Advances in Science and Technology
Research Journal
Volume 9, No. 27, Sept. 2015, pages 50–57
DOI: 10.12913/22998624/59084

Original Article

QAM: PROPOSED MODEL FOR QUALITY ASSURANCE IN CBSS

Latika Kharb[1]

[1] Jagan Institute of Management Studies (JIMS), Rohini, Delhi, India, e-mail: latika.kharb@jimsindia.org

Received: 2015.05.31
Accepted: 2015.08.05
Published: 2015.09.01

ABSTRACT

Component-based software engineering (CBSE) / Component-Based Development (CBD) lays emphasis on decomposition of the engineered systems into functional or logical components with well-defined interfaces used for communication across the components. Component-based software development approach is based on the idea to develop software systems by selecting appropriate off-the-shelf components and then to assemble them with a well-defined software architecture. Because the new software development paradigm is much different from the traditional approach, quality assurance for component-based software development is a new topic in the software engineering research community. Because component-based software systems are developed on an underlying process different from that of the traditional software, their quality assurance model should address both the process of components and the process of the overall system. Quality assurance for component-based software systems during the life cycle is used to analyze the components for achievement of high quality component-based software systems. Although some Quality assurance techniques and component based approach to software engineering have been studied, there is still no clear and well-defined standard or guidelines for component-based software systems. Therefore, identification of the quality assurance characteristics, quality assurance models, quality assurance tools and quality assurance metrics, are under urgent need. As a major contribution in this paper, I have proposed QAM: Quality Assurance Model for component-based software development, which covers component requirement analysis, component development, component certification, component architecture design, integration, testing, and maintenance.

Keywords: component-based software systems (CBSS), QAM, software architecture, components, quality assurance.

INTRODUCTION

The life cycle of component-based software systems

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 [1]. In recent future, software component based systems will become the most preferred industry approach towards development of improved software. In general, the architecture of software defines a system in terms of computational components and interactions among the components. The emphasis is on composing and assembling components that have been developed separately and/or independently. Component-based software systems (CBSS) are developed by selecting and assembling various components together rather than programming overall system from scratch: this is the differentiation between the life cycle of component-based software systems and life cycle of traditional software systems.

The life cycle of component-based software systems consists of phases like: requirements analysis; software architecture selection, construction, analysis, and evaluation; component

identification and customization; system integration; system testing; and software maintenance. Typically, the development of component-based systems starts with a collection of existing components [2]. In component based software systems (CBSS), software components are assembled so that they interact with each other and satisfy the predefined functions, so that each component has to provide a pre-specified service with other components and thus interface is an important concern to be discussed before proposing metrics for measurement of integration complexity [3]. Integration helps to decide upon how to provide communication and coordination among various components of a target software system. The life cycle of component-based systems includes two main parts:

- In order to assess a component; evaluation of each candidate COTS component is done on functional and quality requirements; and
- Before integration, each candidate COTS component is customized/ modified.

Quality assurance for component-based software systems

Quality assurance for component-based software systems during the life cycle is used to analyze the components for achievement of high quality component-based software systems. As quality assurance technologies for component-based software systems are currently premature, due to the difference between some specific characteristics of component systems and traditional systems.

Although some quality assurance techniques [4] and component based approach to software engineering [5] have been studied, there is still no a clear and well-defined standard or guidelines for component-based software systems. Therefore, identification of the quality assurance characteristics, quality assurance models, quality assurance tools and quality assurance metrics, are under urgent need.

PROPOSED QAM: A QUALITY ASSURANCE MODEL FOR COMPONENT-BASED SOFTWARE SYSTEMS

In this paper, we have tried to stress upon the above said urgent needs. As much work is yet to be done for component-based software development; so in context of component-based software systems (CBSS), quality assurance technologies have to address the two inseparable parts namely:

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

Firstly, to answer the above said questions related to quality assurance; it seems necessary that some quality assurance models should be proposed to define the overall quality of components in software systems; secondly in order to evaluate a component, we must determine how to certify the quality of the component. The quality characteristics of components help us to guarantee the quality of the components, and thus guarantee the quality of the whole component-based software systems.

Because component-based software systems are developed on an underlying process different from that of the traditional software, their quality assurance (QA) model should not only address the process of components but also the process of the overall system. In this section: keeping in mind the thought that Quality cannot be achieved by assessing an already completed product, I have proposed a QA framework model for the component-based software development paradigm. The aim of proposing a QA model is to prevent quality defects or deficiencies in the first place, and to make the products with complete quality assurance measures. The main practices related to components and systems in this model contain the following phases:

- Component Requirement Analysis Phase,
- Component Development Phase,
- Component Certification Phase,
- Component Architecture Design Phase,
- Component Integration Phase,
- Component Testing Phase,
- Component Maintenance Phase.

Details of these phases and their activities are described as follows.

Component Requirement Analysis Phase

Component Requirement Analysis Phase is the first stage in the software development process. It encompasses those tasks that go into determining the needs or conditions to develop/ modify a software product, taking account of all the conflicting requirements of the various stakeholders/ users. For development of effective and quality software, requirements must be measurable, testable, related to identified business needs

or opportunities, and defined in detail for sufficient fulfilment of requirements in design phase.

Conceptually, requirements analysis includes three types of activity:

- **Eliciting requirements:** the task of communicating with customers and users to determine what their requirements are. This is sometimes also called requirements gathering.
- **Analyzing requirements:** determining whether the stated requirements are unclear/ incomplete/ ambiguous and then resolving these issues.
- **Recording requirements:** Requirements might be documented in various forms, such as use cases or process specifications.

In CBSD, component requirement analysis is the process of discovering, understanding, documenting, validating and managing the requirements for each candidate component. The objectives of component requirement analysis include production of complete, consistent and relevant requirements that a component should realize, inclusive of constraints with respect to programming language, operating platform and interfaces.

Our proposed component requirement analysis process overview diagram is as shown in Figure 1. Initiated by the request of users or customers for new development or changes on old system, component requirement analysis consists of four main steps:

- **Requirements gathering and definition:** From first stage of requirement gathering and definition to second stage of component requirement analysis, a format or structure of URD (user requirement documentation) is produced as output of requirement gathering and definition stage.
- **Requirement analysis:** In second stage of requirement analysis, draft of URD is made through component requirement analysis stage.
- **Component modelling:** In this phase, updated version of CRD (current-user requirement documentation) along with model is provided by component modelling stage to next stage of requirement validation stage.
- **Requirement validation:** In this phase, current CRD (current-user requirement documentation) is validated through requirement validation and once its validated, it's followed till component maintenance phase; else changes are made iteratively starting with requirement gathering and definition stage.

The output of this phase is the current-user requirement documentation (CRD), which is transferred to the next phase: Component Development Phase and in case of changes in user requirement; requirement gathering and definition stage is done else it progresses smoothly till component maintenance phase.

Component development phase

Component development phase is the process of implementing the requirements for a well-functional and high quality component having multiple interfaces. The objectives of component development are the final component products satisfying requirements with correct/expected results, the flexible interfaces, and unambiguous development documents.

The proposed component development process overview diagram is as shown in Figure 2. Component development consists of four procedures:

- **Implementation:** In this phase, input is the component requirement document on the basis of which the draft structure of component is devised and submitted to next procedure of function testing.

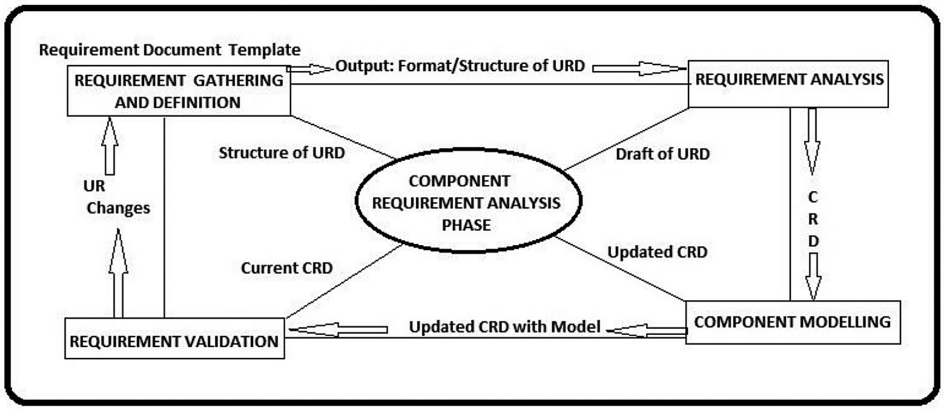

Fig. 1. Component requirement analysis phase

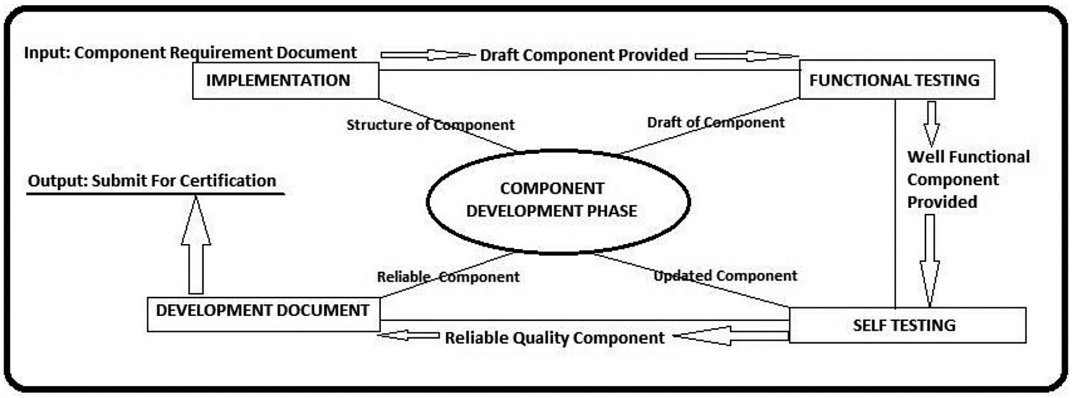

Fig. 2. Component development phase

- **Function testing:** This phase gets the draft structure of component as input and a well functional component is provided to next procedure.
- **Reliability testing:** The updated Reliable Quality Component (RQC) is the output from self-testing procedure and given as input to development document.
- **Development document:** The development document contains RQC details and is submitted to next phase for certification after reliability checks.

The input to this phase is the component requirement (CRD) along with its development document and output should be developed component and its documents ready for later phases of certification and system maintenance.

Component certification phase

A component should be able to be developed, acquired and incorporated into the system and composed with other components independently of time and space [6]. Certification certifies that it will do precisely this (for all contexts where its dependencies are satisfied). It will therefore provide a basis for component certification. The objectives of component certification phase are to outsource, select and test the candidate components and check whether they satisfy the system requirement with high quality and reliability [7]. The governing policies in component certification are:

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

The component certification process overview diagram is as shown in Figure 3. The input to this phase should be component development document, and the output should be testing documentation for system maintenance. Component certification phase involves four procedures:

- **Component outsourcing:** It includes defining component functions and release of the candidate component to next stage. This stage involves managing a component outsourcing.
- **Component testing:** This procedure confirms that a candidate component satisfies the re-

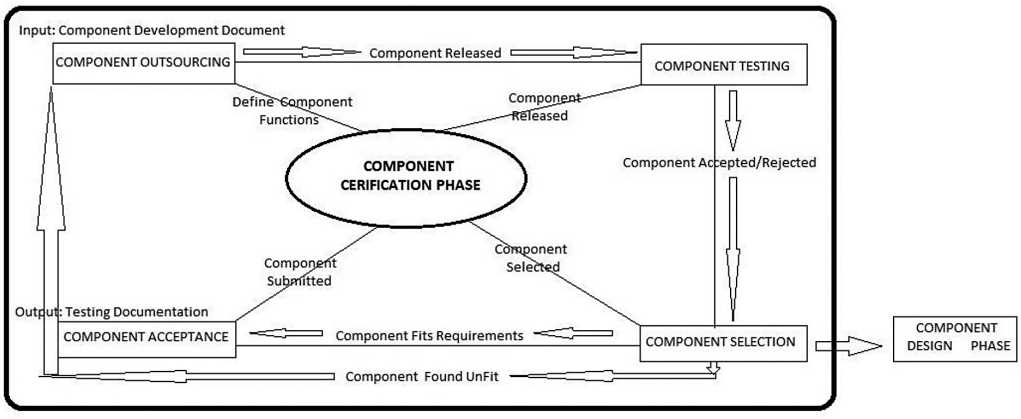

Fig. 3. Component certification phase

quirement with required quality and reliability. It either accepts and/or rejects the component when released.

- **Component selection:** This procedure is used for selecting the right components in accordance to the requirement for both functionality and reliability.
- **Component acceptance:** This procedure confirms the acceptance of submitted component and forwards the same to component outsourcing stage. If a component is found unfit, it is sent to the first phase of iterative cycle of component certification phase.

Component architecture design phase

In general, a system architecture or systems architecture is the conceptual model that defines the structure, behaviour, and more views of a system [8]. Technically speaking, an architecture description is a formal description and representation of a system that is organized in a way that supports reasoning about the structures and behaviours of the system.

Software architecture can be explained through three principles [9]:

- **Principle 1:** Software architecture should be defined in terms of elements that are coarse enough to allow for human intellectual control and specific enough to allow for meaningful reasoning.
- **Principle 2:** Business (and/or mission) goals determine quality-attribute requirements.
- **Principle 3:** Quality-attribute requirements guide the design and analysis of software architectures.

Component architecture design phase is the process of evaluating, selecting and creating software architecture of a component-based system. The software architecture of a program or computing system is that structure/ structures of the system which comprises of software components and their relationships. Documenting software architecture not only facilitates communication between stakeholders but also helps in documentation about high-level design and allows reuse of design components and patterns between projects. Software design is a process of problem-solving and planning for a software solution. After the purpose and specifications of software are determined, software developers will design or employ designers to develop a plan for the solution. The objectives of our component architecture design phase include:

- collection of user's requirement,
- identification of system specification,
- selection of appropriate system architecture,
- selection of implementation details such as platform, programming languages, etc.

Component architecture design should address the advantage for selecting a particular architecture from other architectures.

The process overview diagram is as shown in Figure 4. Component Architecture Design Phase starts with a structured requirement document template that includes a draft component requirement document. After the analysis of the component requirement document, component certification document is created that is integrated with component architecture to finally create a testable component specification document. This phase consists of:

- **Requirement gathering:** The input to requirement gathering stage is the format/structure of requirement document template. A draft component requirement document is submitted to next stage of analysis.

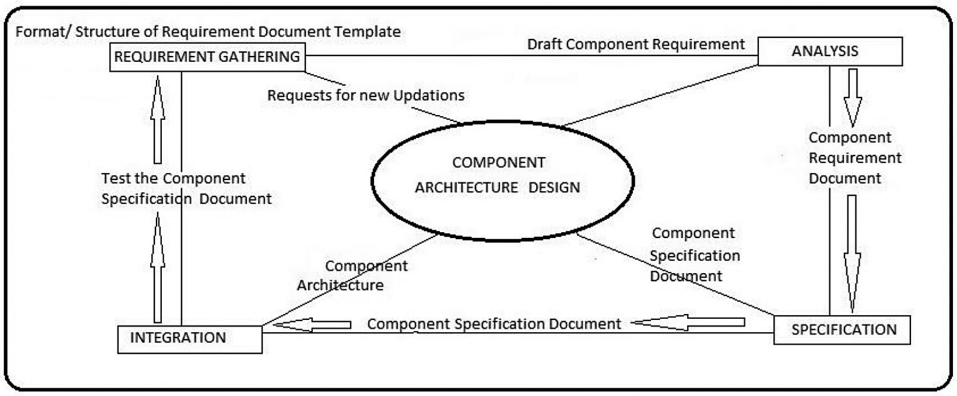

Fig. 4. Component architecture design phase

- **Analysis:** The draft component requirement document is analyzed in this stage and submitted to next stage of specification.
- **Specification:** Component specification document after analysis is submitted to component architecture for integration.
- **Integration:** In integration stage, candidate component are integrated to create a testable component.

The output of this phase should be the system specification document for integration, and system requirement for the system testing phase and system maintenance phase.

Component integration phase

Typically, the development of component based systems starts with a collection of existing components [10]. Component integration is the process of assembling components selected into a whole system under the designed system architecture. A software component is a unit of composition that can be deployed independently by third parties and contains only the contractually specified interfaces and the explicit context dependencies [3].

The process overview diagram is as shown in Figure 5. The input is the system requirement documentation and the specific architecture. There are four steps in this phase:

- **Integration:** Input of this step is component requirement document and component architecture design that provides a draft component to next stage of testing.
- **Testing:** In this step, we test component for faults. If faults are found, we go to next stage of component change.
- **Component change:** During this step, new component is selected; component integration document is generated and provided to next step.
- **Reintegration (if necessary):** If faulty component is found and has to be changed, then component is re-integrated in this stage with current component and all later stages go iteratively.

After exiting this phase, we will get the final system ready for the system testing phase and the document for the system maintenance phase.

Component testing phase

Software testing is an important technique for validating and checking the correctness of any kind of software [11]. In other words, we can say that the goal of the testing activity is to find as many errors as possible before the user of the software finds them. We can use testing to determine whether a program component meets its requirements. To accomplish its primary goal (finding errors) or any of its secondary purposes (meeting requirements), software testing must be applied in a systematic fashion [12].

We perform testing for evaluating a system to:

- confirm that the system satisfies the specified requirements;
- identify and correct defects in the system implementation.

Software testing, depending on the testing method employed, can be implemented at any time in the development process. However, most of the test effort occurs after the requirements have been defined and the coding process has been completed.

The process overview diagram is as shown in Figure 6. This phase consists of selecting testing strategy, component testing, user acceptance test-

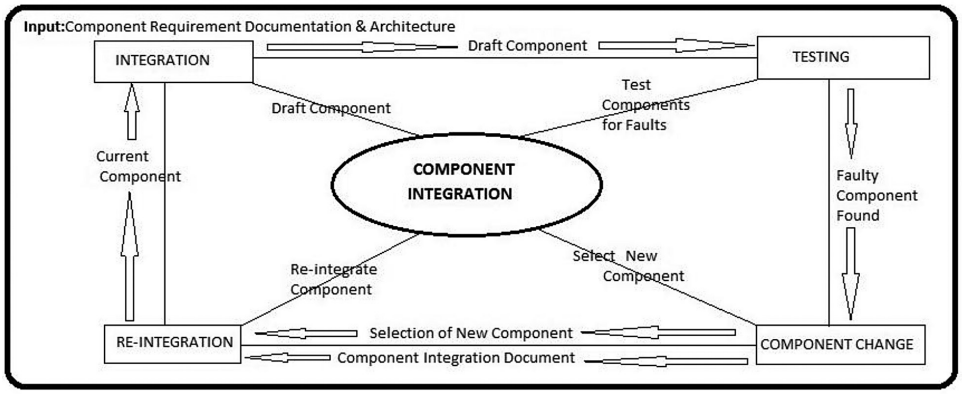

Fig. 5. Component integration phase

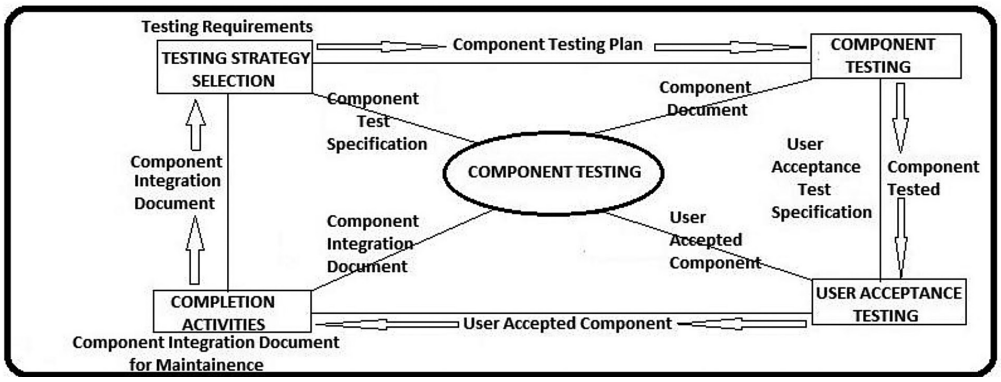

Fig. 6. Component testing phase

ing, and completion activities. The activities of each stage are:

- **Selecting test strategy:** The input to this stage is pre-integrated components and on the basis of their nature test strategy is selected and test cases are managed accordingly.
- **Component testing:** In this stage, we check integrated components for design and development. The ready to deliver component is now delivered to next stage of user acceptance testing.
- **User acceptance testing:** User acceptance testing of final product/components before delivery is essential for quality assurance. This stage could provide results in two ways: user acceptable or user rejected/ to be modified product.
- **Completion activities:** A complete testable user accepted product is delivered to stakeholder. This product is future ready for maintenance. If user acceptance testing initiates some kind of rejections, then this stage is skipped and integrated components goes to stage one for modifications.

The input should be the documents from component development and system integration phases. And the output should be the testing documentation for system maintenance.

Component maintenance phase

Maintenance commonly uses the majority of a software project's development resources. Around 75% of maintenance work is responding to changing requirements and operating environment. Most of the rest is bug-fixing. Only 1 in 5 development groups surveyed made a point of checking maintainability during QA, and just 3% of maintenance time is spent improving maintainability. Important technical factors affecting maintainability include:

- readability
- preservation of knowledge
- modifiability
- testability.

To improve productivity and quality during maintenance, and to avoid moving to servicing prematurely, it is important to support all four of these areas. The software will have to be modified to cater for the addition of the new functionality. Software maintenance is the process of providing service and maintenance activities needed to use the software effectively after it has been delivered. The objectives of system maintenance are to provide an effective product or

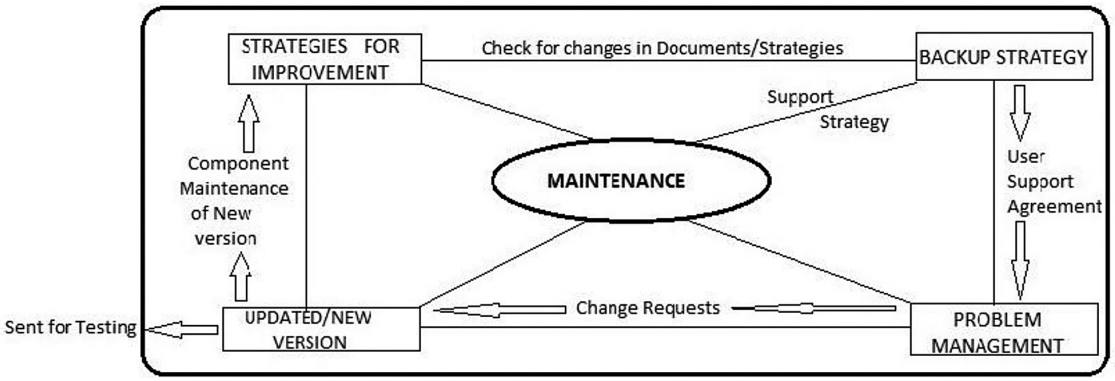

Fig. 7. Component maintenance phase

service to the end-users while correcting faults, improving software performance or other attributes, and adapting the system to a changed environment. There shall be a maintenance organization for every software product in the operational use. All changes for the delivered system should be reflected in the related documents.

The process overview diagram is as shown in Figure 7. According to the outputs from all previous phases as well as request and problem reports from users, system maintenance should be held for determining support strategy and problem management (e.g., identification and approval). As the output of this phase, a new version can be produced for system testing phase for a new life cycle.

CONCLUSION

In this paper, we conclude that COSD is a promising discipline to be considered as one of the alternatives to promote software evolution and also to improve the software development process of currently complex systems with the help of our proposed QAM: Proposed Model for Quality Assurance in CBSS. Research and practice in the areas of component technology, and software architecture to date has been conducted largely in isolation and has only touched on a few core issues. In this paper, we survey current component-based software technologies and the features they inherit. We propose a QAM model for component-based software development, which covers both the component quality assurance as well as their interactions. As our future work we will apply the QAM model to real world projects so that it can actually guide the practices of component based software development.

REFERENCES

1. Kharb L.: Complexity metrics for component-oriented software systems. ACM SIGSOFT Software Engineering Notes, 33(2), 2008, Article No. 4.
2. McInnis: Component-based development: The concepts, technology and methodology. Castek Software Factory Inc., www.CBD~HQ.com.
3. Kharb L., Rathore V.S.: Software component complexity measurement through proposed integration metrics. Journal of Global Research in Computer Science, 2(6), 2011.
4. Yacoub S.M., Cukic B., Ammar H.H.: A component-based approach to reliability analysis of distributed systems. Proceedings of the 18th IEEE Symposium on Reliable Distributed Systems, 1999, 158–167.
5. Yacoub S.M., Cukic B., Ammar H.H.: A scenario-based reliability analysis of component-based software. Proceedings 10th International Symposium on Software Reliability Engineering, 1999, 22–31.
6. Ning J.Q., Miriyala K., Kozaczynski W.: An architecture-driven, business-specific, and component-based approach to software engineering. Proceedings Third International Conference on Software Reuse: Advances in Software Reusability, 1994, 84–93.
7. Szyperski C.: Component software: Beyond object-oriented programming. Addison-Wesley, 1998, 213.
8. Kharb L.: CCTF: Component Certification & Trust Framework. International Journal of Scientific Research in Computer Science and Engineering, 1(6), 2013.
9. Jaakkola H. and Thalheim B.: Architecture-driven modelling methodologies. In: Anneli Heimbürger et al. (Eds) Proceedings of the XXII Conference on Information Modelling and Knowledge Bases. IOS Press. 2011, p. 98.
10. Bass L., Kazman R., Klein M.H.: The components of software architecture design and analysis. Published in News at SEI on April 1, 2005: http://www.sei.cmu.edu/library/abstracts/news-at-sei/architect20054.cfm.
11. Kharb L.: Proposed C.E.M (Cost Estimation Metrics): estimation of cost of quality in software testing. International Journal of Computer Science and Telecommunications, 6(2), 2015.
12. Kharb L.: Assessment of component criticality with proposed metrics. INDIACom-2008: Computing for Nation Development, by AICTE, IETE, and CSI, 2008, 453–455.