International Journal of Hybrid Information Technology
Vol.9, No.1 (2016), pp. 241-252
http://dx.doi.org/10.14257/ijhit.2016.9.1.21

Evaluating and Certifying Component-Based Software Using Weighted Assignment Technique

[1]Lata Nautiyal and [2]Preeti

[1]Graphic Era University, Dehradun
[2]Gurukula Kangri Vishwavidyalaya, Haridwar
E-mail: [1]lata.nautiyal1903@gmail.com, [2]preetishivach2009@gmail.com

Abstract

Certification refers to the verification of definite feature of an object, person, or an organization. This verification is often, but not always, provided by some form of external review, education, assessment, or audit. In general, the main certification idea is to bring quality process to a certain software product. One of the core goals to achieve quality in component based product is to acquire reliability, quality and compatibility with other components. We have proposed an evaluation process for certification of component-based software. Certification is performed at component as well as system level. The six factors given by ISO are used to certify the system or component. Some other factors are also taken into consideration for certification process. Unstructured weighting technique is used to assign weights to these seven factors.

Keywords*: Component, component-based software system, quality, certification, weighted assignment technique*

1. Introduction

Certification refers to the verification of definite feature of an object, person, or an organization. This verification is often, but not always, provided by some form of external review, education, assessment, or audit. We can also say that the process of verifying a property value associated with something, and providing a certificate to be used as proof of validity is called as certificate [1 and 2].

Component certification is a process to ensure that software components conform to precise standards; based on this certification process, trusted assemblies of components can be developed. However, the task of certifying the component seems to be very difficult because the software engineering community has expressed many and often divergent properties to evaluate software components. In general, the main certification idea is to bring quality process to a certain software product. One of the core goals to achieve quality in component based product is to acquire reliability, quality and compatibility with other components [3-5]. Normally, the software component quality assurance process occurs through models that measure its quality. These models describe and organize the component quality characteristics that will be taken into account during the quality assurance process. So, to measure the quality of a software component, it is highly important to develop a quality model. In this way, we aim to investigate a Component Quality Model, identifying its characteristics, the sub-characteristics, the quality attributes and the related attributes that compose the model [6].

Rest of the paper is organized as follows; next section presents the certification process of component-based software engineering (CBSE). In Section 3, we have discussed the work done in the certification area of CBSE. Section 4 presents the quality model, in Section 5 we have divided the certification into two levels. Section 6 discusses the proposed evaluation process. In Section 7, we have applied the proposed approach to an example. Finally Section 8 concludes the research work.

ISSN: 1738-9968 IJHIT

2. Certification in Component-Based Software Engineering

According to [7], there is a lack of an effective assessment of software components. Besides, the international standards that address the software products' quality issues (in particular, those from International Standard Organization (ISO) and IEEE) have shown to be too general for dealing with the specific characteristics of components. While some of their characteristics are appropriate to the evaluation of components, others are not well suited for that task. Even so, the software engineering community has expressed many and often diverging requirements to CBSE and trustworthy components. A unified and prioritized set of CBSE requirements for trustworthy components is a challenge in itself [8]. Still, as cited early, there are several difficulties in the development of component quality model, such as

(a) which quality characteristics and quality attributes should be considered,
(b) how we can evaluate them and
(c) who should be responsible for such evaluation? In this way, there is still no well-defined standard and component quality model to perform component certification [1, and 2]. This fact is due also to the relatively novelty of this area [3].

Although recent, we found into literature some component quality models. The promising works are based on ISO 9126 [4]. This standard is a generic software quality model and it can be applied to any software product by tailoring it to a specific purpose.

Even so, the works found into literature looking for analyze the ISO 9126 standard and propose such one model that are specific for software components [4]. The researchers aim to verify if each characteristics of ISO 9126 are adequate to the component context or if new characteristic need to be added or removed to the model. Thus, the quality models were proposed based on the component technology and software quality experiences of the researchers.

However, these models were not evaluated into academic or industrial scenario. In this way, the real efficiency to evaluate software components using these models remains unknown. Additionally, [4] did not specified the quality attributes and the metrics that should be used to measure the quality characteristics proposed in the model, becoming difficult to use this model in whatever scenario.

In this context, we are investigating effective ways to demonstrate that component certification is not only possible and practically viable, but also directly applicable in the software industry. And, through certification, some benefits can be achieved, such as: higher quality levels, reduced maintenance time, investment return, reduced time-to-market, among others. According to Weber & Nascimento [6], the need for quality assurance in software development has exponentially increased in the past few years. This fact could be seen through a nationwide project launched by the Brazilian government, whose main concerns are: developing a robust framework for software reuse [9], in order to establish a standard to the component development; and defining and developing a repository system and a component certification process. This project has been developed in conjunction with the industry and academia in order to generate a well-defined model that will be capable of developing, evaluating quality, storing and, after that, making possible for software factories to reuse these components [10, and 11].

3. Related Work

The increasing use of commercial components in large systems makes selection and assessment of components a vital activity [12]. In [13], the authors describe the association between the evaluations performed during certification and their selection. They propose a components-based life cycle for COTS and software product line development. There are various other life cycle models proposed by [22, and 23]. Authors

also identify the process characteristics between the two types of evaluation and finally classify the required qualities during certification process. In [14], the authors propose a concept called SCL (Software Certification Laboratories); they recommend that this concept must take part in the certification product role which then offers the consumer's trust. In this process, SCL took all the information from the developer's site and passed it to the consumer's site and then returned back to collect this information from the user and used it to produce the warranties according to these results.

In [15], a complete component-based business document modeling was produced. This modeling is built upon existing standards that are extended by introducing the concept of generic business document template out of the specific needs of the user's document. The result part of this paper is a complete library of reusable business components that has been developed to easily produce a new business model. Another work is [16], where the objective of this work was to describe a process to measure and certificate the ability of software component to perform the reliability quality.

In [17], author proposes a test-based approach to validate performance specifications against deployed component implementations. Woodman, *et al.*, [18] analyzed some processes involved in various approaches to CBD and examined eleven potential CBD quality attributes. According to Woodman et al., only six requirements are applicable to component certification: *Accuracy, Clarity, Replaceability, Interoperability, Performance* and *Reliability*. Concomitantly, with the objective of obtaining the properties that a component should have, in 2003, Hissam, *et al.*, [19] introduced Prediction-Enabled Component Technology (PECT) as a means of packaging predictable assembly as a deployable product. A PECT is the integration of a component technology with one or more analysis technologies. During 2003, a CMU/SEI's report [20] extended the Hissam, *et al.*, work [19], describing how component technology can be extended in order to achieve Predictable Assembly from Certifiable Components (PACC). SEI's approach to PACC is PECT.

According to [21], the position of the certification in CBSE is shown in Figure 1. The role of the component certification is impressive through the first and second stags of CBSE activities. The requirements are specified and the right component is chosen. That is done according to some considerations that are specified through component certification which is stored in the repository of components. Certification process signifies the reliability and safety of software product in a manner that it can be analyzed by an independent authority with the help of some tools and techniques used in the certification process itself [25].

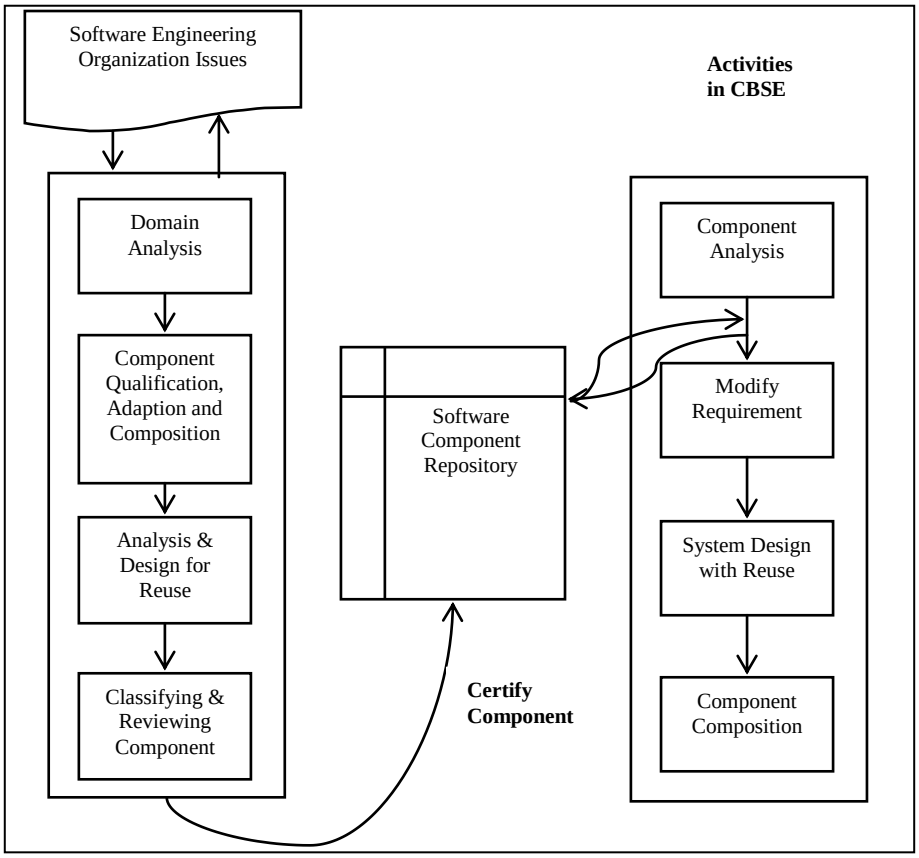

Figure 1. Position of Certification in CBSE

4. Quality Model

After analyzing this study and the ISO 9126, we developed the model. Table 1 shows the component quality model proposed, which is composed of seven characteristics (see Figure 2), as follows:

- **Functionality:** This characteristic expresses the ability of a software component to provide the required services and functions, when used under specified conditions;
- **Reliability:** This characteristic expresses the ability of the software component to maintain a specified level of performance when used under specified conditions;
- **Usability:** This characteristic express the ability of a software component to be understood, learned, used, conFigured, and executed, when used under specified conditions;
- **Efficiency:** This characteristic express the ability of a software component to provide appropriate performance, relative to the amount of resources used;
- **Maintainability:** This characteristic describes the ability of a software component to be modified;
- **Portability:** This characteristic is defined as the ability of a software component to be transferred from one environment to another.
- **Component Replaceability:** This characteristic describes the ability of a software component to be replaced.

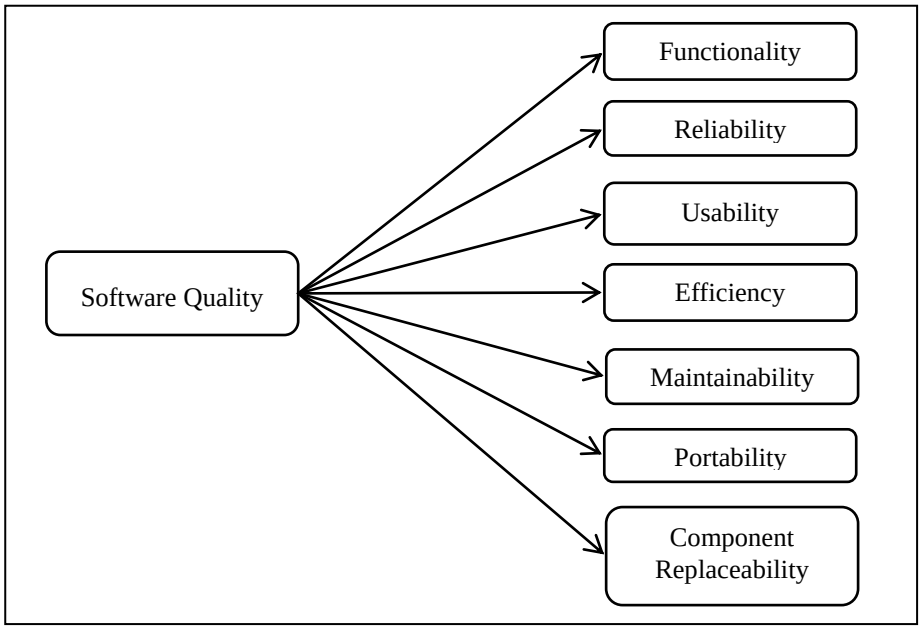

Figure 2. Software Quality Model

5. Level of Certification in Component-Based Software Engineering

After doing an exhaustive survey of certification in CBSE, we have identified that certification should not be performed at the end of the CBD process instead it should also be applied when we choose components from the repository or third party. So there should be two levels of the certification (refer to Figure 3);

(1) Certification at the Component Level
(2) Certification at the System Level

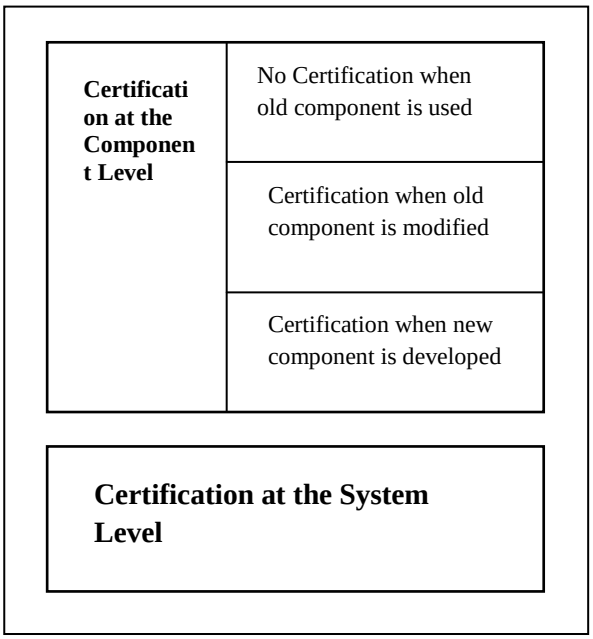

Figure 3. Levels of Certification in CBSE

5.1. Certification at Component Level

As we know that software productivity can be increased with software product reuse because reused software components need not to be developed from scratch. When we start CBD of software, we have three types of the component (refer to Figure 4);

(1) Reuse of a Component without Modification: The components of this category are used without modification. When these components are used in the software, the services provided by the component remains same as the services provided by the component when it was stand-alone software. In this case, the certification is not needed, because we have not altered the component at all. And we have assumed that the components are already certified by some authority.

(2) Reuse of a Component with Modifications: The components of this category are used after some modifications. The change in the service is a result of the change or adaptation of the component. Therefore, the services of the individual component must be changed, some new services are added. So a new certification must then be obtained based on the new services. The component has either to be certified with the expected functionality, or the reliability of the component must be derived.

(3) Developing a New Component: If a component is developed from the scratch, a certification must then be obtained based on the services *etc*.

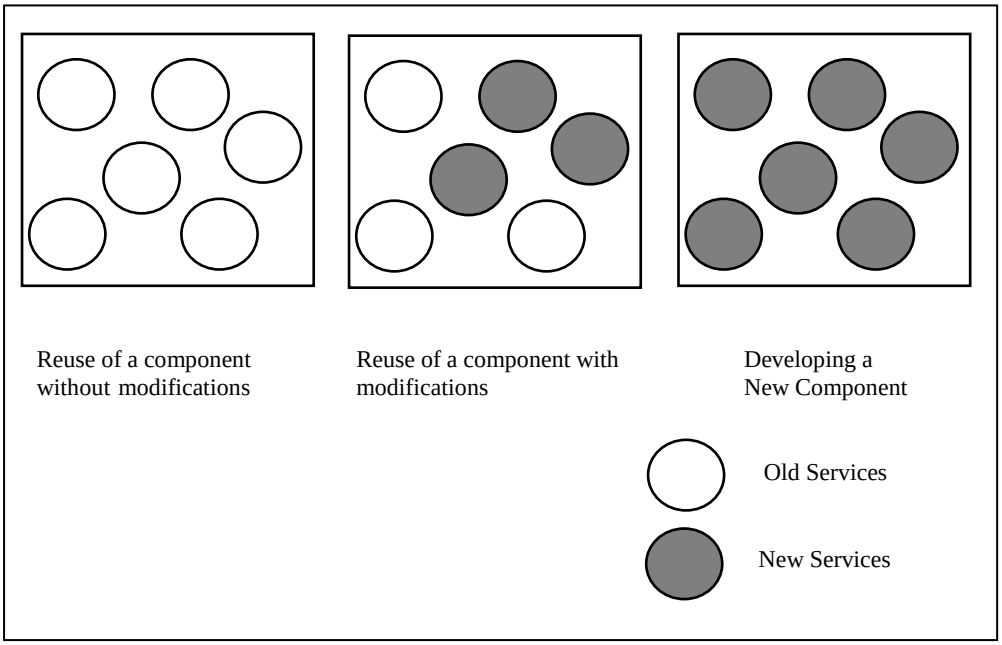

Figure 4. Three Types of Components

5.2. Certification at System Level

In this level, we are concerned about certifying the software as a whole.

6. Proposed Evaluation Process

The quality concept which makes the products in the center says that the quality of products is the sum of the related attributes and characteristics which can meet various demands. The attributes and characteristics which used by describing the quality of the software product often be called the software quality factors. As stated earlier we have seven characteristics for measuring the software quality.

Assign weights to these different parameters as per their roles in component-based software development. Suppose the weights are w_f , w_r , w_u , w_e , w_m , w_p , *and* w_{cr}

respectively. Where, $w_f + w_r + w_u + w_e + w_m + w_p + w_{cr} = 1$ (Sum of all weights is equal to 1). Quality of software is a linear function of theses seven factors, so

$$Quality = w_f * functionality + w_r * reliability + w_u * usability + w_e * efficiency + w_m * maintainability + w_p * portability + w_{cr} * component\ replaceability \quad \ldots\ldots\ldots eq(1)$$

We have five evaluation grades (depicted in below Figure 6):

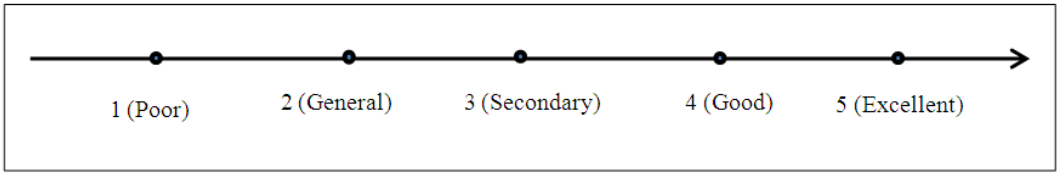

Figure 6. Certification Scale

6.1. Techniques for Weighting Criteria

A variety of techniques can be used to assign weights to criteria. All involve making a judgment based on understanding of the system. Three popular approaches include:

- **Unstructured Weighting:** One or more people determine weights based on their common understanding of the system and their experience. This is probably the most popular method, but not necessarily the best. We have also adapted it for assigning weights.
- **Delphi Technique:** Individuals use their own approach for deriving initial weights. The Delphi technique helps the team converge on a single weight and is a popular method for gaining team consensus.
- **Pair-wise Comparison:** A judgment is performed by comparing pairs instead of whole sets of criteria. Criteria are ordered in pairs, and the team agrees on the relative importance of the criteria in each pair. Pair-wise comparison is at the core of the Analytic Hierarchy Process (AHP). Tools such as Expert Choice support AHP and pair-wise comparison by computing weights for each criterion from the total set of pair-wise comparisons.

6.2. Measuring the Seven Factors

First of all to certify any component or software we need the requirement specification to match the testing results with required results. To measure these seven factors, we have taken five secondary quality factors (according to the features) for each factor. If the system fulfills a particular factor requirement then we will assign a value 1 to that factor otherwise 0. So each primary factor is assigned a value between 1 and 5. The Table 1 below depicts these secondary factors:

Table 1. Primary and Secondary Factors

Primary Factors	Secondary Factors
Functionality	Execution Time
	Input Range
	Resource Demand Precision
	Hardware Environment
	Software Environment
Reliability	Compute Reliability of the system/component
Usability	Documentation Available

	Documentation Quality
	Time and effort to understand
	Provided Interface
	Effort for Operating
Efficiency	Throughput (out)
	Processing Capacity (in)
	Memory Utilization
	Disk Utilization
	Process Capacity
Maintainability	Test suits provided
	Proof of components/software
	Extensive component/software test cases
	Extensibility
	Customizability
Portability	Configuration Capability
	Backward Compatibility
	Architecture Compatibility
	Modularity
	Crosscutting concerns level
Component Replaceability	Efforts needed for replacing the component

6.2.1. Functionality: We have defined the five factors as follows:

- **Execution Time:** This factor indicates whether the system provides the output within specified time. If, 'yes' then assign value 1 otherwise 0. For example, a specification can state that for a given environment an undisturbed execution takes between 4.9 and 5.1 ms, and the undisturbed execution takes 5 ms when executed, assign 1 to factor *execution time*.
- **Input Range:** The input range specifies the parameter ranges of parameterized specifications for which the specification is valid. For example, if a specification has the parameter file size and it is validated for file sizes between 3 and 50 MB this should be stated as input range.
- **Resource Demand Precision:** The resource demand precision states how exact resource demands of the specification are within the input range in relation to the implementation executed in the specified environment. The specification is considered valid as long as specified and measured resource demands are below the deviation threshold defined as resource demand precision.
- **Hardware Environment:** The hardware environment describes the hardware and its configuration for which the specification is valid. If a specification is validated for an environment it may also be valid for other environments.
- **Software Environment:** The software environment describes the software and it's configuration for which the specification is valid.

If the system satisfies all the five factors then functionality is assigned a total of 5, 1 for each.

6.2.2. Reliability: There are various methods available to measure the reliability of the system [24]. Here, we are not concerned about the method, what we only do is to convert the reliability to a number between 1 and 5. Suppose the reliability is 60%, then this factor is assigned value 3.

6.2.3. Usability: We have defined the five factors as follows:

- **Documentation Available:** If proper documentation is available then this parameter is assigned value 1.
- **Time and effort to understand:** This attribute tries to measure the time and effort needed to master some specific tasks (such as usage, configuration, administration, or expertise the component). If the time to understand is high then assign 0, else 1.
- **Provided Interface:** This parameter indicates whether the provided interface matches with required interface.
- **Documentation Quality:** If the documentation is good then it is assigned a value 1 otherwise 0.
- **Effort for Operating:** If the efforts required, for operating the software are high then assign a value 0 otherwise 1.

6.2.4. Efficiency: We have defined the five factors as follows:

- **Throughput ('out'):** This attribute measures whether the output that can be successfully produced over a given period of time.
- **Processing Capacity ('in'):** This attribute measures whether the amount of input information that can be successfully processed by the component over a given period of time.
- **Memory Utilization:** It indicates whether the amount of memory (needed by a component/software to operate) is valid according to specifications.
- **Disk Utilization:** This attribute specifies the disk space used by a component is valid according to specification.
- **Processing Capacity:** This attribute indicates whether the capacity of the component/software supports a vast volume of data with the same implementation. If 'yes' then assign 1 otherwise 0.

6.2.5. Maintainability: We have defined the five factors as follows:

- **Test Suit Provided:** This attribute indicates whether some test suites are provided for checking the functionality of the component/software. If 'yes' then assign 1 otherwise 0.
- **Proofs of Component/Software:** This attribute indicates if the component/software was formal tested, if 'yes' then assign 1 otherwise 0.
- **Extensive Component/Software Test Cases:** This attribute indicates if the component/software was extensive tested until be available to the market, if 'yes' then assign 1 otherwise 0.
- **Extensibility:** This attribute indicates the capacity to extend the component/software functionality. If it can be extended then assign 1 otherwise 0.
- **Customizability:** This attribute indicates whether the component or system offers customizable parameters.

6.2.6. Portability: We have defined the five factors as follows:

- **Configuration Capacity:** This attribute indicates the percentage of the changes needed to transfer a component to other environments; if changes are large in number then assign 0 otherwise 1.
- **Backward Compatibility:** This attribute is used to indicating whether the component is "backward compatible" with its previous versions or not, if it is then assign 1 otherwise 0.

- **Architecture Compatibility:** This attribute indicates whether the component/software depends on a specified architecture, if 'yes' then assign 0 otherwise 1.
- **Modularity:** This attribute indicates whether the component/software has modules, packages. If it has then assign 1 otherwise 0.
- **Crosscutting Concern Level:** This attribute indicates if the component code is interlaced (*e.g.*, business role code with interface code and SQL's code), becoming difficult its reusability. If component/software is interlaced then assign 0 otherwise 1.

6.2.7. Component Replaceabilty: This factor indicates the efforts required to replace a particular component. Effort may be low, moderate, average, high, and very high. Assign values 1 to low, 2 to moderate, 3 to average, 4 to high, and 5 to very high.

After assigning values to these secondary factors, we will compute the quality of the component/software by using equation 1.

7. Example

Let us assign arbitrary values of weights as follows,

w_f =0.20, w_r =0.25, w_u = 0.10, w_e = 0.10, w_m =0.10, w_p=0.15, and w_{cr} = *0.10*

Below is the matrix, depicts the values assigned to the secondary parameters after performing evaluation (Table 2). These values are taken for illustrating the proposed approach.

Table 2. Values Assigned

Factors \ Values						Total
Functionality	1	1	1	0	0	3
Reliability	80%					4
Usability	1	1	0	0	0	2
Efficiency	0	1	1	0	0	2
Maintainability	1	0	0	1	0	2
Portability	1	0	0	1	1	3
Component Replaceability	3					3

So the quality of the software will be given by

Q = 0.20 * 3 + 0.25 * 4 + 0.10 * 2 + 0.10 * 2 + 0.10 * 2 + 0.15 * 3 + 0.10 * 3

= 0.60 + 1.00 + 0.20 + 0.20 + 0.20 + 0.45 + 0.30

= 2.95

So the software is given the certificate ***"Secondary"***.

8. Conclusion

Component certification is a process to ensure that the software components conform to precise standards. In general, the ultimate goal of certification is to bring reliability, quality and compatibility to a certain software product. Certification process signifies the reliability and safety of software product in a manner that it can be analyzed by an independent authority with the help of some tools and techniques used in the certification process itself. We propose a framework to estimate the quality of software components as well as component-based software system in a proficient way. We have proposed an evaluation criteria which is based on seven primary factors; functionality, reliability, usability, efficiency, maintainability, portability, and component replaceability. The

values for these seven primary factors are obtained from secondary factors for each factor. The quality of the system is a linear function of these seven factors. In future, we can also consider adopting other quality factors to the proposed quality model in order to give most appropriate results.

References

[1] J. Voas and J. Payne, "Dependability certification of software components", Journal of Systems and Software, vol. 52, no. 2–3, **(2000)** June, pp. 165–172.
[2] J. Morris, G. Lee, K. Parker, G. A. Bundell and C. P. Lam, "Software component certification", Computer, vol. 34, no. 9, **(2001)** September, pp. 30–36.
[3] M. Goulao and F. B. e Abreu, "The quest for software components quality", Proceedings 26th Annual International Computer Software and Applications, **(2002)** August, pp. 313–318.
[4] "Information Technology – Product Quality – Part1: Quality Model", ISO 9126, International Standard Organization, **(2001)** June.
[5] R. P. S. Simão and A. D. Belchior, "Quality Characteristics for Software Components: Hierarchy and Quality Guides", in Lecture Notes in Computer Science, Springer Science + Business Media, **(2003)**, pp. 184–206.
[6] K. C. Weber and C. J. do Nascimento, "Brazilian software quality in 2002", Proceedings of the 24th International Conference on Software Engineering. ICSE, **(2002)**, pp. 634–638.
[7] M. Bertoa and A. Vallecillo, "Quality Attributes for COTS Components", 6th International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE), Spain, **(2002)**.
[8] H. Schmidt, "Trustworthy components—compositionality and prediction", Journal of Systems and Software, vol. 65, no. 3, **(2003)** March, pp. 215–225.
[9] R. Seacord and M. Bass, "Building Systems from Offthe- Shelf Components", in Software Architecture in Practice, Second., Addison Wesley, **(2003)**.
[10] J. Stafford and K. C. Wallnau, "Is Third Party Certification Necessary?",, 4th Workshop on Component-Based Software Engineering (CBSE), Lecture Notes in Computer Science (LNCS) Springer-Verlag, Canada, **(2001)**.
[11] A. S. Andreou and M. Tziakouris, "A quality framework for developing and evaluating original software components", Information and Software Technology, vol. 49, no. 2, **(2007)** February, pp. 122–141.
[12] A. Alvaro, E. S. de Almeida and S. L. Meira, "Towards a Software Component Certification Framework", Seventh International Conference on Quality Software (QSIC), **(2007)**, pp. 298–303.
[13] R. Land, A. Alvaro and I. Crnkovic, "Towards Efficient Software Component Evaluation: An Examination of Component Selection and Certification", 34th Euromicro Conference Software Engineering and Advanced Applications, **(2008)** September.
[14] T. Janner, "A Core Component Based Modeling Approach for Achieving E-business Semantic Interoperability", Journal of theoretical &applied E-commerce research, vol. 31, no. 3, **(2008)**, pp. 1–16.
[15] T. Sen and R. Mall, "State-Model-Based Regression Test Reduction for Component-Based Software", ISRN Software Engineering, vol. 2012, **(2012)**, pp. 1–9.
[16] A. Alvaro, E. S. Almeida and S. R. L. Meira, "Component Quality Information Provided by Software Component Markets and a Brazilian Software Factory", 5th International Conference on Quality Software (QSIC), **(2005)**.
[17] H. Groenda, "Certification of Software Component Performance Specifications", FZI Forschungszentrum Informatik, Software Engineering, Karlsruhe, Germany, **(2000)**, pp. 10–14.
[18] M. Woodman, O. Benebiktsson, B. Lefever and F. Stallinger, "Issues of CBD Product Quality and Process Quality", Proceedings of the 4th ICSE Workshop on Component-Based Software Engineering (CBSE), Canada, **(2001)** May.
[19] S. Hissam, G. Moreno, J. Stafford and K. Wallnau, "Enabling predictable assembly", Journal of Systems and Software, vol. 65, no. 3, **(2003)** March, pp. 185–198.
[20] K. C. Wallnau, "Volume III: A Technology for Predictable Assembly from Certifiable Components", Software Engineering Institute, **(2003)**.
[21] L. K. Ahmed, "Role of Component Certification in CBSE Activities for Building High Quality Software", World of Computer Science and Information Technology Journal (WCSIT) ISSN: 2221-0741, vol. 5, no. 1, **(2015)**, pp. 11-15.
[22] L. Nautiyal and N. Gupta, "ELICIT – A New Component Based Software Development Model", International Journal of Computer Applications, vol. 63, no. 21, **(2013)** February, pp. 53-57.
[23] L. Nautiyal and N. Gupta, "ELITE PLUS- Component Based Software Process Model", International Journal of Computer Applications, vol. 90, no. 5, **(2014)** March, pp. 1-7.
[24] L. Nautiyal, N. Gupta and S. C. Dimri, "Measurement of Reliability of a Component-Based Development Using a Path Based Approach", ACM SIGSOFT Software Engineering Notes, vol. 39, no. 6, **(2014)** November, pp. 1-4.

[25] L. Nautiyal and N. Gupta, "A Contemporary Certification Process for Component Based Development", International Journal of Computer Technology & Applications, vol. 6, no. 1, **(2015)** February, pp. 127-132.

