International Journal of
Emerging Research in Management &Technology
ISSN: 2278-9359 (Volume-4, Issue-7)

Research Article

July 2015

An Innovative Model of Software Quality Assurance for Component-Based Software Systems

Sushil Chandra Dimri
Professor and Head, Graphic Era University,
Dehradun, Uttarakhand, India

Abstract—

In this present Era software applications are growing more complex and with more focus on reuse. A component-based system has become an important part for Software industry. The quality of product and services has become one of the most important factors that attract national and international business all over the world. Software Quality Assurance is the most important part of the software development process which can not be ignored. This paper is focusing on providing an overview for quality assurance for component-based systems. In this paper we are proposing an innovative quality assurance model for component-based development.

Keywords- Software Engineering, component-based development, software quality assurance

I. INTRODUCTION

Over the last several decades, as software systems become more the most important part of our lives but software community has faced the challenge of high development cost, and low productivity [1]. This has created a great demand for rapid and cost-effective development of large-scale, complex and highly maintainable software systems [2, 3].

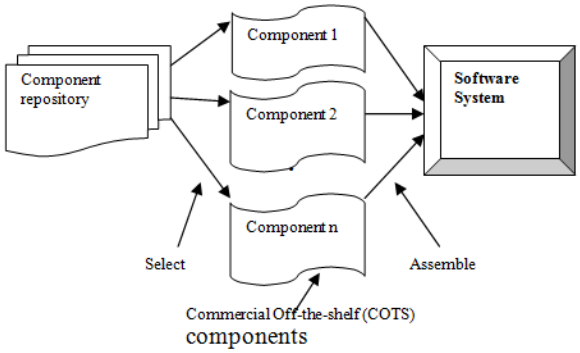

Fig 1. Component-Based Process

The most up growing solution for today's software development community is component-based software development approach. This approach is focused on the idea that developing software systems by selecting appropriate off-the-shelf components and assembling rather than implementing the system from scratch [4,5].These components can be available subsystems by internal or external sources or commercial off-the-shelf (COTS) components developed by different in-house developers using different languages and different platforms [6]. In engineering disciplines, a component is an identifiable part of a software system. A component provides a particular function or group of related functions. In programming design, a system is divided into components that in turn are made up of modules [7, 8]. As per the object-oriented programming, a component is a reusable program building block that can be combined with other components in the same or other computers form an application [9,10]. Examples of a component include: a single button in a graphical user interface, a small interest calculator, an interface to a database manager.
In general, a component has three main features [11, 12]:

1) A component is an independent part of a system that fulfills a clear function;
2) A component works in the perspective of a well-defined architecture;
3) It interacts with other components by the interfaces.

1.1 Excellence Attributes of Components

Till date there is a lot of work has been done on the development of component-based software. Even then still important attributes are missing in Component-based systems.
To analyze various different components and systems which are already existed we should know the quality attributes of the components. These important attributes are as follows.

1. Reliability

The basis for the definition of reliability is the probability that a system will fail during a given period. One definition is that reliability is inversely proportional to this probability as mean time to failure (MTTF) and it is defined for a specified period under stated conditions.

2. Availability

Availability is defined as the probability of a module being available when needed. Basically it is defined as the mean time to failure divided by the mean time between failures (MTBF), which in turn is the sum of the MTTF and the mean time to repair (MTTR).

3. Safety

Safety is a attribute involving the interaction of a system with the environment and the possible consequences of the system failure. It is a system property, neither a component nor an assembly property. Safety depends on where and how the system is deployed.

4. Confidentiality

A system would not be dependable if unauthorized access or, even worse, unauthorized alterations of the system data, were easy or even permitted. Confidentiality is defined as a measure of the absence of unauthorized disclosure of information.

5. Maintainability

Maintainability is defined as the capacity to undergo repairs and modifications. Maintenance is of three different types, corrective, adaptive and perfective Maintenance. Corrective maintenance is the removal of faults, thereby improving the reliability of components under the condition that no new faults are introduced. Adaptive maintenance is performed when a component is modified to meet.

II. SOFTWARE QUALITY ASSURANCE

In terms of software engineering, **software quality** refers to the degree of conformance to explicit or implicit requirements and expectations. Software functional quality reflects how well it complies with or conforms to a given design, based on specifications [13, 14]. It can also be described as the fitness for purpose of a piece of software or how it compares to competitors in the marketplace as a worthwhile product. Conventionally quality is defined as conformance to requirements, and failures arise when the software is not met the user requirements [15, 16]. Software Quality factors involves

1. Correctness
2. Reliability
3. Efficiency
4. Integrity
5. Usability

Software Quality Assurance is defined as a set of activities for ensuring quality in software engineering processes the activities establish and evaluate the processes that produce products. Quality Assurance focused on both the product and the process [17, 18].

Software Quality Assurance is also defined as a process of evaluating the quality of a product and maintaining the appropriate standards and procedures. It is an umbrella activity that ensures conformance to standards and procedures throughout the SDLC of a software product [19, 20].

Quality assurance activities include:

1. Administration
2. Documentation
3. Standards and Practices
4. Testing
5. Tools, Techniques and Methods

III. PROPOSED SOFTWARE QUALITY ASSURANCE FOR COMPONENT-BASED SOFTWARE SYSTEMS

Quality Assurance for component-based software systems should always addresses the main activities to analysis the components and achieve high quality component-based software systems. A quality Assurance technology for component-based software systems is currently not in much use because of the specific characteristics of component systems from traditional systems. The proposed model consists of following important stages.

1) Selecting the components for certification
2) Apply functional testing on the component
3) Modifying the component if not fulfilling the requirements
4) Certification Process
5) Performing components integration with other components
6) Finally to provide certification

The aim of component certification process is to outsource, select and test the components and check whether they satisfy the system requirement or not. Functional testing is the main activity of evaluating a system to fulfill the two following requirements:

1) Confirm that the system satisfies the given requirements;

2) Find out the defects and correct defects before the implementation process.

Functional testing is one of the important activities of certification process. This process consists of selecting testing strategy along with the number of team members involved. And if the component fulfills all the requirements then finally certification is provided for that particular component.

Apart from the Functional testing, structural Testing is applied on some parts of our proposed quality assurance model. Various testing activities will be included.

- **Unit testing.** Testing of the individual component using both black-box and white-box type tests.
- **Regression testing.** Testing to determine if changes to the software or fixing a defect cause any problems to other components in the system.
- **System Integration Testing.** Testing done to ensure that the entire product functions as intended and to specifications.

The component certification process overview diagram is as shown in Figure 2

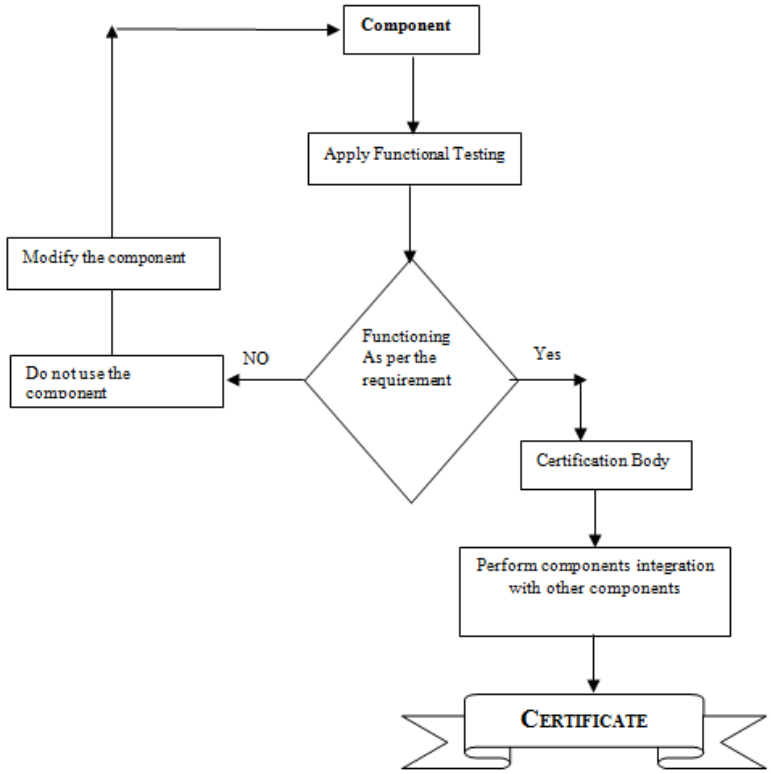

Fig 2. Certification process of a component

IV. CONCLUSION

Component-Base Software Development is a new approach of software development, which has potential to reduce cost and time-to-market, and improve overall quality of application. Quality Assurance is very important for component-based software systems, especially when the components come from different developers by using different tools and technologies. In this paper, we have tried to give an innovative new model to support the Quality Assurance. In future we would like to implement our model in practical manner to make sure that the factors work in true way.

REFERENCES

[1] Bertrand Meyer, Christine Mingins, and Heinz Schmidt, "Providing Trusted Components to the Industry," IEEE Computer, vol. 31, no. 5, pp. 104–105, May 1998.

[2] Bertrand Meyer, "The Grand Challenge of Trusted Components," Proc. 25th Intl Conf on Software Engineering (ICSE 2003), Portland, Oregon, USA, 03–10 May 2003. IEEE Computer Society Press, 2003, pp. 660–667.

[3] Szyperski C., (1998). Component Software, Beyond Object-Oriented Programming, ACM Press, Addison-Wesley, NJ.

[4] M. Sitaraman and B. W. Weide, "Special Feature Component-Based Software Using RESOLVE", ACM SIGSOFT Software Engineering Notes 19, No. 4, 21-67, October 1994.

[5] Ian Sommerville, Software Engineering. 9th Edition, Addison-Wesley, March 2010.

[6] IEEE Std 610.12–1990, IEEE Standard Glossary of Software Engineering Terminology, IEEE Standards Board, 28 Sept 1990.

[7] Jerry Zeyu Gao, H.-S. Jacob Tsao, and Ye Wu, Testing and Quality Assurance for Component-Based Software. Artech House Inc., September 2003.

[8] Brown, Alan W., Wallnau, Kurt C. (1998): The Current State of CBSE. IEEE Software Journal, September/October 1998, pp. 37-46.

[9] Han, Jun (1998): Characterization of Components. In proceedings of International Workshop on Component-Based Software Engineering, 1998.
[10] Manfred Broy, Anton Deimel, Juergen Henn, Kai Koskimies, Frantisek Plasil, Gustav Pomberger, Wolfgang Pree, Michael Stal, and Clemens Szyperski, "What Characterizes a (Software) Component?" Journal of Software – Concepts and Tools, vol. 19, no.1, pp. 49–56, June 1998, Springer.
[11] George T. Heineman and William T. Councill (Eds.), Component-Based Software Engineering: putting the pieces together. Addison-Wesley, May 2001.
[12] Jeffrey Voas, "Composing software component 'ilities'," IEEE Software, vol. 18, no. 4, pp. 16–17, July/Aug 2001.
[13] Ivica Crnkovic and Magnus Larsson (Eds.), Building Reliable Component-Based Software Systems. Artech House Inc., 2002.
[14] G. Rothermel, M. J. Harrold, and J. Dedhia. Regression test selection for C++ software. *Software Testing, Verification& Reliability*, 10(2):77–109, June 2000.
[15] T. H. Tse and Z. Xu. Test case generation for class-level object-oriented testing. In Proceedings of 9th International Software Quality Week (San Francisco, California, May 21–*24)*, pages 4T4.0–4T4.12, 1996.
[16] C. D. Turner and D. J. Robson. The state-based testing of object-oriented programs. In *Proceedings of the International Conference on Software Maintenance*, pages 302–310, Sept. 1993.
[17] Lazic L., Kolasinac A., Avdic D., "The Software Quality Economics Model for Software Project Optimization", WSEAS Transaction on Computers, Issue 1, Vol. 8, January 2009.
[18] NASA Software Quality Assurance Center Software Assurance Guidebook_NASA-GB_A201,1989
[19] Liao H., Enke D., and Wiedbe H., "An Expert Advisory System for ISO 9001 Quality System," Expert System with Application, vol. 27, pp. 313-322, 2004.
[20] Li H., Meissner J., "Improving Quality in Business Process Outsourcing through Technology", 2008.

