iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 9 **Issue:** VIII **Month of publication:** August 2021

DOI: https://doi.org/10.22214/ijraset.2021.37632

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue VIII Aug 2021- Available at www.ijraset.com

Component Based Software Engineering

Shubh Shah[1], Vyom Shah[2]
[1, 2]*Computer Science & Engg, University, Ahmedabad, India*

Abstract: ***The central idea of Component Based Engineering is to develop a system software by selecting the well defined software components not used often and assembling them with certain system architecture. Nowadays the software development pattern is far different from the earlier approach as many new concepts are being taken into consideration E.g. QA (QualityAssurance). This term paper includes a detailed description of all the current component based software techniques used as well as their advantages and disadvantages. We also address the quality assurance issue of component based software engineering.***

I. INTRODUCTION

In the present scenario, the demand for new software development patterns is highly increasing and the main reason for that is that modern software development systems have many disadvantages like high production cost, complex and difficult to control and unmanageable software quality. The most preferred solution to this problem is component based software development. The main idea behind this model is by selecting appropriate components from the repository and combining them with a well defined architecture. This component based software model can only be implemented in scratch and this is the main difference between earlier approach and the present approach. Developers can develop this commercial off the shelf components(COTS) using different languages on different platforms. The main advantage of Component Based Software Development(CBSD) is that , it can reduce development cost and time-to-market, maintainability, reliability and overall quality of and improve software systems. Due to this the amount of interest research community and software industry has increased gradually. The main focus of CBSE is that lifecycle of CBSD is far different from that of traditional approach.

The concept of Component Based Software Engineering is explained briefly in the following diagram.

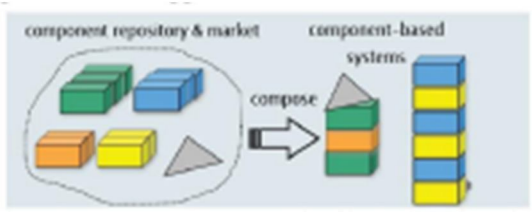
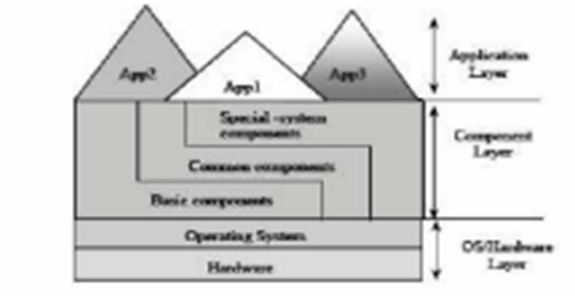


Figure 1. Concept of Component-based software engineering

So, here the most important word is 'Component', however there is no specific definition for this word. But instead of this there are certain features that a component must have. Following are the features which a component should have:

1) A component is an independent and replaceable part of a system that fulfills a clear function.
2) A component works in the context of a well defined architecture.
3) The communication among components is through it's interfaces.[1]

The most important factor for a component based software system to run properly and effectively is system architecture. According to research as well as industry the system architecture should follow layered and modular architecture. The architecture to be followed is shown below.



Here the second layer consists of components which are engaged in some specific domain or say application domain. Also there are some components which are useful in more than one domain. The third layer consists of cross-business middleware components which have common software and interfaces to other established entities. And finally the lowest layer consists of basic components that have an interface with the hardware components and operating system.

There are many component technologies which have been used to implement different software systems such as object oriented distributed components and web enterprise based applications.

Some commercial companies like IBM,BEA, Microsoft,Sun are also involved in the software component revolution.

II. INTERFACE STANDARDS USED IN COMPONENT BASED ARCHITECTURE

As the primary objective of CBSE is component reusability, certain frameworks like COM/DCOM, JavaBean, EJB, CORBA, .NET, web services, and grid services use CBSE. These types of services are used to provide reliable coordination and communication for the application of the software. These frameworks help in communication between the components of a software application.These technologies are widely used in local desktop GUI application design such as graphic JavaBean components, MS ActiveX components, and COM components which can be reused by simply drag and drop operation.[2]

A. Component Object Model (COM) and Distributed COM

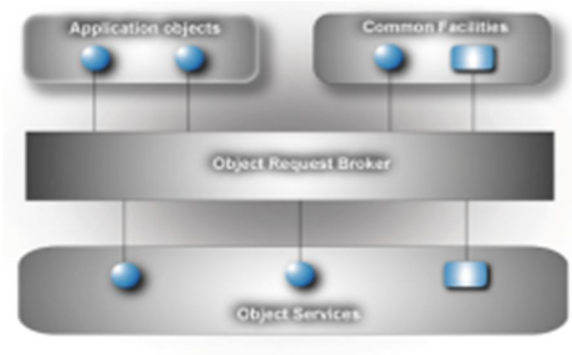
This interface standard was designed by Microsoft in 1993. The COM provides platform dependent component based software engineering. COM and DCOM are generally used in windows. They are also used in language-independent component based applications.

COM defines interaction of the clients with components. No intermediate component is required for interaction of component and client. Dynamic interoperability is maintained by using binary standards between components and clients.

Distributed COM is a protocol that provides a network for communication between client and components. This network is reliable, secure and efficient. DCOM is used in multiple network transports, including Internet protocols like HTTP.

B. Common Object Request Broker Architecture (CORBA)

CORBA is an open standard for application interoperability that is defined and supported by the Object Management Group (OMG)[3]. It is known to be the world's leading middleware solution for enabling the communication between the components of softwares. It is a design specification of Object Request Broker (ORB). ORB provides mechanisms for software components to communicate with each other.



It is often known as ‘Software Bus’. The diagram above gives a brief explanation of the function performed by CORBA. Data communication between client and server takes place with object oriented services. ORB takes care of the location of the target sends requests and receives responses from the caller. The caller does not need to know the location of the target as it is done by the ORB. CORBA is generally used in distributed systems which include component based software engineering.

C. JavaBean and Enterprise JavaBean

The classes that encapsulate many objects into a single object are known as javabeans.

The problem of reliability and security can be solved by using the java platform . Java provides certain applications including the interoperating across multi vendor servers, propagating transaction and security contexts, servicing multilingual clients and supporting ActiveX via DCOM/CORBA bridges.

D. Web Services and Grid Services

Web services are the services used by internet based applications and also for the communication between client and server. Certain difficulties are faced while using web services. To overcome these difficulties grid services were introduced.Grid services are web services with improved characteristics and services. These are generally used in Component Based Software Engineering.

III. QUALITY ASSURANCE FOR COMPONENT BASED SOFTWARE

A. The Life Cycle of Component Based Software Systems:

The main difference between component based systems and normal systems is that component based systems are developed by choosing various components and joining them together rather than programming an overall system from the beginning. Due to this, the life cycle of component based systems is different than that of simple systems. To summarize the life cycle of component based systems we divide it into 5 parts 1)Requirement Analysis 2)
Software Architecture selection, construction, analysis and evaluation 3) Component identification and customization 4) System Integration 5) System Testing 6) Software Maintenance.

Software architecture is defined with respect to the computational components and interaction among that components. The main aim is to formulate and assemble components which demand separate and independent development.Component identification , customization,and integration is heart of the life cycle of component based software development. It mainly includes 2 sub parts

1) To evaluate each candidate COTS component based on their functional and quality requirements which are used to assess that particular component.

2) To customize those candidate COTS components that are to be updated before integrating them into new component based software systems. Integration is defined as taking key decisions on providing communication and coordination among various components of the target software system.

Quality assurance for component based software systems should verify it's life cycle and it's key activities to analyze the components and achieve high quality component based software systems. Due to differences between characteristics of component and traditional software some quality assurance technologies are still premature.

B. Quality Characteristics of Components:

Quality assurance for component based software systems should verify it's life cycle and it's key activities to analyze the components and achieve high quality component based software systems. Due to differences between characteristics of component and traditional software some quality assurance technologies are still premature.

Though less work is done in the field of component based software development , QA technologies are expected to address 2 inseparable parts

1) How to certify quality of component?

2) How to certify the quality of a software based on components?

	CORBA	EJB	COM/DCOM
Development environment	Underdeveloped	Emerging	Supported by a wide range of strong development environments
Binary interfacing standard	Not binary standards	Based on COM; Java specific	A binary standard for component interaction is the heart of COM
Compatibility & portability	Particularly strong in standardizing language bindings; but not so portable	Portable by Java language specification; but not very compatible.	Not having any concept of source-level standard of standard language binding.
Modification & maintenance	CORBA IDL for defining component interfaces, need extra modification & maintenance	Not involving IDL files, defining interfaces between component and container. Easier modification & maintenance.	Microsoft IDL for defining component interfaces, need extra modification & maintenance
Services provided	A full set of standardized services; lack of implementations	Neither standardized nor implemented	Recently supplemented by a number of key services
Platform dependency	Platform independent	Platform independent	Platform dependent
Language dependency	Language independent	Language dependent	Language independent
Implementation	Strongest for traditional enterprise computing	Strongest on general Web clients.	Strongest on the traditional desktop applications

Table 1. Comparison of current component technologies

So firstly to evaluate a component we must determine how we can rate the quality of that particular component. The quality characteristics of components are based on guaranteeing the quality of components. The list of recommended characteristics for quality of the components are : 1) Functionality 2) Interface 3) Usability 4) Testability 5) Maintainability 6) Reliability.
Software metrics should be proposed to find various features like size, complexity, reuse frequency and reliability. Let's discuss in brief about each feature.

a) *Size:* Change in size leads to change in both reuse cost and quality. Small size can lead to loss as the benefits would not exceed cost of managing whereas large size would result in compromising the quality.
b) *Complexity:* This also leads to change in resume cost and quality. Components with low complexity are not profitable to reuse whereas components with high complexity result in low quality.
c) *Reuse Frequency:* Number of occasions when a component is used says all about it's usefulness.
d) *Reliability:* It can be determined by failure free operations under some tough determined scenarios and situations.

IV. A QUALITY ASSURANCE MODEL FOR COMPONENT BASED-SOFTWARE SYSTEM

As we all know that the logic behind component based software engineering is different from that of traditional software engineering concept, it's quality assurance model should address both the process of components and process of overall systems. In this section, we discuss about a quality assurance model for the component-based software development paradigm. The different phases related to components and system in this model are

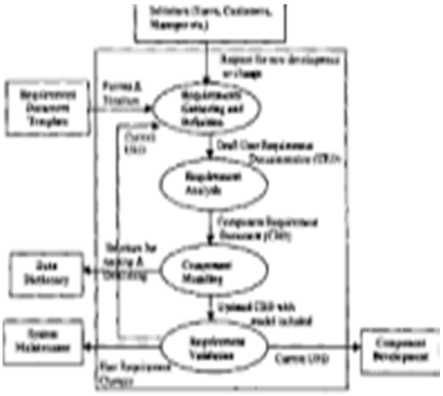


Figure 4. Component requirement analysis process overview

1)Component Requirement analysis 2)Component Development 3)Component Certification 4)Component Customization 5) System Architecture Design 6) System Integration 7) System Testing 8) System Maintenance.

A. Component Requirement Analysis

The process of discovering, understanding, documenting, validating and managing the requirements for a component is called component requirement analysis. The main objective of this phase are to produce complete, consistent and relevant requirements that a component should realize. With that the objective also is too produce complete, consistent and relevant information regarding the programming language and other interfaces too. The overview of the component required diagram is shown in the figure below. Starting with the request of users or customers for updating or changes in the old system, the component requirement analysis consist of 4 main steps.1)Requirement Gathering and Definition 2) Requirement Analysis 3) Component Modeling 4) Requirement Validation. After following 4 steps we will get current user requirement documentation which should be transferred to the next development phase. Fllowing 4 steps we will get current user requirement documentation which should be transferred to the next development phase .

B. Component Development

The process of implementing the requirements for a well functional high quality component with multiple interfaces is known as component development. Final component products , the interfaces and development documents are the main objectives of the component development[4].The final outcome should lead to final components satisfying the requirements with the expected results, well defined behaviour and flexible interfaces.

The overview of the component development process is shown in the figure below.

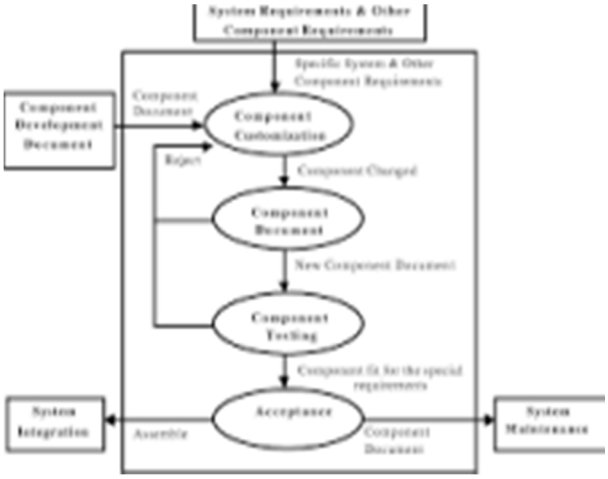
There are a total 4 procedures in the component development process named as 1) Implementation 2)Function Testing 3) Reliability Testing 4) Developed Document.

The input in this phase is the output of the previous phase i.e. component required document. The output of the following phase should be developed components and it's documents ready for the following phases of component certification and system maintenance respectively.

C. Component Certification

Component certification involves certain processes like component outsourcing i.e to manage component outsourcing contract and to audit the performance of the contract, component selection which involves selection of the right component according to its performance and reliability, component testing which confirms that the selected component is perfect according to the requirements with acceptable quality and reliability.

The objective of component certification is to check whether the candidate components satisfy the system requirements with high quality and reliability. Certain governing policies include : Software contract managers should charge for component outsourcing. All candidate components that are tested should be free from all defects. Testing should always be done in a targeted environment or a simulated environment. The component certification can be shown in the below figure.



D. Component Customization

There are 3 processes involved in component customization. Following are the 3 processes involved 1) To modify components for the specific requirement 2) doing necessary changes to run components on a special platform. 3) To upgrade specific component for better performance and higher quality The objective of component customisation is making necessary changes for a developed component for using it in a specific environment and to cooperate with other components to. The component customization overview diagram is shown in the below figure. All the components must necessarily be customised according to the operational system requirements or there is an other option for customisation according to the

Interface requirements with other components in which components should work. The input to the component customisation is output of the previous phase i.e. system requirement, component requirement and the component development document. The output of the following should be the customized component and document for system integration and system maintenance i.e. for input of the next phase which is system architecture design.

E. System Architecture Design

Evaluation, selection and creation of software architecture of component based systems comes under System Architecture Design.

The main objectives of system architecture design are collection of user requirements, identifying system specification, selection of appropriate systems and determining the implementation details.

The outcome of system architecture design is the selection of a particular architecture from other architecture.The phase shown in the figure.

This phase consists of requirement gathering, analysis, system architecture design, and system specification.

F. System Integration

The process of assembling the components selected into a whole system under the designed system architecture is known as system integration. The overview of the system integration process diagram is shown below.

The main objective of system integration is the final system i.e. the system which we expected composed by the selected components. The input to this phase is system requirement documentation and specific architecture which is the output of the previous phase. 4 steps are involved in this phase which are 1) Integration 2) Testing 3) Changing Component 4) Reintegration. The output for the following phase is the final system which will be used for system testing and system maintenance phase.

G. System Testing

The process of evaluating a system is known as system testing. The evaluation of the system confirms that the system satisfies the specified requirements and identifies the defects in the system and corrects them.

The outcome of the system testing is the final system integrated by components selected in accordance to the system requirements. This phase consists of definition, planning, design, execution, automation, Bug testing and fixing.

H. System Maintenance

The process of providing the service and maintenance activities needed to use the software effectively after it's delivery is known as system maintenance.

Providing an effective product or service to end users while correcting faults, improving software performance or other attributes and adapting the system to a changed environment are the main objectives of the system maintenance phase.

It is suggested that there should be a maintenance organisation for every software product which is in operational use. All the changes in the delivered system should be reflected in the related documents. According to outputs from all the previous phases as well as request and problem reports from the users system maintenance should be held for determining the support strategy and problem management. We can say that the output of this phase, a new version can be produced for the system testing phase for a new life cycle.

V. CONCLUSION

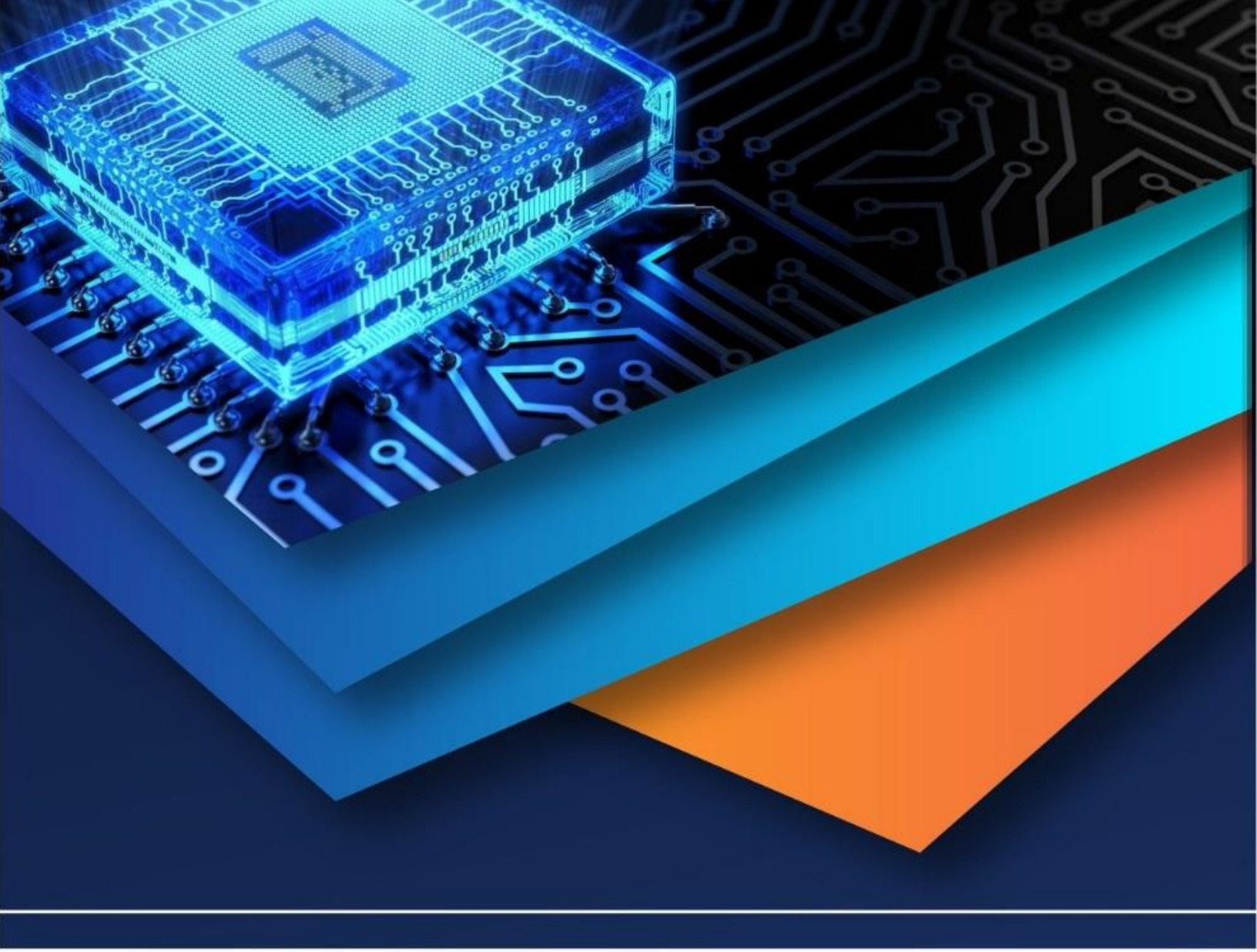
In this paper we give an enormous explanation about Component Based Engineering and its modules along with a theoretical model for quality assurance of a software using component based technologies. The use of component based models is limited to framework, certain procedures can actually be used for pure software creations. The practise of the quality assurance model can lead to a component based engineering evaluation.

REFRENCES

[1] www.cs.umd.edu › Teaching › Fall2006 › StudentSlides
[2] https://www.tutorialspoint.com/software_architect ure_design/component_based_architecture.htm
[3] https://www.computer.org/csdl/magazine/co/1995/ 03/r3068/13rRUwgQpxK
[4] https://www.geeksforgeeks.org/distributed-componen t-object-model-dcom/
[5] https://mgmdrgyp.org/mcqs/Notepad/M.Sc.%20I. T/2nd%20Sem/Soft.Engineering.html

Books used for reference
[1] George. T. Hienaman, William. T. Councill, “Putting The Pieces Together”, 2001
[2] Santosh Kumar, “Component Based Software Engineering”, 2020
[3] G. Pour, “Enterprise JavaBeans, JavaBeans & XML Expanding the Possibilities for Web-Based Enterprise Application Development,” Proceedings Technology of Object-Oriented Languages and Systems, 1999

Websites used for reference:
[1] GeeksforGeeks/CBSE
[2] ResearchGate/CBSE
[3] TutorialPoint



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY