Egyptian Computer Science Journal ,ECS ,Vol. 37 No. 1, January 2013 ISSN-1110-2586

Developing a Software Certification for Quality Assurance

Mohamed Shaheen Elgamel[1], Ezzat A. Korani[2] and Kariman R. El Helow[2]

[1]Arab Academy for Science, Technology, and Maritime Transport, Alexandria, Egypt,
[2]Institute of Graduate Studies and Research, Alexandria University, Egypt
cshaheen@hotmail.com, kariman_ramzy@yahoo.com,

Abstract

One of the major problems facing the software development is the lack of direct measures for activities that are carried out in organizations. These measures not only help to improve the project standard but also minimize the time, labor and fiscal budget of the project. Fuzzy logic has long excelled at delivering exact results from imprecise or ambiguous information, and its primary use has been in controllers. A fuzzy rule-based model for measuring, analyzing, and certifying software development is proposed. The proposed model uses fuzzy logic to combine measured attributes into a model which integrates the two existing process and product models, (SCfM_prod), and (SPAC).

Keywords: *Software certification model, Process approach, Product quality model,Software metrics, Fuzzy logic model.*

1. Introduction

Software quality issues are considered the main assets by means of which a firm can enhance its competitive global position. This is one reason why quality has become essential to ensuring that a company's products and processes meet users' needs. As the methods for certifying software quality continue to multiply, process-based approaches such as International Organization for Standardization (ISO 9000) and Capability Maturity Model (CMM) make software publishers take decisions concerning which development standards and processes they will use [1]. Software companies spend 60% of their IT budget on 'running the business' and only the remaining 40% on 'growing and transforming' it. However, those using measurements can cut their IT expenditures on 'running the business' to less than 50%, thus leaving more than 50% for 'growth and transformation' which can make a significant difference to their business [2].

Software Quality Assurance (SQA) teams agree that the quality assessment of software will continue to deteriorate if it is done through testing the software alone. Furthermore, Software quality is measured through static assessment of code's structure. So, software quality must be more than just static features, it should also comprise non-functional such as behavioral and human aspects [3].

The certification is a "procedure by which a third party gives a written assurance that a product, process or service conforms to specified characteristics" [4].

The fuzzy logic approach is a procedure that consists of analyzing and defining problems, creating sets and logical relationships, converting information to what are called fuzzy sets, and interpreting all parts of the model. It is a mathematical tool for dealing with uncertainty and also to provide a technique to deal with imprecision and information granularity. It can use a number of criteria to determine whether a fuzzy-logic approach would lend itself to solve a specific problem by specifying prerequisites. These prerequisites include the level of ambiguity of the data (determined mathematically) and the required accuracy of the output [5].

This paper shows an improvement through verifying an effort in software product certification process; particularly in outlining an approach through fuzzy logic to merge measured attributes and determine the quality of a software product. The attributes are Usability, Portability, Functionality, Integrity, Security, Correctness, Accuracy, Consistency, Maintainability, Reliability, and Efficiency. A rule base is prepared based on merging estimations of these attributes. This is done through the merging of the two models, Software Certification Model by Product (SCfM_prod), and Software Process Assessment and Certification Model (SPAC). The rest of the paper is organized as follow:
The literature survey section explores the previous models for the certification of software quality. The methodology section describes the proposed model. The experimental environment section describes the use of the questionnaire and the equations to measure the quality attributes. The results and discussion section describes the test and the implementation in the real environment. Finally, we conclude the paper in the conclusion section.

2. Literature Survey

It is important nowadays, that the subject of the certification to be safety-related to the system or system function. This task is accomplished through hardware and software used for this purpose. There are several models for software quality certification [6-10]. The first model is named (SPAC) [6], [7] as shown Figure (1).

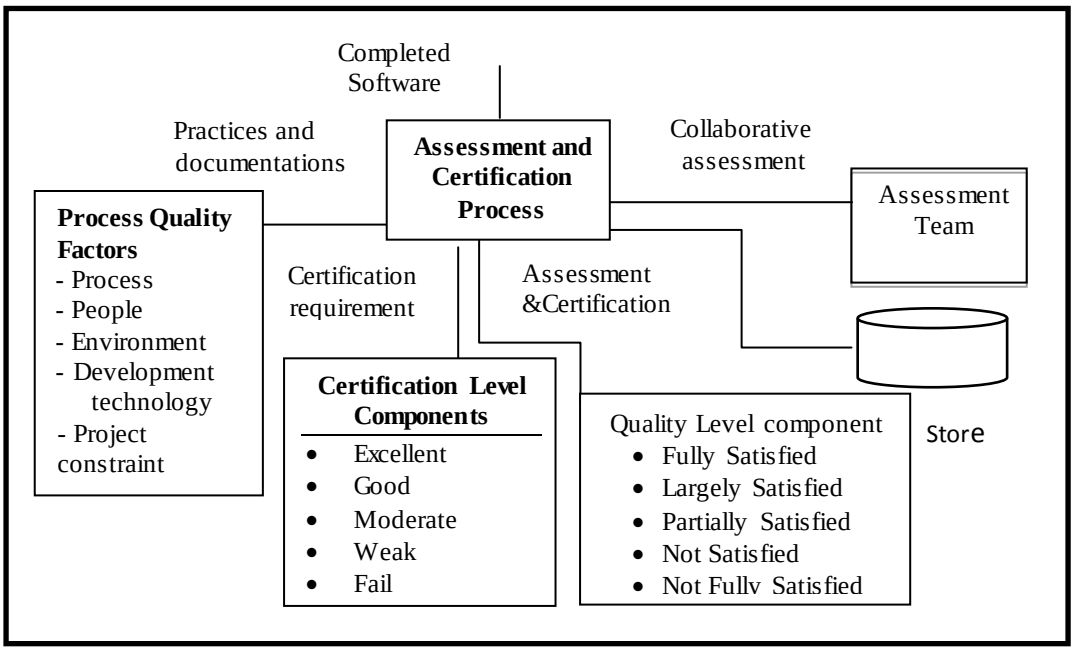

Figure1. SPAC model

SPAC consists of seven components:

The first component is the Process Quality Factor (PSQF). It identifies factors that affect the quality of the software process. The five factors are: process, people, environment, development technology and project constraint.

The second component is the candidate software to be assessed. This candidate is a completed product that is ready to be delivered to users or customers.

The third component is the assessment team.

The fourth component is the assessment and certification process.

The fifth and sixth components are the quality and certification level. The seventh component is the repository, which stores all information and results from assessment and certification exercises. The advantages of this model are:

1- The implementation of this model can be conducted several times during its life span.
2- The owners of the software are able to monitor the progress and performance of their software product operating in certain environment.
3- This certification environment supports a continuous improvement during the life span of the product.

The disadvantages of this model are:

1- The maintainability will be difficult and complex.
2- The assessed product is weak in terms of usability.

The second model is named (SCfM_*prod*) [8], [9], as shown Figure (2).

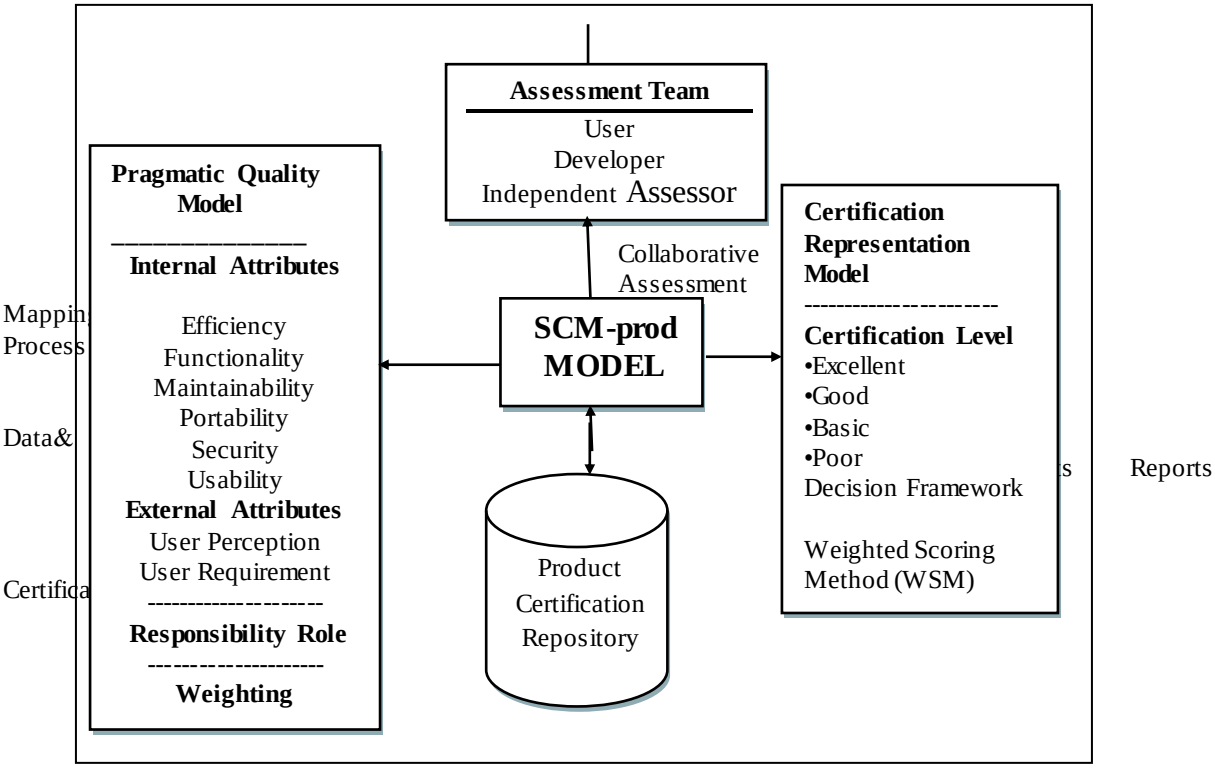

Figure2. SCfM_*prod model*

SCfM_prod focuses on the certification model through the product quality perspective. The SCfM_*prod*model consists of four main components:

The first component is the pragmatic quality factor (PQF) which is the quality certification guideline and standard for measuring software product quality. There are two sets of attributes, namely:

1) The behavioral attributes, which include usability, efficiency, functionality, maintainability, portability, and reliability.

2) The Impact Attributes are referring to the human aspect of quality toward the product.

The second component in this model is the Certification Representation Method which is elaborated in three main sections related to weight scoring method, certification level and decision process.

The third component is Product Certification Repository Which stores data and reports of software product certification candidates.

The fourth component is The Assessment Team; this model applies a collaborative perspective assessment between user, developer and independent assessor.The advantages of this model are:

1- Eliminates bias assessment and evaluation of the product by including an independent assessor in the team.
2- Removes unfair evaluation by including the owner or users of the product to participate in the assessment process.
3- Accelerates the process because the team is familiar with the product and its environment.
4- Protects data confidentiality and privacy by only permitting users to have direct access to the software.

The disadvantages are:

1- It is a static model of quality.
2- This quality model is unable to improve its components or characteristics according to current and future requirements.
3- The model also may not be able to handle multiple assessment and certification exercises easily and efficiently.

Previous attempts to measure complexity relied on sharp boundaries between what is and what is not complex [10]. The fuzzy approach more closely models the way that managers think in degrees such as somewhat, moderately, and highly complex. Fuzzy logic is a multi-valued logic that allows for degrees (e.g., normal versus slow or fast) of set membership—a more practical way to deal with the issues you face in the real world. Unlike binary (yes or no) information, fuzzy logic emulates your ability to reason and make use of approximate data to find precise solutions [10].

3. Methodology

The proposed model merges measured attributes through fuzzy logic to a new model which compiles two models, (SCfM_prod), and (SPAC). The proposed model covers both human and technical aspects, and thus provides better balance in software quality assessment. The approach consists of two parts: The first part is the software product, and the second part is the software process.

The first part consists of the pragmatic quality which is the quality certification guideline and the standards for measuring software product quality. It includes two sets of attributes, namely the behavioral and the impact attribute, Figure (3), and these attributes are: internal and external attributes, and responsibility role. The internal attributes (The Behavioral Attributes) are the indicators of a metric or a combination of metrics that provide insight into the product. The indicators here include efficiency, functionality, maintainability, portability, reliability, usability, correctness, accuracy and consistency, security. Integrity, are not included in ISO 9126 model but included in the proposed model, this attribute measures the ability to with-stand an attack to its security that comprises the program, data and document. It covers threat and security aspects.

The external indicators indicate the conformance in user requirements and perception of a particular product to the user experience. The external attributes (The Impact Attributes) that are defined in PQF, are important to balance the quality model between the technical measurements of the software and the human factor. PQF refers to the human aspect of quality toward the product. It illustrates the impact of the software in terms of quality to the users and also measures the conformity of software to the user requirements. The three external factors are:

User perception**:** is defined bydelivery on-time and within budget, law and Regulation, and Expectation, Environmental adaptability.

User requirement:specifies what functions the system has to fulfill.

Responsibility role**:** It is defined as the responsible person to answer the questions related to metrics. ThePQF has identified specific interviewees to be responsible in giving the assessment score of each metric. The interviewee could be the user, developer, independent assessor or a combination of these interviewees.

A Model has been developed so that it can be used to validate the software products, tested and implemented in the real environment. It has adopted weighted scoring method (WSM) which applies different levels and categories of attribute with different measure factors. Each software quality attribute must not have the same level of importance in the real world environment to represent the actual business requirements, in order to meet the needs of groups with different interests related to software quality. The proposed model incorporates the fact that there are some degrees of importance of each quality attribute. The measure factor of the product is a factor that influenced the certification level of a software product. Thus, this model accommodates measure factors for all attributes with different level of importance to reflect individual business requirements.

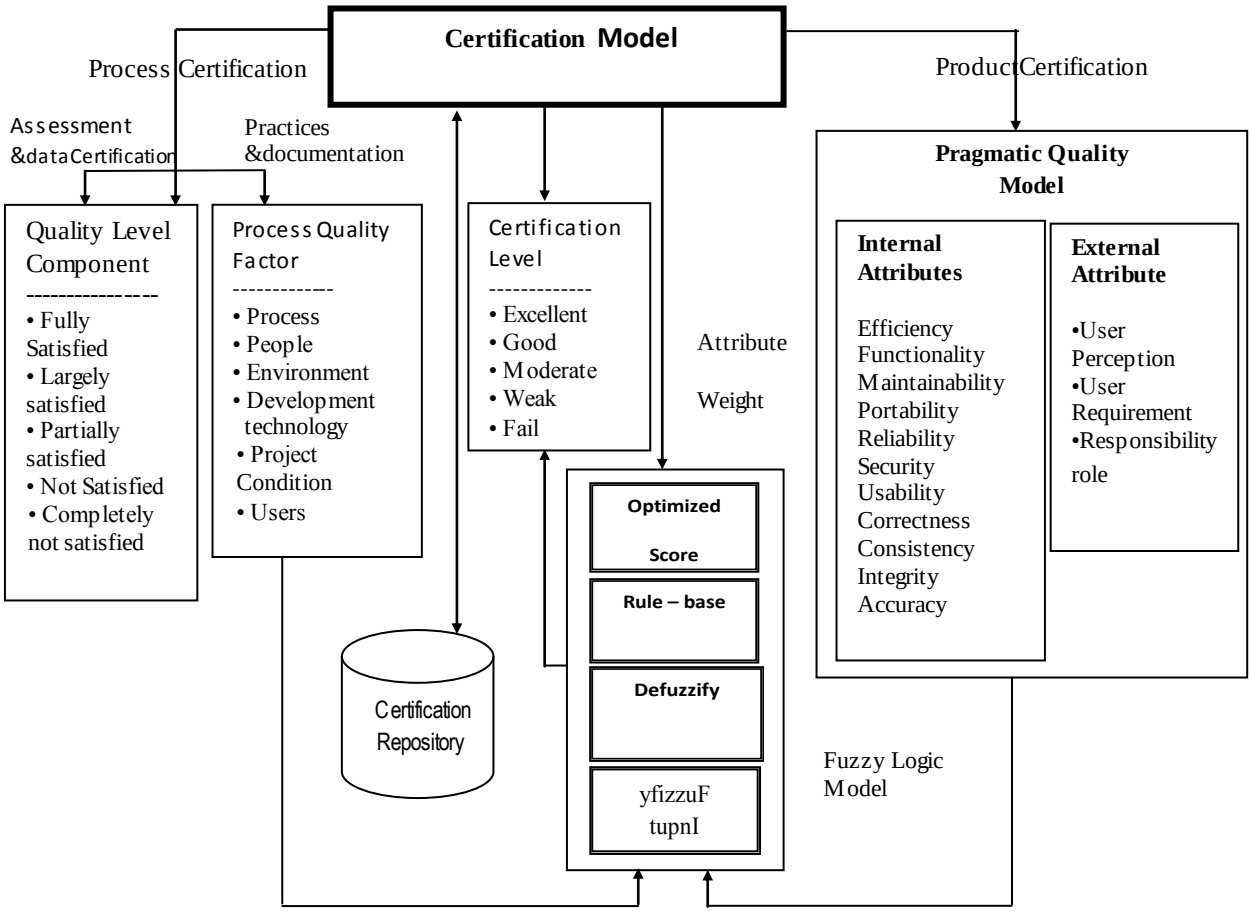

Figure3. NewModel for software certification

A questionnaire is used to measure the software process attributes. The case study was conducted on one of the major organization. A structured questionnaire with 32 questions was used in our case. The questionnaire was divided into six sections.About 45 sets of questionnaires were distributed to one of the major organizations that work in the software. Out of this amount, only 38 were returned, 9 sets without answers, 4 sets were not completed and. This left us with only 25 sets which were completely answered and were used for analysis.

We understand that this is a small set, but it was enough to feed our model with practical data and to validate it on real environment beside the data that we used from other published work. The set of questions asked are used to assess the view of the respondents. The questions are easy and direct in order to be clear to the users and do not need explanations. In the survey, respondents were asked to indicate the level of their satisfaction by 1= completely not satisfied, 2= not Satisfied, 3= partially satisfied, 4= largely satisfied and 5= fully satisfied of all the quality attributes, according to the Likert scale from 1 to 5 Likert scale [11],[12] is defined as something that measure the satisfaction based on perception. Each degree of

agreement is given a numerical value from one to five. Thus a total numerical value can be calculated from all the responses. The scale used in this approach is recommended to 1= Fail, 2= Weak, 3= Moderate, 4= Good, 5= Excellent. The measuring attributes defined in this model are based on findings from (case study). These measures are taken into account during assessment exercise of software product in the organization. Then, attributes are sorted in these classifications which will be used in the certification of the proposed factors. SPSS is used to conduct data management and analysis, and the measure of each attribute is calculated using SPSS. The classification level is shown in Table1. The following is the list of the selected attributes and analysis of the SPSS results.

- Process**:** The factor of process includes three basic activities, which are development, management and support activities.
- Development process, – the analysis shows that in this factor, obtain level of partially satisfactory in the quality assessment level.
- Management process: in this factor, there are five attributes, which consist of project management, change management, quality management, technical management, and risk management. Each attribute obtains a level of satisfactory except the quality management obtains level of partially satisfied.
- Support process: consists of resource management, training management, staff affair and documentation.All attributes in this factor achieve level of satisfaction. This factor obtains level of partially satisfied.

•People**:** This factor is measured in terms of skill, experience, knowledge, team commitment, user involvement and management responsibility. (In this factor all attributes obtain a level of no satisfaction). This factor obtains level of not satisfied.

• Environment**:** This factor measures the comfort ability and safety aspects in the work place. This factor obtains level of fully satisfied in this assessment.

• Development technology**:** This factor is measured in terms of standard and procedure, tools, methods and techniques and process origin. (This attribute obtains level of partially satisfactory in this assessment, while in the attribute of tool and technique, the quality level obtained is partial satisfaction). This factor obtains level of partially satisfied.

• Project constraint: This aspect of quality measures the time delivery and budget; both attributes obtain level of partial satisfaction and largely satisfactory respectively). This factor obtains level of largely satisfied.

• Users**:** Users may select their interested attributes of quality to meet their organizational requirements and targets. This offers flexibility in the certification exercise. This factor obtains a level of not fully satisfied.

Table 1: The classification level

Score (*F*)	Certification Level	Certification Status	Certification Description
0.8<= F <=1	5	Excellent	Software fully satisfied all quality criteria and achieves quality level of excellent.
0.6<= F< 0.8	4	Good	Software quality is largely satisfied.
0.5<= F < 0.6	3	Moderate	Software quality is partially satisfied, which also means average and Moderate.
0.4<= F < 0.5	2	Weak	Software quality is not satisfied.
0 <= F < 0.4	1	Fail	Software quality is completely not satisfied.

The second part of proposed model is the software product, which includes eleven attributes. These attributes include efficiency, functionality, security, maintainability, portability, reliability, usability, correctness, accuracy, consistency, and Integrity; the following equations are used to measure the features of software products [13], [14]:

$$\textbf{Correctness = Defects per KLOC} \quad (1)$$

Where:
KLOC means thousands (Kilo) Of Lines of Code.) .

$$\textbf{Integrity} = \Sigma\ [(1 - \textbf{threat}) \times (1 - \textbf{security})] \quad (2)$$

Where:
Threat = probability that an attack of certain type will happen over a period of time.
Security = probability that an attack of certain type will be removed over a period of time.

$$\textbf{Security (MTTSF)} = h_G p_a^{-1} + h_V p_a^{-1} + h_A + p_m h_{MC} + (1 - p_m - p_u)\, h_{TR} / 1 - p_m \quad (3)$$

Where:
p_a probability of injecting a successful attack, given that the system was vulnerable
p_u probability that a successful attack has remained undetected
p_m probability that the system successfully masks an attack
h_G mean time for the system to resist becoming vulnerable to attacks
h_V mean time for the system to resist attacks when vulnerable
h_A mean time taken by the system to detect an attack and initiate triage actions
h_{MC} mean time the system can keep the effects of an attack masked
h_{TR} mean time the system takes to evaluate how best to handle an attack
$1 - P_m - P_u$ = TR (the probability that an attack will be detected and the system will enter the TR state (triage state)

Defect Removal Efficiency (DRE) = E / (E + D) **(4)**

Where:
E = Number of Errors found before delivery of the software.
D= Number of Errors found after delivery of the software.

Reliability = (MTBF / MTTR) **(5)**

Where
MTBF = Mean time between failure
MTTR = Mean time to repair

Maintainability = 1 / (1 + MTTR) **(6)**

Where:
MTTR = Total amount of Repair Time spent in a specified period (hours) / number of repair events in that specified period.

7-Usability: it means Effectiveness

Effectiveness = Temporal Efficiency * Task Time **(7)**

Where:
Effectiveness: This metric is derived from measures of the quantity and quality of task output and measures whether users succeeded in achieving their goals when working with a system.
Temporal Efficiency: This relates effectiveness to the task time and thus measures the rate of task output.
Task Time: total time required for each task under study.

Accuracy=1/n $\sum_{i=1}^{n} |act_i - est_i|$ **(8)**

Where:
n is number of projects
act_i is the actual observed value, (i= 1,2,3……,n)
est_i is the estimated value, (i= 1,2,3……,n)

9-Consistency (SDR)

$$SDR = \sqrt{\frac{\sum_{i=1}^{n}(r_i - \bar{r})^2}{n-1}} \quad (9)$$

Where:
r_i = est./act. (i = 1.2,3.....n)
$\bar{r}$ is mean r_i (i = 1.2,3.....n)

Functionality FP = UFB × TCF **(10)**

Where:
UFP = unadjusted function points
TCF = technical complexity factor

11-Portability

1- Quiescent or static Power

P = 1*E or PDQ= IDD*VDD **(11-1)**

Where:
PDQ = Quiescent Power Dissipation
IDD = Device static current taken from the datasheet
VDD = Voltage applied to the device power terminals

2- Internal Dynamic Power

PDINT = (CPD*VC*F) VDD **(11-2)**

Where:
PDINT = Internal Power Dissipation
CPD = Internal Device Power
VS = Internal Voltage Swing
F = Switching Frequency
VDD = Voltage applied to the Device power Rail

3- External Dynamic Power

PDEXT = (CEXT*VS*F) VDD **(11-3)**

Where:
PDEXT = External Capacitive Load Power Dissipation
CEXT = Aggregate External Capacitance Presented at the Device output
F = the Output Switching Frequency
VDD = the Voltage Applied to the Power Rails

4- Total Power Dissipation

PTOTAL = PQD+PINT+PEXT **(11-4)**

= ((CPD+CEXT)*VDD2*F)) + (IDD*VDD)

In previous models, such as Aggarwal[15] and K. Seth and Others [10] we find that:

Aggarwal study suggests a four parameter integrated maintainability of software using the fuzzy model, this lead to the formation of 81 of the rules of fuzzy model.

K. Seth and Others study proposes a fuzzy logic based approach selecting the best five factors that affect the Component Selection Efforts. The output variable i.e. Selection effort has five membership functions. These lead to the Formation of 243 rules for the fuzzy model.

So, the present model proposes a fuzzy logic based approach to merge the seventeen measured software product and software process attributes mentioned above. In the proposed model, a new method which does not have to take all the combinations of attributes values is proposed. It uses less number of inputs based on the rules of division by using Fuzzy logic.

We used the Matlab fuzzy system, and considered all of the 17 attributes as inputs and provided a crisp value of combination measuring attributes using a Rule Based system. The domain of attributes is fixed, (usually the set of real numbers, and whose range is the span of positive numbers in the closed interval [0, 1]. All inputs can be classified into fuzzy sets viz. The output combination measuring attributes are classified as Excellent, Good, Moderate, Weak, and Fail. All possible combinations of the measured attributes are considered to design the rule base. This is the step that converts the input data into linguistic variables with the usage of membership functions which can be numerical or functional. This process is known as fuzzification. These attributes are then processed in fuzzy domain by inference engine based on knowledge base (rule base and data base). Using the rule viewer, output i.e. combination measuring attributes are observed for a particular set of inputs using the MatLab Fuzzy tool box. And finally the process of converting back fuzzy numbers into single numerical values is called defuzzification. The proposed rules are as described below. If we follow a regular decision tree, we will have a very large number of rules using Likert scale from 1 to 5. However, using these 12 smart rules decreased it significantly.

The proposed rules are:

- If (<=1/3 the number of attributes is low) and (>2/3number of attributes is excellent) then certification level is good.
- If (<=1/3 the number of attributes is low) and (>2/3 number of attributes is good) then certification level is moderate.
- If (<=1/3 the number of attributes is low) and (>2/3 number of attributes is moderate) then certification level is weak.
- If (<=1/3 the number of attributes is low) and (>2/3 number of attributes is weak) then certification level is fail.
- If (<=1/4 the number of attributes is low) and (>3/4 number of attributes is excellent) then certification level is good.
- If (<=1/4 the number of attributes is low) and (>3/4 number of attributes is good) then certification level is weak.
- If (<=1/4 the number of attributes is low) and (>3/4 number of attributes is moderate) then certification level is fail.
- If (<=1/4 the number of attributes is low) and (>3/4 number of attributes is weak) then certification level is fail.
- If (<=1/2 the number of attributes is low) and (>1/2 number of attributes is excellent) then certification level is moderate.
- If (<=1/2 the number of attributes is low) and (>1/2 number of attributes is good) then certification level is weak.
- If (<=1/2 the number of attributes is low) and (>1/2 number of attributes is moderate) then certification level is fail.
- If (<=1/2 the number of attributes is low) and (>1/2 number of attributes is weak) then certification level is fail.

The second method divide the inputs so that the result is a clearer and more accurate, which provides the organization or a skilled developer of the software the knowledge of any department needs to improve his performance. We have designed a new system that classifies the input into 4 main categories each is targeting a department in the manufacturing as shown in Table 2. The analysis and classification leads to three distinct categories as shown in Table 3.

Table 2: The classification level

Process	Process, People, Environment, Development technology, Project, users
Function	Functionality, Efficiency, Reliability, Usability, Correctness, Consistency,Accuracy
Security	Security, Integrity
Maintenance	Maintainability, portability

Table 3: The classification level (in the second method)

Level	Measures
Low	0 - .4
Moderate	. 4 - .7
High	.7– 1

Then, features are sorted in these classifications which will be used in the certification of the proposed rules following:

- If (process is low) and (security is low) and (function is low) and (maintenance is high) then certification level is moderate.
- If (process is low) and (security is low) and (function is high) and (maintenance is low) then certification level is high.
- If (process is low) and (security is high) and (function is low) and (maintenance is low) then certification level is moderate.

There exists 81 rules or there exist 81 different if statements that are considered. This method uses the original 12 rules applied to 4 different sets of attributes. In addition, 81 more rules were added. This leads to a total of 129 rules.

Software quality attributes do not have the same level of importance. It depends on the organization's requirements and the expectations that meet the needs of groups with different interests related to software quality.

The simplest and valid way to consider the relative merits of the attributes in the proposed model is to virtually increase the number of attributes with higher merits keeping its original measured value. This means that an attribute which has a higher weight will be presented more than one time depending on its relative merits. This will not lead to any change in the number of rules we have in the proposed model. We derive a counter example to explain our solution. We still have the original 12 rules and a set of 17 attributes with different values. Assume that we decided that the maintainability attribute is of more merit to the system. In this case, we assume that the maintainability has a "low" value. We will increase the number of attributes with "low" value by m. The new total number of attributes will by 17+m where m>=1 depending on how important the maintainability attribute is. This simple technique can be applied to more than one attribute at the same time.

4. Results

This section reports and analyzes the results from the experimental evaluation to the proposed model of the software certification. Data from real projects were collected and analyzed. We were able to gain access permission to data from one of the major organizations that work in software industry for government buddies in the business sector. It is summarized in Table 4. A fuzzy logic system was built taking into account that the proposed model combines attributes measured by the fuzzy logic to a model which integrates the two existing process and product models, (SCfM_prod), and (SPAC), by providing a larger number of attributes with the lowest number of rules.

Table 4:Certification results for conducted software projects

P 4	P 3	P 2	P 1	Project
.61	.71	.63	.68	Correctness
.56	.45	.39	.54	Security
.79	.77	.66	.75	Integrity
.82	.65	.70	.855	Reliability
.42	.50	.67	.59	Portability
.70	.68	.59	.72	Consistency
.90	.76	.68	.82	Accuracy
.67	.66	.59	.769	Usability
.51	.41	.50	.166	Maintainability
.77	.70	.78	.97	Efficiency
.91	.82	.72	.95	Functionality
.63	.69	.66	.8	Environment
.60	.55	.54	.61	Project
.55	.53	.64	.59	Development Technology
.43	.45	.62	.50	Process
.39	38.	.54	.49	People
.35	.34	.51	.20	Users
83.6%	80.8%	80.2%	86.25%	Certificate

Based on the results shown in Table 4, the certification level of the product is determined by comparing the score value ((F) Fuzziffication)) obtained in the certification exercise, according to Table 1.

From the analysis, the approach is used to group and classify attributes into five distinct classifications namely Fail, Weak, Moderate, Good, Excellent. Then, the attributes are sorted into these classifications; the analysis shows that efficiency is (97%) obtains the highest in this analysis, functionality (95%), reliability (85.5%), accuracy (82%), environment (80%). These five attributes (efficiency, functionality, reliability, accuracy and environment) are classified in the classification group of excellent (these attributes are the more effects to certificate). Second group of classification defined as good includes usability (76.9%), integrity (75%), consistency (72%), correctness (68%) and project (61%). On the other hand, the third group of classification defined as moderate includes portability (59%), development technology (59%), and security (59%). Process (50%), people (49%) are weak. Users (20%) and maintainability (16.6%) are fail (these attributes are less effects to certificate). Following the assessment and obtaining the quality level of all attributes, the next step is to compute the certification level. The certification in this case obtains a score of .8625, which is at highest level and equivalent to excellent.

The proposed certification model is flexible and scalable to accommodate different number of process and product attributes. To validate the proposed system, it was employed to generate the certification for different set of attributes as provided by four different previous systems [7], [8], [9], [16]. Our proposed model results were compared to others as summarized in Table 5.

Table 5: Comparison with other models

Attributes	[9]	[8]	[16]	Attributes	[7]
Efficiency	7	.746	7	Process	7
Functionality	10	.724	9	People	6
Maintainability	7	.678	7	Technology	4
Portability	4	.64	4	Project	9
Reliability	8	.66	9	Environment	8
Usability	6	.64	7		
Integrity	10	.734	10		
user	-	.706	-		
Others' models certificate	69%	70%	74%		65%
Our model certificate	68%	67%	68%		60%

The case study in [7] was conducted in a semi-government organization in Malaysia. The assessed system is a large system and operating in the headquarters as well as integrated to various branches throughout Malaysia. The results are incorporated in column [7] in table 5. It is noticed that the reported parameters are small in number and different than the set of parameters used by other authors or by our model. However, the proposed model succeeded to deliver a certification level very close to the reported results.

The case study in [8] was carried out in local companies and organizations in Malaysia. The data collection was conducted through a collaborative perspective assessment among members in the team which consists of the independent assessor, developer and users. Following the data collection, data analysis was conducted. Their results are in incorporated in column [8] in Table 5.

The case study in [9] was carried out with a semi-government corporation (TH Corporation) and one of the systems in the corporation was selected to be assessed and certified. The certification process was applied separately after the assessment was completed. The obtained score in the assessment exercise done by the authors is presented in column [9] in table 5.

The results from the case study in [16], which was launched collaboratively with industries in Malaysia, are incorporated in column [16] in Table 5

5. Conclusion

The proposed model discusses merging measurement attributes for software certification with the help of fuzzy model, where it is the easier and simpler to provide accurate results to ambiguous data. The model is capable of providing results more smoothly than those used in conventional systems. The proposed model also provides the continuous improvement of the software product and software process, which provides the continuous improvement of the quality model applied in the certification process. The advantages of the proposed model are:

1- The approach adds value through its comprehensiveness (from requirements to tests), and integrity, its focus on correctness and by establishing a standard to perform software certification that it uniformly establishes what to check and how to check it.
2- Unlike ISO 9000 and the CMM, the proposed model allows innovative processes to emerge.
3- This approach avoids the risk that in ISO and CMM, (To protect data confidentiality and privacy by only permitting users to have direct access to the software, because it includes integrity).
4-This approach eliminates bias assessment and valuation of the product by including an independent assessor in the team.
5- It removes unfairness evaluation by including the owner or users of the product to participate in the assessment process.
6- It accelerates the process because the team is familiar with the product and its environment.
7- This model can be applied in cases which are collaboratively implicated with large organizations.
8- This quality model is able to improve its components or characteristics according to current and future requirements.
9- The model also may be able to handle multiple assessment and certification exercises easily and efficiently.
10- In the proposed model the rules become fewer in number.
11- Facilities the process of product enhancement with the least cost and effort.
12- Focus of error detection for fast recovery for the product.

Reference

[1] L. K. Rierson.: Using the Software Capability Maturity Model for Certification Projects, Proceedings of 17th Information Conference on Digital Avionics Systems Conference (DASC), pp. C241-C248, Washington, USA, 31 Oct.-7 Nov., (1998).

[2] E. Wong. : Managing Software Quality Through Process Improvement Techniques and KPIs (Key Performance Indicators), Proceedings of International Conference on Hong Kong Software Quality Assurance (HKSQA), Hong Kong, pp. 1-11, 18 April, (2012).

[3] A. Deraman, J. H. Yahaya, and F. Baharom.: Continuous Quality Improvement in Software Certification Environment, Proceedings of the International Conference on Electrical Engineering and Informatics, pp.11-17, Bandung, Indonesia, 7-19 June, (2007).

[4] P.M. Heck.: A Maturity Model for Software Product Certification, In Proceeding International Workshop on Software Certification Certsoft'06, pp. 17-28, Canada, 26-27 August, (2006).

[5] R. Bhatnagar, V. Bhattacharjee, and M. K. Ghose.: A Proposed Novel Framework for Early Effort Estimation using Fuzzy Logic Techniques, Global Journal of Computer Science and Technology, Volume10, Issue 14, pp. 66-72, November, (2010).

[6] M. Ortega, M. Perez and T. Rojas.: A Model for Software Product Quality with a Systemic Focus, Proceedings of the 6th International Conference on Information Systems Analysis and Synthesis (ISAS), PP. 395-401, Orlando, Florida, July, (2000).

[7] F. Baharom, A. Derman and A. R. Hamdan.: Introducing Software Process Assessment and Certification (SPACE) Model, Proceeding of the 3rd Malaysian Software Engineering Conference , Kuala Lumpur, pp. 59-63, (2007).

[8] J. H. Yahaya, A. Deraman, and A. R. Hamdan.: SCfM_PROD:A Software Product Certification Model, Proceedings of International Conference of Interaction & Communication Technologies from Theory to Applications(ICTTA'08), pp.1-6, Damascus, Syria, 7-11April, (2008).

[9] J. H. Yahaya1, A. Deraman, and A.R. Hamdan.: A Case Study in Applying Software Certification Model by Product Quality Approach, Proceedings of theInternational Conference on Electrical Engineering and Informatics, pp.706-709, Bandung, Indonesia, 17-19 June, (2007).

[10] K. Seth, A. Sharma & A. Seth.: Component Selection Efforts Estimation– a Fuzzy Logic Based Approach, International Journal of Computer science and Security (IJCSS), Volume 3, Issue 3, pp.210-215, June, (2009).

[11] F.Fishel, J. Ferrell.: User Perceptions of the University of Florida's On-line System for Continuing Educational Opportunities for Certified and Licensed Pesticide Applicators, Journal of Extension, Volume 47, No. 1, pp. 1-4 , February, (2009).

[12] M. Minisini, L. A. Sheppard, and A. Jones.: Self-Efficacy Beliefs and Confidence of Rural Physiotherapists to Undertake Specialist Paediatric Caseloads: A Paediatric Example, The International Electronic Journal of Rural and Remote Health Research, Education Practice and Policy, Volume 60, No. 6, pp. 1-11, (2010).

[13] B. W. N. Lo and X. Gao.: Assessing Software Cost Estimation Models: Criteria for Accuracy, Consistency and Regression, Australasian Journal of Information Systems (AJIS), Volume 5, No. 1, pp. 30-44, (1997).

[14] C. R. Symons.: Function Point Analysis: Difficulties and Improvements, IEEE Transactions on Software Engineering, Volume 14, Issue 1, pp. 2-1, (1988).

[15] K. K. Agawam, Y. Singh, P. Chandra and M. Puri.: Measurement of Software Maintainability Using a Fuzzy Model, Journal of Computer Sciences, Volume1, Issue 4, pp. 538-542, (2005).

[16] A. Deraman and J. H. Yahaya.: The Architecture of an Integrated Support Tool for Software Product Certification Process", Journal of Information & Systems Management, Volume 1, No. 1, pp. 46- 56 , March, (2011).