paper also shows different models of component based systems and the factors that affect component based development.*

***Keywords*— *Component Based Software Development, Reusability, Systematic Software Reuse, Software Component.*

I. Introduction

A single software component is a program package, a web service or resource that has a set of similar tasks (or data). All system processes are kept in separate components so that all the data and functions within each component are inconsequential (such as with the contents of the classes). Due to this principle, it is often said that components are modular and united.



In most building disciplines, frameworks are outlined by forming existing segments that have been utilized as a part of different frameworks. Programming building has been more centered on unique improvement yet it is presently perceived that to accomplish better programming, all the more rapidly and at bring down cost, we have to receive a plan procedure that depends on deliberate "reuse".

In this way, normally, segments are fairly little however autonomous parts of a framework. Be that as it may, a vast framework all in all can be viewed as a part too. It is imperative to perceive segments are runtime substances. They exist while the framework is running, actually: the framework comprises of segments, and is a part itself. Segments are not simply plan substances like classes in protest introduction are. As said before, at his minute everyone is raving about parts, and appears to expect a ton from it. What is normal shape parts, and why is everyone that aficionado? The normal points of interest to be gotten from the utilization of segments are outlined in later areas of this report. Segments give an administration without respect to where the part is executing or its programming dialect. A segment is a free executable substance that can be comprised of at least one executable articles. The part interface is distributed and all associations are through the distributed interface. Parts can extend in estimate from basic capacities to whole application frameworks.

Component Based Software Development can be referred as the process which uses reusable software components to design and construct the software. Rather than just coding in a particular style, its main focus is on reusing and adapting the existing components. Thus Component Based Software Development encourages both programming as well as composing the software.

Elementary role of component-based engineering is, addressing the development of the system as an assembly of parts, in the form of reusable modules, and the maintenance and upgradation of the system by customizing and altering such parts. This requires established methods and equipment support in which the entire component and system lifecycle is included in technical, organizational, marketing, legal, and other aspects. In traditional themes of software engineering, new methods are needed to support component-based development.

There are many advantages of using components based software system as it allows various aspects like structural, behavioural, functional processes. It helps in developing communication components with the help of different transformational techniques. Communication components are mainly used for the effective communication which is taken as manual process of transformation. Through which the implementation of the same can be done. In this a protocol is used which includes various types classical component properties that helps in making the framework more efficient.

The component based software is has some disadvantages such as in state of art middleware, where composition of application components are not properly designed due to which its applicability do not perform well in the designed framework. Due to which they are not able to identify the requirement of the application component interface. The composition of AO deals with some complex processing in which deployment descriptor of the applications tends to exists. In this cross cutting of composition of application, components are matched with the application service.

Reusability plays an important role in CBSD and it also works as the basis of CBSD. The basic benchmark for calculate the component is reusability if an element is not reusable then the full approach of component-based software development falls.

In most engineering subjects, systems are created by creating systems that are designed by creating existing components being used in other components. Current components are used in other systems. Software engineering is focusing more on basic development but it is now recognized that in order to achieve better software, at a faster and lower cost, it needs to adopt a design process which is based on the sign process: Systematic Software Reuse.

In order to reuse the design components, follow the components, you have to follow the verdict made by the original developer of the component. • It can limit the chances of reusing it; it can limit the chances of using it again. Nevertheless, a more abstract form of reuse is reused in concept, nevertheless, again. The concept is reused in a more abstract form of usage, when a particular approach is described independently in the implementation and after that an implementation is developed. And an implementation has been developed. • There are two main approaches to concept reuse: two main approaches to concept reuse are: • Design patterns • Generating programming.

Software reusability is a feature that refers to the expected re-use capability of the software component. The software evolution community is constantly moving toward the promise of complete software reuse, in which any new software system can be captured practically from the existing code. Consequently, an increasing number of organizations are using the software, not just in the past but in the form of component components of larger applications, as all-inclusive applications. In this new aspect, the acquired software must be unified with other software functionality.

Reusability is a significant feature of a high-aspect software fundamental. Programmers should design and implement such software components so that they can reuse many different programs.

II. CBSE is Based on Sound Software Engineering Design Principles

(1) Components are free so do not interfere with each other
(2) Component usage is covered
(3) Communication is well through communicating well through specialty interfaces
(4) Component steps are shared and reduce correction costs.

III. Objectives of Component Based Software Engineering

The primary goals of the part-based programming building are given below
(1) Cost and time constraints for the creation of comprehensive and unscrupulous frameworks: The primary goal of component-based approach is to create a complicated programming framework using the rack segment, which is a fundamental effect of collecting the product. . The adequacy of the cost of the current technique can be dissected using efficiency points or using different strategies.
(2) Improve the nature of the product: By increasing the nature of the component, the nature of the product can be increased. Although this idea is not valid when everyone has done it. In some cases the nature of the collected settings cannot be clearly identified by the nature of the keyword that by increasing the nature of the clause does not actually suggest the change of the framework
(3) Detection of incomplete within the framework: From component perspective, this section is encouraged to check so that it can be immediately dealt with by examining sections. In any case, it is difficult to find out the loopholes of the loopholes whether there should be an incident of a part-improvement approach.

IV. Examples of Component Based Approach

To begin with, let us give a case of straightforward stereo framework, in which there is a possibility of a wafer, sub-wave, sound box and so forth that someone should create stereo frameworks by closing the rack segment like a sound box. Then he can face some favorable conditions compared to those people who build the structure from the original circuits.

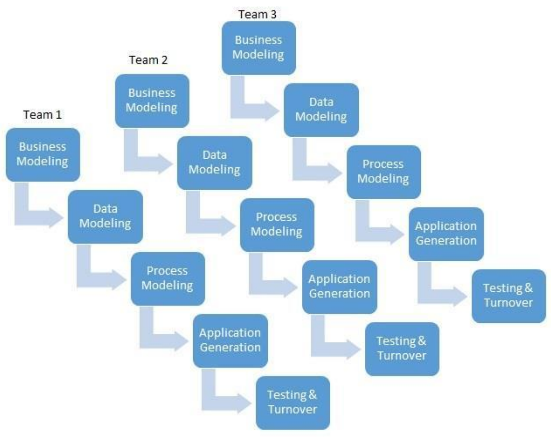
Indeed, nowadays, the part of every straight and circled structure is used to use, where some sections are made by some engineers and they are kept in the library for reuse. The primary consideration is that the rack parts should not be changed for their special purposes in order to stop. On the off chance, we need to change a portion of the parts to fit their hard work, we have to modify it and make them Must be stored in.

There is a case of some complex designs, where each section can be seen as a framework, it is a combat structure of the navy. There are some radars, helicopters, submarines, rocket launchers, some military aircraft and so forth to identify the structure. Every section has some huge frameworks here and in this case, affiliation is also quite confused.

V. Component Based Software Development Models

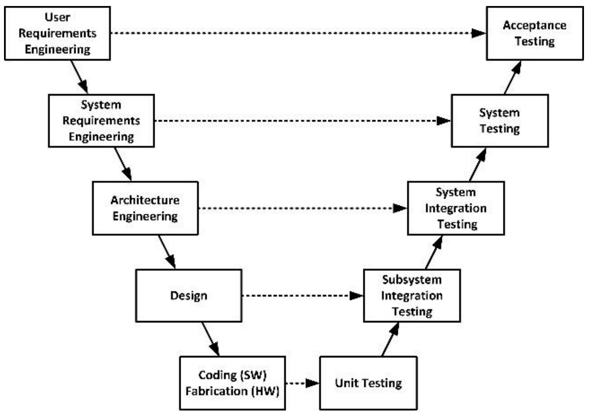
1. *Rapid Application Development (RAD)*

This model uses component based construction approach for fast development. There are some limitations to this model like RAD teams. This model is not correct when there is high technical risk. This model is used when requirements and solutions can be composed to independent software components, where each module could be develop by different teams. After this small modules are combined to form a large software system.



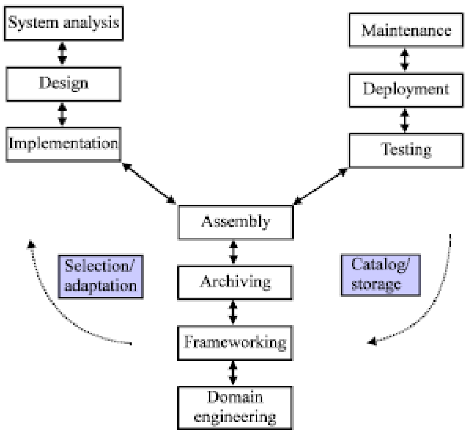
2. *V-Model*

The V-Model can be used connect many other process models. "V" can be called as graphical arrangement of the different phases. Verification and Validations are the two synonyms of "V". This model is very simple to develop and easy to understand. Time connection development and test activities gets clear by ordering the activities in this model.


User Requirements Engineering
System Requirements Engineering
Architecture Engineering
Design
Coding (SW) Fabrication (HW)
Unit Testing
Subsystem Integration Testing
System Integration Testing
System Testing
Acceptance Testing

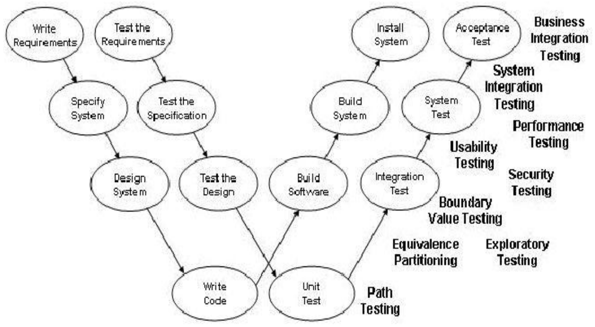
3. *Y-Model*

"Y" Software Life Cycle Model describes software reusability during CSBD. The "Y" size of the model understands recurrence and overlapping. Although each other can be overlapped and allowed to run in the main steps, these are the schematic steps: domain engineering, frame working, assemblies, archive, system analysis, design, implementation, testing, deployment and maintenance



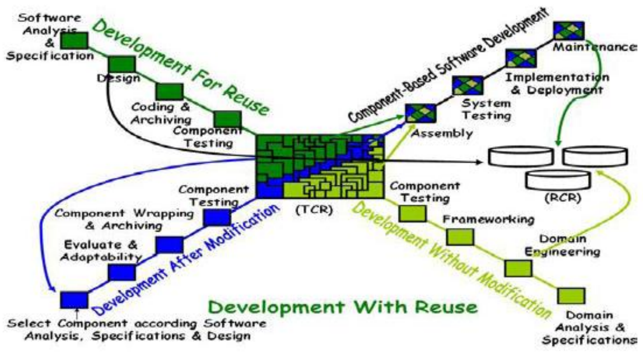
4. *W-Model*

The W model is a sequential approach to testing the product and can only be done once the product is completed, no modification is needed in between. Such tests are most suitable for short-term projects in the form of medical applications. The W model symbolizes one-to-one relationship between documents and test activities. The use of this model helps ensure that the test of the product starts from the first day of product development. This model is known for dealing with problems that cannot be solved using V-model.



5. *X-Model*

In this X model, there are procedures required engineering and requirement started by specification. The main feature of this software life cycle model is reusability, in which the software has been developed by reusable and testing able components with software development and the creation of reusable components for software development. In software development, it uses two main approaches, develops software components for reuse and software development with or without revision in the reusable component


Software Analysis & Specification
Development For Reuse
Design
Coding & Archiving
Component Testing
Component-Based Software Development
Maintenance
Implementation & Deployment
System Testing
Assembly
(RCR)
(TCR)
Component Testing
Component Wrapping & Archiving
Evaluate & Adaptability
Development After Modification
Development Without Modification
Frameworking
Domain Engineering
Development With Reuse
Select Component according Software Analysis, Specifications & Design
Domain Analysis & Specifications

VI. Factors Affecting Cost of Component Based Software Development

1. *Identity and Acquisition Cost*

Before searching for reusable components, the manufacturer should develop a complete description of the product and requirements from the consumer. Depending on these requirements, the developer develops components to reuse them in the future. The manufacturer may also be required to modify the existing process. Component Identification and Acquisition Costs reflect the cost of mining and reusable assets, including the necessary efforts to find the

International Journal of Scientific Engineering and Science
Volume 2, Issue 1, pp. 26-32, 2018. ISSN (Online): 2456-7361

right properties, With the necessary efforts to search for the appropriate assets, whether in other areas of the organization, being present in the reusable component repository, such expenses in the public domain or the market would include asset hearing, verification, and subsequent purchases. This is the factor which affects the cost

2. *Modification Costs*

During the modification, there are two modes in conversion of repository properties:
(1) Optimization for reuse, which is an amendment of an existing repository property.
(2) White box reuse, in which the amendment to the property of the same asset has been included in the same application.
Property modification costs include:
(1) Black box software has been reused to integrate the reused component into a new product to make additional effort to modify the white box, the second product.
(2) Reusable components and all the features of all features make the interface between work, application and reusable component, they should be identified and properly specified to obtain the desired functionality.

3. *New Development Costs*

The cost of developing new properties is included in the software reuse, these costs fall into two categories: Manufacturers and consumer producers create a new treasure house with scratches, in which the property will be in line with the specific standards which allow reusability Give up. , And consumer components include reusable assets in software components. Consumers can develop new components necessary to integrate reusable software into their applications.

4. *Integration and Testing Costs*

Product integration costs include: 1) Cost of partial and complete integration; 2) the cost of data transfer from the previous application for new applications to verify and verify any component in the reuse of reuse; And3) The cost of review of the design, which are necessary to review and summarize a document, the integration costs include verification and verification activities (i.e., technical design reviews, formal code walkthrough and unit test plans), which can be used for new components Costs for costs, as well as costs of direct costs, along with the cost of coding

5. *Infrastructure Costs*

Before installing the reuse program, it is necessary to reuse a new development process and establish a reuse shop. The cost of setting up and maintaining treasures include: 1) database analysis and design; 2) Cost of equipment development or purchase, whose new technology textbook or online training is required; And 3) cost of the database administration costs and inventory reserves included in artifacts; 1) the cost of time required for the approval of artifacts for the treasure; 2) The cost of analyzing the metadata necessary for employing efficient discoveries of artifacts in the list; And 3) the cost of a mechanism to recover property from the list. In the storage phase, the manufacturer should classify and store the property, which will be kept in a repository for consumer recovery.

VII. Component Software Proposed by Different Industries

Since the reuse of product business and the potential impact of CBSE is very large, various important organizations and industry associations have proposed standard for segment programming:
(A) OMG / Corba: Object Management Group has distributed a specific resistance to dealer engineering (OMG / COBRA). To ask a representative, a question (ORB) gives a classification on the administrations that empowers the reusable parts to speak with different segments, keeping a little of your area within a framework Happened. At that point, when the clauses are manufactured using the OMG / CORBA standard, coordination of those parts (without adjusting) within the frame is guaranteed, if an interface definition bid (IDL) interface is created for each segment Using client / server imagery, ask questions within the client application for at least one administration from the ORB server. The solution is done through an IDL or is powerful at run time. There are all the basic statistics about the demand and response status of administration in an interface store.
(B) Microsoft Comm: Microsoft has created a Segment Exposure (CAM) that decides to use the regions distributed by different merchants within a single application running under the operating framework of Windows. COM contains two components: an arrangement of the system for encrypting and passing messages between COM interfaces (actual as COM objects) and COM interfaces. From the perspective of the application, "There is no emphasis on how [COM objects] is executed, in the same manner that there is an interface to the opposition, which registers it with the structure, and it will talk to others For partial framework uses COM objects. "
(C) SUN JavaBeans Components: JavaBean part framework is a convenient, stage-independent CBSE Foundation to use Java programming bidding. The JavaBean Framework has spread Java Applet 4 so that more sophisticated programming segments required for part-based improvements can be completed. The JavaBean section structure includes an arrangement of tools called Bean Development Kit (BDK), to help designers break (1) existing beans (Segment) works, (2) changes in their conduct and appearance, (3) (4) Develop custom beans for use in a particular application, and (5) test and evaluate Bean conduct for system coordination and correspondence.

VIII. Impact on Quality, Productivity and Cost

There is a natural interest in the component-based programming building. In theory, product collaboration should be presented with favourable conditions in the quality and good luck. Apart from this, they should be converted into cost investment funds. In any case, there are difficult information that are helping in our instinct? In order to answer this inquiry, we should first understand that what can actually be reused in the setting of a product creation and after that there are expenses associated with the actual reuse. As a result, it is worth the imagination to create a cost / profit test for reuse.

(1) Quality

In a correct setting, the part of a product created for reuse will be confirmed and there will be no distortion in it. As a general rule, formal confirmation is not completed on a regular basis, and incompleteness can occur and may happen. It may be that with every reuse, surrender is found and it is settled, and the quality of one segment increases accordingly. After some time, this section turns out to be distortion of all the objectives and objectives. In the investigation led by Hewlett Packard, Lim reported that the distortion rate of the Reliance code is 0.9 incomplete for each KLEC, whereas the recently created programming rate is 4.1 incomplete for each KLOC. For an application that was carried out 68 percent re-code, there was a sarcastic rate of 2.0 flaws for each key-51 percent change at a normal rate, the application was made as a non-re-used product. Henry and Faller [Hen 9 5] reports 35 percent variation in quality although the outstanding report crosses a sensible wide range of price change rates. It is appropriate for the expression that a non-profit profit is given in the form of quality and trustworthy for reused programming.

(2) Productivity

When reusable parts are added through all the product process, then there is less time spent for planning, model, report, code and information needed to create a deliverable framework. After this, it takes that the same level of utility is expressed to the customer with less information. After this, the efficiency is progressing. Despite the fact that the efficiency of the rate is quite hard to interpret changes reports, it creates the impression that 30 to 50 percent reuse can bring profitability up to 25 to 40 percent.

(3) Cost

The net cost investment funds are assessed for reuse that the cost of the venture is being pegged at a closed end, so that it was made without preparation, CS and after that reuse, CR and real To reduce the related expenses completely, the cost of the said product, CD CS can be controlled on implementing at least one of the estimated methods. Reuse related expenses, CR, included

a) Domain check and display
b) Domain engineering improvements
c) To increase the documentation to encourage reuse
d) Reuse Segment Support and Upgrades
e) Royalties and licenses for remote receiving segments.
f) Creating or receiving re-experiment collections and operations.
g) Task Force training in the framework and development for reuse.

Despite the fact that expenses related to the operation of the space test and reuse shop can be very important, keeping in mind the remarkable alternative costs here, which is a piece of good programming creation practice, again Whether to use or not.

IX. Comparison of CBSE with Traditional Software Engineering

S.no.	Attribute	Component Based Software Engineering	Traditional Software Engineering
1.	Cost	It is in the software creation software to rebuild the cost so that cost can be reduced.	No reuse of components and therefore there is no low cost
2.	Development Time	Due to the use of pre-built software components, the time of development has decreased.	Pre-built components are not used and therefore there is no shortage of time during development.
3.	Quality	Software components that are used for software creation, display quality, characteristics such as performance, reliability and applicability, thus increasing quality.	No reusable components are used here and therefore no increased quality is achieved.
4.	Applicability	This method of software development applies only to software with pre-built software components.	There is no such restraint with traditional software engineering

X. Software Reusability

Component-based software development is established on the concept to advance software systems by selecting relevant off-the-ledge factors and then to combine them with a well-defined software structure. This concept gives the perception on reliability.

1. *Idea of Recyclability*

Recyclability engages an indispensable job in component based software development and also precedes the base for CBSD. The elementary standard for assessing component is recyclability. If an element is not recyclable then the entire idea of component based software development becomes unsuccessful.

2. *Aspects that Influence Recyclability*

Aspects in the manner of complication, customizability, flexibility, interface, attestation, standard, recognizable, versatility, efficiency, condition, expenses, configurability and consistency influence the process of recyclability.

3. *Ease of Software Recyclability*

The chief ease of software recyclability are come behind

a. Boost standard
b. Definitive
c. Reduce expenses
d. Accumulating efficiency
e. Interoperability

XI. Divergent Variety Of Recyclability

1. *Impromptu Ad-hoc Recycles*

Ad-hoc reuse is favoured when recycle happen inside undertakings. Bidirectional trade is been done between the application gatherings.

2. *Storehouse Based Recycle*

Repository Based recycle is done when segment vault is utilized and can be gotten to by different application gatherings. It depends on amount in light of the fact that any number of segments can be put into the store and there is no influence over their quality and convenience. Here the archive is the trade medium between the application gatherings.

3. *Concentrated Recycle*

In Centralized recycle part bunch is unequivocally in charge of vault. The gathering figures out which parts are to be put away in the store and guarantees the nature of these segments and the accessibility of fundamental reports and hence helps in recovering reasonable segments in a specific reuse situation. Here every application assemble associate with the part gathering, assist they are associated bidirectional with the archive.

4. *Space Based recycle*

In Domain Based recycle every area assemble is in charge of segments in its space e.g. System parts, UI segments and database segments. Here every application aggregate is related with its area gathering. Space gatherings can additionally connect with other application assemble also. The area bunches are joined with the vault.

Accuracy is increased since the components have previously been tested in various contexts and recyclability reduce the development time. A portion of the rules are proposed to improve the nature of the product. These rules help in accuracy enhancing quality and efficiency of associations embracing CBSD and proposed a metric suite for measuring the recyclability of such black box components based on limited information that can be gotten from the outside of parts with no source codes. Five measurements have been characterized for measuring a parts understand ability, flexibility and convey ability. Recyclability metric is given by combining proposed metrics based on a recyclability model. This can effectively identify black box component with high recyclability.

a) Black Box Testing:-It is a technique in which the tester doesn't know the internal working of the product being tested.
b) White Box Testing:-It is a technique in which the internal functioning of the product is tested.

XII. Recycling Programming Segments

Recyclable programming clauses are not new exercises in any case, for example, the Microsoft Com + Enterprise Java Beans and Corbie Segment are reaching more distances due to the development of the model. Many organizations now rehearse programming recycle after collecting new segments or data frameworks by collecting the previous segment (across areas or beyond). Pointing to implementing segments or re-refreshing programming frameworks that are already created or used by existing resources. The reuse of programming is something that has increased the idea for a long time on board of programming engineers, but has been neglected to be fully read for a critical degree. Fortunately, the segment-based programming improvement has clearly been repetition and it now clears the way for recycle benefits to be gathered by the institutions. In that capacity, the benefits of recycling of programming parts in segment-based progress are divided into segments together.

XIII. Analysis and Design for Reuse

Data can be used to illustrate useful and practical models (talked in detailed documents) how to get a specific application. The details prepared for painting these models are used, the total depiction of anything else is the result. In an ideal world, test performance is done to determine those components of those models, which indicate the current reusable segment. This problem seeks to remove data from those shapes that can signal "determination coordination". Belinzoni, Gagini, and Perencei painted an approach to the crossroads of the opposing system: On different levels of reflection, the ingredients are specially marked as special, plan and use classes - with each category, has been produced. Information about special education-upgradation is kept in the form of re-recommendation sections, in which bearings are used to retrieve reusable areas on the basis of their depiction and to make and fit them after recovery.

XIV. Various Key Issues that Frame a Reason for Outline for Reuse

Standard information:

The application area should be investigated and standard information structures (for example, document structures or a complete database) should be identified throughout the standard world. Then all outline sections will be described for using these standard information structures.

Standard Interface Conventions:

Three levels of interface conference should be created: the nature of intra-mary interfaces, plans for external specialization, and human / machine interfaces

Program Layouts:

The structure model can fill the form of the format of engineering program of another program.

XV. Advantages of Programming Recycle

As said in the first segment, there are many advantages of reusing programming parts in data frameworks improvement. At the point when accurately connected and executed, reuse can increment productivity, shorten time-to-showcase, enhance programming quality, decrease upkeep cost, consider between application interoperability, diminish dangers, use specialized abilities and learning, and enhance framework usefulness.

Aside from profitability picks up, segment recycle enable associations to diminish the basic way in the conveyance data frameworks applications, lessening an opportunity to-market and start to collect benefits prior. With appropriate arranging of segment interfaces and framework plan, distinctive advancement groups at various areas can build up their own parts simultaneously. Besides that, product framework can gather segments crosswise over limits at run time or configuration time which energizes circulated programming improvement.

The nature of data frameworks created utilizing this approach will likewise have less bugs and imperfections if contrasted and recently worked starting with no outside help frameworks.

From a cost point of view, if an advantage's expenses can be amortized through an expansive number of employments, it would then be workable for the administration to use more exertion and dispense more spending plan to enhance the nature of programming segments. This thusly diminishes the level of hazard looked by the advancement exertion and will evidently enhance the probability of accomplishment.

Keeping up heritage frameworks is a bad dream for each organisation. Almost 80% of programming improvement costs are utilized to keep up the frameworks after they have been implemented. Therefore, one noteworthy favourable position of a part is its attachment and play include which permits simple arrangement and incorporation in the data frameworks exertion. Associations can discard undesirable segments and amass them with further developed segments in view of their needs without influencing the elements of different parts.

When frameworks are produced utilizing reused components, they are relied upon to be more interoperable as they depend on regular systems to execute the vast majority of their capacities. Discoursed and interfaces utilized by these frameworks would be comparable and would enhance the expectation to absorb information of clients who use a few distinct frameworks manufactured utilizing similar segments. Programming recycle in CBSD likewise enables authorities to improve the product parts and the segment based advancement design being created which could then be recycled by different engineers whose fundamental undertakings meet the item would include needs and the required usefulness as determined by the users. Hence, it is urgent that associations actualize the right methodologies to ensure the appropriation and proceeded with utilization of recycled programming segments in a precise way.

XVI. Conclusion

In this paper, different concepts of component based systems are studied for the software development. The criterion was to study the papers related to qualification for the CBSD. In the review of the entire Paper, it considers the re-usable component at the time of selection, one of the most indecisive factors is considered. Reusability plays an important role in CBSD, helping to regain a particularly suitable component. And also serves as the basis for CBSD. The original criterion re-use scenario here is interacted with each app group. Reusability for evaluation of component if a component is a group, then they are connected to two if it cannot be used again, then the entire concept of the component has been guided with the repository software development fails. In this paper it is discussed about the various CBSD Models. CBSE is still an emerging field in software engineering and there is plenty of place for research in this field. Although there are advantages of CBSE, but there are also the components of maintenance cost, changing requirements (project specific requirements), disgruntled requirements, stock management and component management of the component etc.

XVII. Acknowledgment

We want to thanks Prasenjit Banerjee and Dr. Parul Gandhi for their constant support and advices during the writing of this paper.

References

[1] M. Kaushik and M. S. Dulawat, "A comparison between traditional and component based software development process models," *J. Comp. & Math. Sci.*, vol. 3, issue 3, pp. 308-319, 2012.

[2] A. Irshad Khan, Noor-ul-Qayyum, and U. Ali Khan, "An improved model for component based software development," *Software Engineering*, vol. 2, issue 4, pp. 138-146, 2012.

[3] I. Crnkovic, "Component-Based software engineering – New challenges in software development," *Journal of computing and information technology – CIT 11*, vol. 3, pp. 151-161, 2003.

[4] S. Thakral, S. Sagar, and Vinay, "Resuability in component based software development," *World Applied Sciences Journal*, vol. 31, issue 12, pp. 2068-2072, 2014.

[5] H. M. Haddad, N. R. Ross, and W. Kaensaksiri, "Software reuse cost factors,".

[6] W. K. Yen, G. G. Guan Gan, and M. Toleman, "Challenges and strategies for software component reuse in information systems development: A Review,".

[7] A. Sharma, R. Kumar, and P. S. Grover, "A critical survey of reusability aspects for component-based systems," *World Academy of Science, Engineering and Technology, International Journal of Industrial and Manufacturing Engineering*, vol. 1, no. 9, 2007.

[8] http://www.engpaper.com/free-research-papers-software-engineering-component-based-development.htm

[9] G. T. Heineman and W. T. Councill, *Component Based Software Engineering*.