Periodicals of Engineering and Natural Sciences ISSN 2303-4521
Vol. 7, No. 3, September 2019, pp.1177-1185
Available online at: http://pen.ius.edu.ba

A new model for study of quality attributes to components based development approach

Rafid Nabil Jaffar[1], **Alaa Abd Al muhsen Hussain**[2], **Wisam Chiad**[3]
[1,2] Departement of Computer Information Systems, College of Computer Science & IT, University of Al-Qadissiyah
[3] Departement of Chemistry, College of Science, University of Al-Qadissiyah
rafid.jaffar@qu.edu.iq[1], alaa.abd@qu.edu.iq[2], w.chiad@qu.edu.iq[3]

Article Info

Received

***Keyword*:**

Component Based Approach

Quality Model

Quality Attributes

A-Model for CBD

ABSTRACT

Software development costs, time-to release and quality product are important factors affecting the construction of software. Different types of tools and techniques are suggested by researchers to improve in delivering quality software systems with lower cost and reduce time to delivery. One such practice is development of software using Component Based Software Development (CBSD) techniques [1]. CBSD recommended building software systems using existing reusable components, instead of writing from scratch. The main objective of CBSD is to writes once and reuse any number of time with no or modification. Some of the advantages that a company may available by adapting CBSD for the Software development are shorter development time which results in meet tight dead line, increase productivity and Quality Product. CBSD also, support reusability [3]. The aim of this paper is to develop the new model of software product and describe the characteristics of some selected of attributes of CBSD models that are widely practiced in software industries with explain about IT project. We proposed a complete model for Component Based Software Development for reuse. This Model will cover both component based software development as well as Component development phases for A-Model. This Model is represent one good solution for Component Based Development with reduce cost and time to deliverable and save the quality of product.

***Corresponding Author*:**

Rafid Nabil Jaffar,
Departement of Computer Information Systems, University of Al-Qadissiyah,
College of Computer Science & IT, Iraq
E-Mail: rafid.jaffar@qu.edu.iq

1. Introduction:

Component Based Development (CBD) is becoming increasingly important for the software industries are not provide by another models. Component-based development is a CBSE activity that occurs in parallel with engineering paradigm. Using first three phases like analysis and architectural design with design specifications methods. The software team refines an architectural style that is appropriate for the analysis approach created for the application to be built[2] . CBD is supposed to reduce the cost and time to deploy the product of software applications while increasing their quality. Since components are reused, they are likely to be more reliable than software developed from scratch, as they were tested under a larger variety of conditions. Cost and time savings result from the effort that would otherwise be necessary to develop and integrate the functionalities provided by the components in each new software application. In this research, we are describe and concerned with the evaluation of software components quality. By using A-Model this evaluation should be performed using a component quality model [6]. The development and validation of a model that answers these questions would offer a very useful evaluation tool to clients looking for components to include in their software applications . They could use here the following component definition: a software component is an independently deployable implementation of some functionality, to be reused as a wide range of some applications [1].

2. Theoretical framework

2.1 A Quality Model for Components Based Development: Quality Characteristics

There are two main things in the research of software quality: one of them is concerned with evaluating the software process quality while the other focuses on the product quality itself. Several product quality models can be found in the literature, such as [3-9]. Among these models, the most consensual model is the one offered by ISO 9126. These models include generic software quality attributes. Besides, they were conceived at the system level, not at the component one. While some of their characteristics are appropriate to the evaluation of software components, others are not well suited for that task. Some of their quality characteristics, such as fault tolerance are typically evaluated at the system level, rather than for each of the components. A quality model for components should be tailored to use only the characteristics that apply to components. In [10] one such model is proposed for the evaluation of COTS (Commercial Off The Shelf) components. The COTS that one of the software resource that have fully validate and the risk relatively low[2]. The model proposed in this paper has a wide attributes of application, as it is targeted for software components in general. As a starting point, they will use the quality model described in [9] (see Table 1) which is an attempt to improve some of the shortcomings of the ISO9126 model. I try to adapt it to the special needs of quality components[4]. The former are more suited for component quality evaluation on the to the developers of components point of view, while the latter are of special interest to the components consumers (developers using components that already exist in the construction of their applications). Below the (Table-1) that represent the important quality characteristics are model depend it.

Table 1. Quality Attributes

Features	Relation Parts
Maintainability	Diagnoseability, Repairability, Extensibility, Testability, Compliance
Dependability	Reliability, Security, Safety, Compliance
Usability	Learnability, Configurability, Operability
Efficiency	Time behavior, Resource behavior, Compliance

2.2 Criteria of Component Based Software Development

We suggest the Criteria of Component Based Software Development that have five attributes with focus on the style of new design of development and describe in detail all that features below in Fig(1).

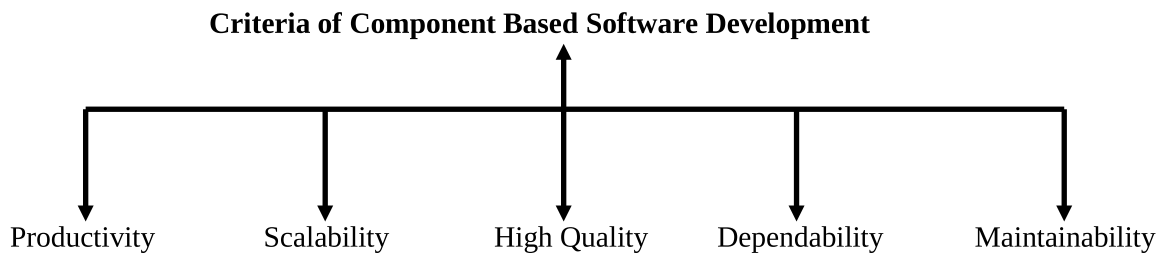

Fig. 1. Criteria of Component Based Software Development

2.2.1 Software Productivity

Productivity in a manufacturing system by counting the number of units that are produced and dividing this by the number of person-hours required to produce them. However, for any software problem, there are many different solutions, each of which has different attributes[6]. One solution may execute more efficiently while another may be more readable and easier to maintain. When solutions with different attributes are produced, comparing their production rates is not really meaningful . Nevertheless, as a project manager, you may be faced with the problem of estimating the productivity of software engineers [11]. You may need these productivity estimates to help define the project cost or schedule, to inform investment decisions or to assess whether process or technology improvements are effective. Productivity estimates are usually based on measuring attributes of the software and dividing this by the total effort required for development. There are two types of metric that have been used like Size-related metrics, Function-related metrics, and Lines of source code per programmer-month (LOC/pm or SLOC/pm) is a widely used software productivity metric [13].

2.2.2 Software Scalability

A system, business or software that is described as scalable has an advantage because it is more adaptable to the changing needs or demands of its users or clients[10]. Scalability is often a sign of stability and competitiveness, as it means the network, system, software or organization is ready to handle the influx of demand, increased productivity, trends, changing needs and even presence or introduction of new competitors [8]. The standard is applicable across several industries covering healthcare, pharmaceuticals and electronics; and it outlines the approach for the design, classification and operation of cleanrooms [12].

2.2.2.1 The Benefits of Scalable Software

Scalability has both long- and short-term benefits. At the outset it lets a company purchase only what they immediately need, not every feature that might be useful down the road[2]. For example, a company launching a data intelligence pilot program could choose a massive enterprise analytics bundle, or they could start with a solution that just handles the functions they need at first. A popular choice is a dashboard that pulls in results from their primary data sources and existing enterprise software [7]. When they grow large enough to use more analytics programs, those data streams can be added into the dashboard instead of forcing the company to juggle multiple visualization programs or build an entirely new system. Building this way prepares for future growth while creating a leaner product that suits current needs without extra complexity [9]. It requires a lower up-front financial outlay, too, which is a major consideration for executives worried about the size of big data investments. Scalability also leaves room for changing priorities. That off-the-shelf analytics bundle could lose relevance as a company shifts to meet the demands of an evolving marketplace

2.2.3 Software Quality

Is reasonably bug or defect free, delivered on time and within budget, meets requirements and/or expectations, and is maintainable. ISO 8402-1986 standard defines quality as the totality of features and characteristics of a product or service that bears its ability to satisfy stated or implied needs [14].

Key aspects of quality for the customer include:

- Good design – looks and style
- Good functionality – it does the job well
- Reliable – acceptable level of breakdowns or failure
- Consistency

2.2.4 Software Dependability

The software should be available whenever required, as well as operating properly, safely and reliably, without any adverse side effects or security concerns [4]. It is essential that the software used in systems in the safety critical and security critical fields is dependable, as the consequence of failure (e.g. the failure of a nuclear power plant) could be massive damage leading to loss of life or endangering the lives of the public Dependability engineering is concerned with techniques to improve the dependability of systems, and it involves the use of a rigorous design and development process to minimize the number of defects in the software [10]. A dependable system is generally designed for fault tolerance, where the system can deal with (and recover from) faults that occur during software execution. Such a system needs to be secure, and able to protect itself from accidental or deliberate external attacks. Table 2 lists several dimensions of dependability.

Modern software systems are subject to attack by malicious software such as viruses that change the behavior of the software, or corrupt data causing the system to become unreliable [6]. There is a trade-off between dependability and the performance of the system, as dependable systems will need to carry out extra checks to monitor themselves and to check for erroneous states, and to recover from faults before failure occurs[11]. This inevitably leads to increased costs in the design and development of dependable systems. In (Table-2) the dimension of Dependability

Table. 2. Dimensions of Dependability

Dimension	Description
Availability	System is available for use at any time
Reliability	The system operates correctly trustworthy and is
Safety	The system does not injure people or damage the environment
Security	The system prevents unauthorized intrusions

Software availability is the percentage of the time that the software system is running, and is a measure of the uptime/downtime of the software during a particular time period. The downtime refers to a period of time when the software is unavailable for use (including planned and unplanned outages), and many companies aim to develop software that is available for use 99.999% of the time in the year (i.e. a downtime of less than 5 min per annum) [7]. This goal is known as five nines, and it is a common goal in the telecommunications sector. Safety-critical systems are systems where it is essential that the system is safe for the public, and that people or the environment is not harmed in the event of system failure [9]. These include aircraft control systems and process control systems for chemical and nuclear power plants. The failure of a safety critical system could in some situations lead to loss of life or serious economic damage. The security of the system refers to its ability to protect itself from accidental or deliberate external attacks, which are common today since most computers are networked and connected to the Internet [11]. Encryption is one way to reduce system vulnerability, as encrypted data is unreadable to the attacker. There may be controls that detect and repel attacks, and these controls are used to monitor the system and to take action to shut down parts of the system or restrict access in the event of an attack [13]. There may be controls that limit exposure (e.g. insurance policies and automated backup strategies) that allow recovery from the problems introduced. It is important to have a reasonable level of security as otherwise all of the other dimensions of dependability (reliability, availability and safety) are compromised. Security loopholes may be introduced in the development of the system, and so care needs to be taken to prevent hackers from exploiting security vulnerabilities [15].

2.2.5 Software Maintainability

Means fixing, updating, servicing and to modify the system or update the software for performance improvements or for the correction of faults. Maintainability also includes the Addition of new functionality or the adaptation of software to meet new requirements for the customer needs . Software maintainability is the degree of an application to repaired or enhanced it [2]. During the system development life cycle (SDLC) this phase requires more development effort than any other phase. Approximately 75 percent of the cost is related to software maintenance . Maintainability increases the reliability, efficiency or safety of the software. It is also used to make future maintenance easier . It is used to increase the lifetime of the software. Maintainability repair or replace the faulty components and make the software even better as compared to the previous condition of the software [8]. Software maintenance is required when the customer demands new features and new functions in the software. Sometimes maintenance is required when the hardware of the system is changed then the modification of software is needed.
Market conditions and organization changes are also the reasons for software modification. It also includes that when the issue is detected, immediately fix it before it becomes a big problem [10]. Sometimes viruses and malware are detected in the software which causes problems for the user than software maintenance is required to fix it or improve the performance. The idea behind the Maintainability attributes to modify the requirements according to customer need. As below Fig(2) to represent the Maintainability attributes for

CBD . The first group is users, the second one is developers of software, the third one is owners of IT companies. Thus, people start joining different formalized and nonformalized interest groups, such as: community, hackathon, co-working, hackerspace, networks, IT-cluster, and others [17].

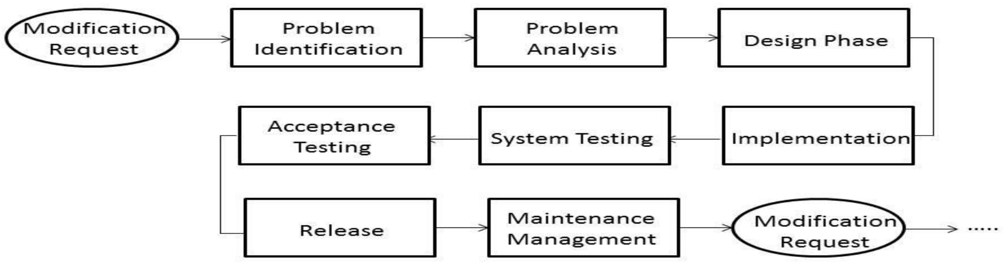

Fig. 2. Maintainability attributes for CBD

3. Research Methodology

3.1 A-Model for Component Based Development

We develop a new model for Component Based Development approach to implement the reusability that introduce the reduce of cost and reduce of time with save on the high level of quality. In the Fig(3) below the A-Model for CBD that we suggest to describe the reusability with the procedures for new style of development approach with different phases of new model that suggest.

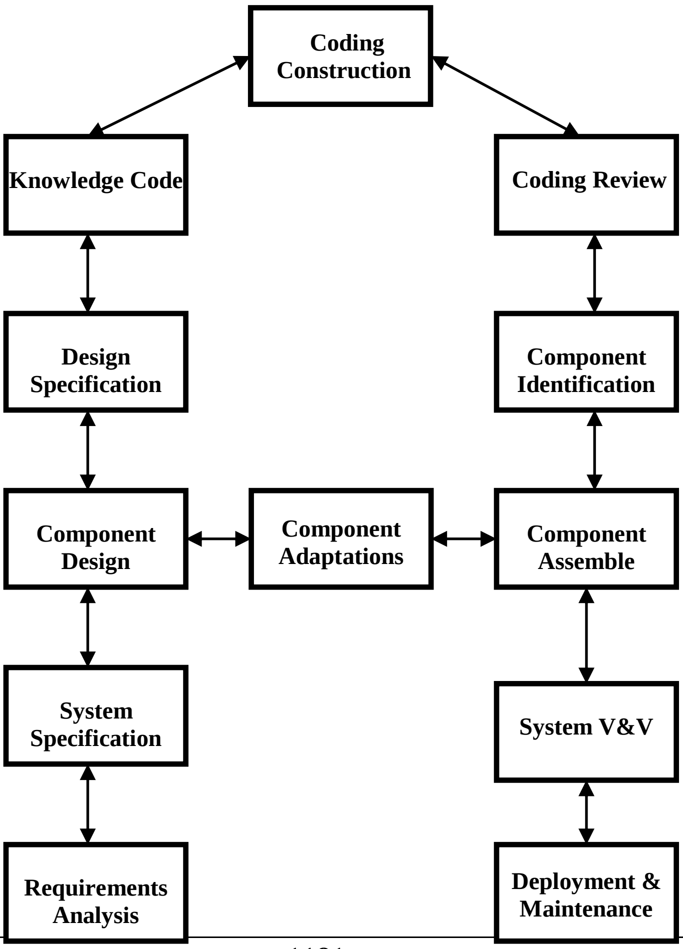

Fig. 3. A-Model for CBD

3.2 Requirements Analysis

The process of defining the expectations of the users for an application that is to be built or modified. Requirements analysis involves all the tasks that are conducted to identify the needs of different stakeholders[1]. Therefore requirements analysis means to analyze, document, validate and manage software or system requirements. High-quality requirements are documented, actionable, measurable, testable, traceable, helps to identify business opportunities, and are defined to a facilitate system design [3].

3.3 Component Design

The activity of the Product Design for creating a Component Design. The Product Architecture identifies a set of Adaptable Components that may be used to implement a work product family . A Component Design is a design specification for one of these adaptable components[9]. Application engineers, using the Generation Procedure, may adapt and compose a set of these components to implement certain work products, or portions [11]. Each component must be designed to satisfy relevant aspects of the Product Requirements and all design structures of the Product Architecture

3.4 Design Specifications

Describe how a system performs the requirements outlined in the Functional Requirements[5]. Depending on the system, this can include instructions on testing specific requirements, configuration settings, or review of functions or code [12]. All requirements outlined in the functional specification should be addressed; linking requirements between the functional requirements and design specification .
Design Specification Examples:

Good requirements are objective and testable. Design Specifications may include:

1. Specific inputs, including data types, to be entered into the system
2. Calculations/code used to accomplish defined requirements
3. Outputs generated from the system
4. Explaining technical measures to ensure system security
5. Identify how the system meets applicable regulatory requirements

3.5 Knowledge Code

The programmer start with bring the tools that enable to construct the code by determine the suitable programming language with operating systems and take important points with different ideas about the shape of software[8]. The role of consultant that have experience about the software development with enable to introduce the solution for problem [14]. The characteristics of different programming languages are restriction depend on type of problem and the way to found the solution .

3.6 Coding construction

This stage in simpler terms is where the “real work begins” and we “build what is needed”.

1. The developers start to code as per the requirements and the developed design.
2. Along with the coding, all the other required set-up will begin. i.e., the database set up by database admin, interface and GUI creation by front-end developers, etc.
3. Along with coding, it is also important for developers to develop unit tests for their module, peer review other module’s unit tests, deploy builds to the intended environment and execute unit tests .

3.7 Coding Review

Code review is a phase in the software development process in which the authors of code, peer reviewers, and perhaps quality assurance (QA) testers get together to review code[3]. Finding and correcting errors at this stage is relatively inexpensive and tends to reduce the more expensive process of handling, locating, and fixing bugs during later stages of development or after programs are delivered to users [16]. Reviewers read the code line by line to check for :

1. Flaws or potential flaws
2. Consistency with the overall program design
3. The quality of comments
4. Adherence to coding standards.

Code review may be especially productive for identifying security vulnerabilities. Specialized application programs are available that can help with this process . Automated code reviewing facilitates systematic testing of source code for potential trouble such as buffer overflows, race conditions, memory leakage, size violations, and duplicate statements [4].

3.8 Component Identification

One of the most important and difficult tasks in developing component-based systems. The existing component development approaches do not provide a standard for Component Identification, and they depend on the experience of developers . The developer suppose identify the components inside the library for reuse instead of develop from scratch [3].

3.9 Component Assemble

Have been integrated via some form of common infrastructure. For consistent component assembly the infrastructure must consist of both a physical communication infrastructure such as a database or messaging infrastructure, and a set of conceptual agreements such as naming conventions that embody the shared semantics among the components [8]. This infrastructure will support component assembly and coordination, and differentiates planned, coordinated component assembly from amalgamation using ad hoc "glue." A number of industry standards for component infrastructures have been developed to facilitate communication among different infrastructure implementations [2].

3.10 Component Adaptations

Amended to address potential sources of conflict among components which are to be assembled to form an application system . Typically, simple scripts are written as a buffer between user requests and component actions. The buffer can be used to provide default information to the components, eliminate access to unwanted component behavior, insulation layer for the replacement of components. The figure implies a kind of component "wrapping," but other approaches are possible (e.g., the use of agents, intelligent mediators, and translators) [11]. The impact of quality management on learning, consider the quality system as a homogeneous set of practices and do not take into account the specificities of its components. However, some authors suggest that quality systems should be split into two categories: technical components (statistical control of processes, the use of benchmarking or the implementation of flexible processes) and social or contextual components (leadership, staff involvement, customer orientation, training and development of staff skills, teamwork, communication, shared vision) [4]. The risks to the project are influenced by factors such as the level of novelty for the organization, the complexity of the project, its duration, availability of resources, including highly skilled specialists, etc [16]

3.11 System Verification & Validation

Are hugely confused and debated terms in the software testing world. You will encounter all kinds of usage and interpretations of these terms, and it is our humble attempt here to distinguish between them as clearly as possible [12]. Verification is the process of evaluating work-products (not the actual final product) of a development phase to determine whether they meet the specified requirements for that phase . The objective of Verification is to ensure that the product is being built according to the requirements and design specifications. In other words, to ensure that work products meet their specified requirements [10]. The Validation is the process of evaluating software during or at the end of the development process to determine whether it satisfies specified business requirements . The objective to ensure that the product actually meets the user’s needs and that the specifications were correct in the first place [6].

3.12 Deployment & Maintenance

System Deployment phase evolved releasing product to a customer in other word software is made available to the customer for use. System deployment must be delivered using some specialized tool to make the deployment easy for the customer . Different version of system release must be maintain and handle properly. If there is an up gradation in the system it must be maintain in a metadata and repository [10]. Reconfiguration, adaptation, reinstallation of installed system need to be address properly to avoid run time issues usually found in installing system . Up-gradation and substitution of components are the main job of the system maintenance . System Up gradation usually occurs when a COTS supplier releases a new version of a component or when a new COTS component is obsolete. Modifications to the code are required in system maintenance . Customer satisfaction approach is the basic foundation of the company's strategy since its creation. Everything is structured around its aspects with a central goal which is of conquering new industries and retains the existents" (Silo Moderator) Once the company's strategic priorities have been defined and the quality policy has been established, the quality process contributed to the deployment of these guidelines into objectives and action plans [13]. Thus, the implementation of standardization was also accompanied by a strengthening of planning systems and control tools. The quality process organized control through its generalization on all processes and providing measurement logic through diverse indicators. The creation of shareholder value is no longer the only performance indicator [5].

4. Results and Discussion

We are develop the new model called A-Model that describe the reusability attributes with reduce the cost of software developmet and time with save the software quality and improve the performace through follow special tools and new techniques for developmet. We should suggest updated the system according to any new requirements with adapted for some attributes inside our model for give suitable results with continue to work on quality attributes.

5. Conclusions and Future Work

We suggest a new model to provide trust to component users are a common market practice enforced by component purchasers and often a legal imposition in other industries. Component certification is such a trust procedures with new approach. As a long term goal, we aim to contribute to the development of a certification process that is adequate to software components. The first step towards that objective is the definition of a quality model that can be used as a framework in component procedures. They currently working towards this goal. In the adapt of model with good attributes of software use by enhance the features of each process in software development with modern approach to reusable components that already exist. We are doing on improve the some features to interoperable with new approach and try to keep on the same track of development and reduce (time with cost) with a raise the degree of quality .

6. References

[1] A.I.Khan, " An Improved Model for Component Based Software Development", Scientific & Academic Publishing, 2012.
[2] B. Meyer, Object-Oriented Software Construction, 2nd ed. NJ, USA: Prentice Hall PTR, 1997.
[3] B. W. Boehm, B. K. Clark, E. Horowitz, R. Maduchy, R. Selby, and C. Westland, “An overview of the COCOMO 2.0 software cost model”, 1995.
[4] C.E.Mokhlis, A. Elmortada, M. Sbihi, K. Mokhlis, "The impact of ISO 9001 Quality Management on organizational learning and innovation: Proposal for a conceptual framework", Periodicals of Engineering and Natural Sciences, Vol.7, No.2, August 2019, pp.944-951.
[5] C.E.Mokhlis, A. Elmortada, S.Marwane, "Diagnosis of Organizational Change: A multi-level approach

(Case study of a French SME certified ISO 9001)", Periodicals of Engineering and Natural Sciences, Vol.7, No.2, August 2019, pp.932-943.

[6] F. B. Abreu, "Comparing Software Quality Models," : INESC Technical Report (in Portuguese), 2001.

[7] G. O'Regan, Concise Guide to Formal Methods, Undergraduate Topics in Computer Science, Springer International Publishing AG 2017.

[8] ISO 9126, "Information Technology - Software Product Evaluation – Software Quality Characteristics and Metrics,". Geneva, Switzerland: International Organization for Standardization.

[9] J. McCall, "Quality Factors," in Encyclopedia of Software Engineering, vol. I+II, J.J. Marciniak, Ed.: John Wiley & Sons, 1994, pp. 958-ff.

[10] Lamkharbach Yassine, Bazi Fathallah, Haji Latifa, "Study of changing statistics model's influence on the exploitation and conformity of results in the new standard version ISO 14644 part 1", Periodicals of Engineering and Natural Sciences, Vol. 6, No. 2, December 2018, pp.436-446.

[11] L. Bass, C. Buhman, S. Comella-Dorda, F. Long, J. Robert, R. Seacord, and K.Wallnau, "Volume I: Market Assessment of Component-Based Software Engineering," Software Engineering Institute, Technical Note CMU/SEI-2001-TN-007, May, 2001.

[12] L. J. Arthur, Measuring Programmer Productivity and Software Quality: Wiley-Interscience, 1985.

[13] M. Bertoa and A. Vallecillo, "Quality Attributes for COTS Components", presented at 6th International Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE'2002), Malaga, Spain, 2002.

[14] R. B. Grady and D. L. Caswell, Software Metrics: Establishing a Company-Wide Program. Englewood Cliffs, NJ, EUA: Prentice-Hall, 1987.

[15] R.S.Pressman, "Software Engineering" book, 5th Edition, McGraw-Hill, 2001.

[16] Vitalina Babenko, Liudmila Lomovskykh, Alvina Oriekhova, Liubov Korchynska, " Features of methods and models in risk management of IT projects", Periodicals of Engineering and Natural Sciences, Vol. 7, No. 2, August 2019, pp.629-636.

[17] V. Honcharenko, A. Panteleimonenko, A. Pozhar, V. Stetsenko, " Cooperatives in IT sector: theoretical and practical aspects", Periodicals of Engineering and Natural Sciences, Vol.7, No.2, August 2019, pp.597-607.