Volume 4, Issue 8, August 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Software Model for Quality Controlled Component Based Software System

Sandeep Chopra
Asst. Professor
SGRR-ITS,
Dehradun, India

Harish C. Sharma
Asst. Professor
SGRR-ITS,
Dehradun, India

Pradeep Semwal
Asst. Professor
SGRR-ITS,
Dehradun, India

Sanjay Sharma
Asst. Professor
SGRR-ITS,
Dehradun, India

Abstract: Component Based Software Engineering (CBSE) is a branch of software engineering which emphasizes the separation of concerns in respect of the wide-ranging functionality available throughout a given software system. This practice aims to bring about an equally wide-ranging degree of benefits in both the short-term and the long-term for the software itself and for organizations that sponsor such software.
Software applications are assembled from components from a variety of sources the components themselves may be written in several different programming languages and run on several different platforms. Merely selection of qualified components is not sufficient but we have to select quality components to increase and maintain the overall quality of the software system. Software Quality Assurance (SQA) for component-based software development is a new topic in the software engineering community. In this paper, we propose a SQA model for component-based software development.

Keywords: Component based software, software quality assurance, component based software development.

I. INTRODUCTION

Component Based Software System (CBS) are mainly constructed from reusable components such as third party components and Commercial-Of-The-Shelf (COTS) components. Due to this, component based systems aredeveloped quickly with minimum engineering efforts and resource cost. Modern software systems are becoming more intricate day by day, apprehensively controlled, resulting in soaring development outlay, low productivity, insurmountable software eminence and lofty menace to budge to new technology [1]. Consequently, there is an emergent demand of searching for a new, resourceful, and cost-effective software expansion paradigm.
One of the most promising solutions in the present day is the component-based software development loom. This approach is based on the initiative that software systems can be developed by selecting appropriate off-the-shelf components and then assembling them with well-defined software architecture [2]. This new software development approach is very different from the traditional approach in which software systems can only be implemented from scratch. These commercial off-the-shelf (COTS) components can be developed by different developers using different languages and different platforms. This can be shown in Figure 1, where COTS components can be checked out from a component repository, and assembled into target software system. [12]

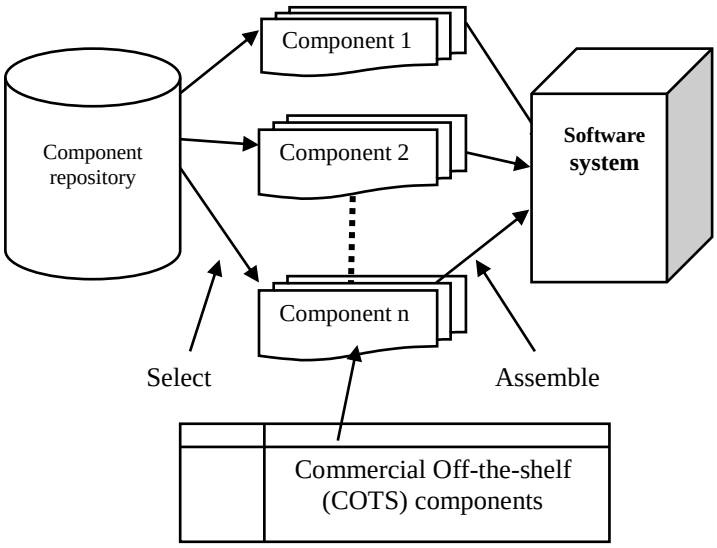
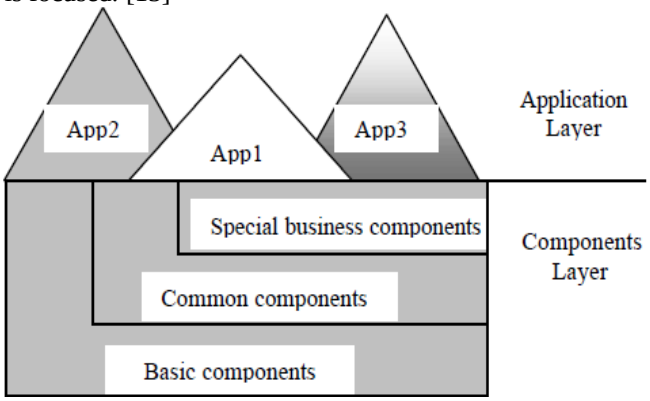


Figure 1. Component-based software development

Component-based software development (CBSD) can radically condense development cost and time-to-market, and perk

up maintainability, steadfastness and overall eminence of software systems [3] [4]. This approach has raised a tremendous sum of interests mutually in the research community and in the software meticulousness. The life cycle and software engineering model of CBSD is much different from that of the traditional ones. This is what the Component-Based Software Engineering (CBSE) is focused. [13]



Architecture of component bases software system, [13]

Up to now, software component technologies are an emerging expertise, which is far from being matured. There is no existing standards or guidelines in this new area, and we do not even have a unified definition of the key item "component". In general, however, a component has three main features:

- A component is an independent and replaceable part of a system that fulfills a clear function;
- A component works in the context of a well-defined architecture; and
- A component communicates with other components by its interfaces [5].

To ensure that a component-based software system can run properly and effectively, the system architecture is the most important factor..

Recent component technologies have been used to implement different software systems, such as Enterprise Java Beans, object-oriented distributed component software [6] and Web-based enterprise application. There are also some commercial players involved in the software component revolution, such as BEA, Microsoft, IBM and Sun [7].

II. QUALITY ASSURANCE FOR COMPONENT-BASED SOFTWARE SYSTEMS

As much work is yet to be done for component-based software development, QA technologies for component-based software development has to address the two inseparable parts:

- How to endorse quality of a Component?
- How to certify quality of software systems based on components?

To respond the questions, models should be promoted to delineate the overall eminence control of components and systems; metrics should be brought into being to gauge the magnitude, convolution, reusability and reliability of components and systems; and tools should be decided to test the existing components and systems.

To assess a component, we must verify how to endorse the worth of the component. The quality characteristics of components are the underpinning to guarantee the quality of the components, and thus the foundation to guarantee the quality of the whole component-based software systems. Here we suggest a list of recommended characteristics for the quality of components: [14]

- Functionality; Each and every component must provide some specific and defined functionality based on the system requirement.
- Interface; To interact with people, other existing components and software, existing hardware and data bases, a component must have proper defined interfaces.
- Usability; Component may access and must provide some services to/from other components, i.e., its usability must be unambiguous and clearly specified.
- Testability; A component must be traceable in terms of defects, faults, errors and bugs. Maintainability; A component must be repairable or replaceable faulty or worn-out parts without having to replace still-working parts.
- Reliability. A component must provide its intended service/services for the certain period of time to meet systems requirement. Software Quality Assurance models can be proposed to assure its quality [8] [9]. Such model often used to classify components includes:
- **Magnitude.** This affects both reuse cost and quality. If it is too small, the benefits will not exceed the cost of managing it. If it is too large, it is hard to have high quality.
- **Convolution.** This also affects reuse cost and quality. A too-trivial component is not profitable to reuse while a too-complex component is hard to inherit high quality.
- **Reuse frequency.** The number of incidences where a component is used is a solid indicator of its usefulness.
- **Reliability.** The probability of failure-free operations of a component under certain operational scenarios.

III. PROPOSED QUALITY CONTROLLED MODEL FOR COMPONENT-BASED SOFTWARE SYSTEMS

As component-based software systems are developed on a core process diverse from that of the traditional software, their quality assurance model should take in hand both the process of components and the progression of the overall system[10]. Many principles and guidelines are used to control the quality activities of software development route, such as ISO 9001 and CMM model. [11]. In this segment, we intend a framework of quality assurance model for the component-based software development archetype. The main practices relating to Components and systems in this model contain the following phases:

i) Component Requirement Analysis Based on System's identified requirements or Defined in Software Requirement Specification (SRS)

Component requirement analysis is the process ascertaining, understanding, documenting, validating and managing the requirements for a component. The objectives of component requirement analysis are to produce complete, consistent and relevant requirements that a component should realize, as well as the programming language, the platform and the interfaces related to the component.

Instigated by the request of users or customers for new development or changes on old system, component constraint analysis consists of four main steps: requirements gathering and definition, requirement analysis, component modeling, and requirement validation. The productivity of this segment is the existing abuser requisite credentials, which should be transferred to the subsequent component development phase and the user requirement changes for the system maintenance phase.[14]

ii) Search new Component from

- Newly Developed Component
- Outsourced from the third party
- Modified Component (exists in the repository
- Existing Component (exists in the repository;

This is the process of implementing the requirements for a well-functional, lofty quality component with manifold interfaces. The objectives of component development are the final component products, the interfaces, and development documents. Component development should lead to the final components satisfying the requirements with correct and expected results, well-defined behaviors, and flexible interfaces.

iii) Testing on each component

System testing is the process of evaluating a system to:

- Confirm that the system satisfies the specified requirements;
- Identify and correct defects in the system implementation.

The objective of system testing is the final system integrated by components selected in accordance to the system requirements. System testing should contain function testing and reliability testing. This phase consists of selecting testing strategy, system testing, user recognition testing, and completion activities. The input should be the documents from component development and system integration phases. And the output should be the testing documentation for system maintenance.

iv) Perform SQA Activity

- **Component Customization**
  Component customization is the process that involves
- Modifying the component for the specific requirement.
- Doing necessary changes to run the component on special platform.
- Upgrading the specific component to get a better performance or a higher quality.
  The objectives of component customization are to make indispensable changes for a developed component so that it can be used in a specific environment or cooperate with other components well.
  All components must be customized according to the operational system or the interface requirements with other components in which the components should work. The input to component customization is the system requirement, the component requirement, and component development document. The productivity should be the customized component and document for system integration and system maintenance.[10][14]
- **System Architecture Design**
  System architecture design is the progression of evaluating, exclusive and creating software architecture of a component-based system.
  The objectives of system architecture design are to collect the users requirement, identify the system specification, select appropriate system architecture, and determine the implementation details such as platform, programming languages etc.
  System architecture design should address the advantage for selecting a particular architecture from other architectures. This phase consists of structure requirement gathering, analysis, system architecture design, and system specification.

The production of this phase should be the structure specification document for integration, and system requirement for the system testing phase and system maintenance phase. [12]

v) Component Certification Process

Component certification is the process that involves:

- **Component outsourcing***:* managing a component outsourcing contract and auditing the contractor performance.
- **Component selection:** selecting the right components in accordance to the requirement for both functionality and reliability.
- **Component testing:** confirm the component satisfies the requirement with acceptable quality and reliability.

The objectives of component certification are to outsource, select and test the candidate components and check whether they satisfy the system requirement with high quality and reliability. The governing policies are:

- Component outsourcing should be charged by a software contract manager
- All candidate components should be tested to be free from all known defects
- Testing should be in the target environment or a simulated environment.

The input to this phase should be component development document, and the output should be testing documentation for system maintenance.

IV. CONCLUSION

The paper focusses on component based software model. In a component based system, the business rules establish and determine the components specification and their relation. The paper present component based software technnology. The paper presents a SQA model for component-based software development, which covers the component SQA and the system SQA as well as their interactions.

REFERENCES

[1] G.Pour, "*Software Component Technologies: JavaBeans and ActiveX," Proceedings of Technology of Object-Oriented Languages and systems*, 1999, pp. 398 – 398.

[2] G. Pour, "*Component-Based Software Development Approach: New Opportunities and Challenges*," Proceedings Technology of Object-Oriented Languages, 1998. TOOLS 26. pp. 375-383.

[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, TOOLS 31, pp.282-291.

[4] Alejandra Cechich and Mario Piattini-Velthuis, "*Component-Based Software Enginnering*", proceedings of The European Journal for the Informatics Professional UPGRADE Vol. IV , No 4 , August 2003, pp.15-19.

[5] Philip T Cox and Baoming Song, "*A formal Model for Component- Based Software*", IEEE Computer Society, Document number 07695-474-4/01, 2001, pp.304-310.

[6] I.Jacobson, M. Christerson, P.Jonsson, G. Overgaard, " *Object-Oriented Software Engineering: A Use Case Driven Approach,*" Addison-Wesley Publishing Company, 1992.

[7] Ivica crnkovic, "*Component-Based Software Engineering – NewChallenges in Software development*", journal of computing and Information Technology –CIT 11, 3,151-161, 2003, pp.151-160.

[8] C. Rajaraman, M.R. Lyu," *Reliability and Maintainability Related Software Coupling Metrics in C++ Programs*," Proceedings 3rd IEEE International Symposium on Software Reliability Engineering (ISSRE'92), 1992, pp. 303-311.

[9] S.M. Yacoub, B. Cukic, H.H. Ammar, " *A Component-Based Approach to Reliability Analysis of Distributed Systems*," Proceedings of the 18th IEEE Symposium on Reliable Distributed Systems, 1999, pp. 158 –167.

[10] Lata Nautiyal, Umesh Kumar Tiwari, Sushil Chandra Dimri, Shivani Bahuguna, "*Elite: A New Component-Based Software Development Model*", International Journal of Computer Technology & Applications, Vol 3, Issue 1, Jan 2012, pp 119-124

[11] C. Rajaraman, M.R. Lyu, " *Some Coupling Measures for C++ Programs,*" Proceedings TOOLS USA 92 Conference, August 1992, pp. 225-234.

[12] Roger Pressman, *Software Engineering: A Practitioner's Approach, Fifth Edition*

[13] Cai, Xia, Michael R. Lyu, Kam-Fai Wong, and Roy Ko."*Component-based software engineering: technologies, development frameworks, and quality assurance schemes.*" In Software Engineering Conference, 2000. APSEC 2000. Proceedings. Seventh Asia-Pacific, pp. 372-379. IEEE, 2000.

[14] Lata Nautiyal, Umesh Tiwari, Sushil Dimri & Shashidhar G. Koolagudi, "Component based Software Development- New Era with new Innovation in Software Development," *International Journal of Computer Applications (IJCA)*, vol. 51, no. 19, pp. 5-9, August 2012, *DOI*- 10.5120/8148-1655b

