Bio-Inspired Intelligent System for Software Quality Certification

Saad Mohamed Darwish*
Department of Information Technology, Institute of Graduate Studies and Research, Alexandria University
163 Horreya Avenue, El Shatby 21526, P.O. Box 832, Alexandria, Egypt.

* Corresponding author; email: saad.darwish@alex-igsr.edu.eg
Manuscript submitted October 1, 2015; accepted January 20, 2016.
doi: 10.17706/jsw.11.4.362-375

Abstract: Recently, software quality issues have come to be seen as an important subject as we see an enormous growth of agencies involved in software industries. However, these agencies cannot guarantee the quality of their products; thus leaving users in uncertainties. Software certification is the extension of quality by means that quality needs to be measured prior to certification granting process. However, building accurate certification model is hard due to the lack of data in the domain of software engineering. This research participates in solving the problem of evaluating software quality by proposing a model that uses a fuzzy inference engine to integrate both of the processes–driven and application-driven quality assurance strategies. The key idea of the suggested model is to improve the compactness and the interpretability of the model's fuzzy rules via employing an ant colony optimization algorithm (ACO), which tries to find a good rule description by set of compound rules initially expressed with traditional single rules. The proposed model is an adaptive one that can be seen as taking certification models that have already been built from software quality domain data and adapting them to context-specific data. The model has been tested by a case study and the results have demonstrated feasibility and practicality of the model in a real environment.

Key words: Ant colony optimization, search-based software engineering, software quality metrics, software assessment.

1. Introduction

The past decade has seen the speedy development and diffusion of software that has become very essential in everyday life; thus the quality of the software is a large concern, vital and critical. IEEE defines software quality as the fitness for use of the software product and to conform to the requirements and to provide useful services [1]. Software firms are competing to produce software that is claimed to be good and fulfill user's expectation and requirements. As a general rule, quality is in the eye of the beholder because different people may view quality in their own perspectives. Software quality assurance affects both immediate profitability and long-term retention of customer interest.

In the previous generation of software development, software quality is measured through static assessment of code's structure. Expediently, a new development generation realizes that software quality is more than just static features; it should also comprise non-functional, behavioral and human aspects [2]. People who are involved in software such as users, developers are becoming more concern on the other aspects or views of quality. As a rule, a good software development processes do not guarantee the excellent

quality of product. Thus, assessment of end product must be independent from the development process. Assessing software quality at the early stages of the design and development process is very difficult since most of the software quality characteristics are not directly measurable. Nonetheless, quality can be derived from other measurable attributes. For this purpose, software quality models have been extensively used.

In the literature, software quality can be measured via three categories of assessments [3]: 1) Internal measures: depend on internal attributes that relating to how the software product is developed; 2) External measures: based on external attributes that typically measuring the behavior of the code when executed; and 3) Quality is use measures: include a basic set of characteristics that affect the software like productivity and satisfaction. Even though there are several models of software quality available from literature (see [4] for more details); it is still believed that quality is a complex concept, because quality is nothing more than a prescription some like it excellent, good or bad. Thus, there can be no single, simple assessment of software quality suitable for everyone. In this context, the researchers speak of software quality assessment models which can be used to build relationships between the measurable attributes on the one hand and the software quality characteristic of interest on the other.

Certification of software offers organizations more certainty and confidence about the software. Certification of software helps software sales, acquisition, and can be used to certify legislative compliance or to achieve acceptable deliverables in outsourcing. The growth of software certification methodology is a recent innovation in the field of software assessment, which will improve user confidence toward the quality of the software. ISO defines certification as a procedure by which a third party gives written assurance that a product, process or service conforms to specified characteristics [5]. Software certification can be viewed in three different perspectives: personal, processes and product known as a certification triangle [6]. The mixture of these perceptions will produce the best results. At the recent time, communities are starting to accept the certification concept in the software industry.

Two reasons that trust in software certification must come from someone other than the software publisher. The reasons are [7], [8]: 1) by hiring a third party to grant software certificates, publishers shift the responsible on liability concerning quality onto someone else, and 2) unbiased assessment from independent agency may benefit end users. The approach on certifying software through independent third party organization has a potential to change the manner in which software is evaluated, graded and sold. One way to conduct this approach is through the involvement of end users in the process. In this approach, the independent certification body collects valuable information from the user's environment and collects on how the product is used. The respective agency then produces the report based on significant operational experience created by the users. The second approach is through self-certification by the developer. Each of these methods to assess software has its advantages and drawbacks.

A few certification models have been developed and built in the literature that associated with two distinct approaches [3], [7]-[13]. The primary goal of the first approach is in ensuring that the software development technique is carried out effectively and efficiently to meet the expected quality criteria. It is mainly focused on key factors such as the quality of the process, the quality of the involved people, the use of development technology, the stability of working environment, and finally the project conditions. The second approach focuses with product quality perspective. This approach puts emphasis on grouping quality factors into several working areas. It consists of four main elements: pragmatic quality, assessment team, certification representation, and product certification repository. Each of these specialized parts has separate quality criteria and performs discrete procedures and tasks in the assessment and certification process.

In general, the above mentioned models are intended to be exhaustive and identify both the internal and external quality attributes of the software. But, rarely these attributes have been organized into a sort of

systematic framework. At the same time, there are no guidelines on how to provide an overall assessment of quality rather than focusing on the user view of software. There is an extra issue that relates to the quality; there are priorities between attributes that require assigning weight (by the owner) to reflect the business requirements. In ISO 9126 all attributes are equally important [5]. Reader looking for more information regarding the survey and state-of-the-art software certification models can refer to [14]. The related works strengthen the demand for better quality evaluation and certification that can be formed to ensure that users will be getting software packages that meet expected and contracted quality.

Recently, fuzzy logic–a mathematical tool for dealing with imprecision and information granularity- is used in many software development applications [15]-[17]. For instance, Seth, K. *et al*. [15] proposed a fuzzy rule based model for guessing the efforts in selecting commercial-off-the shelf software components. This model considers five factors that affect the selections efforts as input to the fuzzy system and provides a crisp value of selection efforts using the rule base. The authors in [16] introduced four integrated parameters to measure software maintainability using a fuzzy model. Still, there are some limitations of fuzzy logic-based software development; the most important of which is that the knowledge does not always completely exist and the manual tuning of all the base parameters takes time. Also, the lack of portability of the rule bases when the dimensions of the software scale change make the difficulty still more serious. To cope with these problems, the fuzzy rule learning (FRL) method has been introduced [18].

1.1. Major Line of the Work

Although, different techniques and third party tools have been proposed in the past for software certification procedure, but to the best of my knowledge no certification scheme automatically handles both of the priorities between attributes and the compactness of the rule bases inside a sort of systematic fuzzy rule learning framework. In comparison with state-of-the-art certification models that require assigning weight by hand to reflect the business requirements; the weight values defined in the suggested model are based on bio-inspired computation that formulates software engineering problems as optimization problems and applies meta-heuristics techniques (genetic algorithms and ant colony optimization) to solve them.

The goal of this paper is to present a fuzzy rule based-software assessment model for developing a software certification application. This model specially considers the problem of combining multiple attributes from direct and indirect measurements to obtain global systematic quality with the aim of improving certification accuracy. The combining strategy is based on matching score, where the measurements generated by the individual attributes are averaged using fuzzy sets and the results is then used in certification evaluation. In this contribution, the proposed model utilizes an ant colony optimization algorithm (inspired from the ant' pheromone trial following behavior) as a way of facing the FRL problem [19]. To do so, the FRL problem will be formulated as an optimization problem and the features related to these kinds of algorithms –such as heuristic information, fitness function, and pheromone updates- will be engaged.

The outline of the remainder of this paper is as follows. The second section describes the proposed framework for software certification. The test results and a discussion are shown in Section 3. A short summary of this paper and an outlook of future works are given in Section 4.

2. The Proposed Certification Model

The certification model defined in this research is an intelligent tool to support and manage both of process and product quality perspectives. It also provides a mechanism for monitoring of quality and continuous improvement of software quality throughout its life cycle. For certification two types of input are required: 1) one or more software artifacts. The model does not consider only the final software product

(the source code or working system), but also intermediate deliverables like user requirements and detailed design, and 2) one or more properties of these artifacts that are to be certified. These properties can be of one of the following categories: consistency, functional, behavioral, quality, and compliance. The model has been developed underpinning the hypothesis that software assessment can be estimated with the help of fuzzy model. The implementation of the model is divided into four stages as sketched in Fig. 1. The following subsections describe the details of the model.

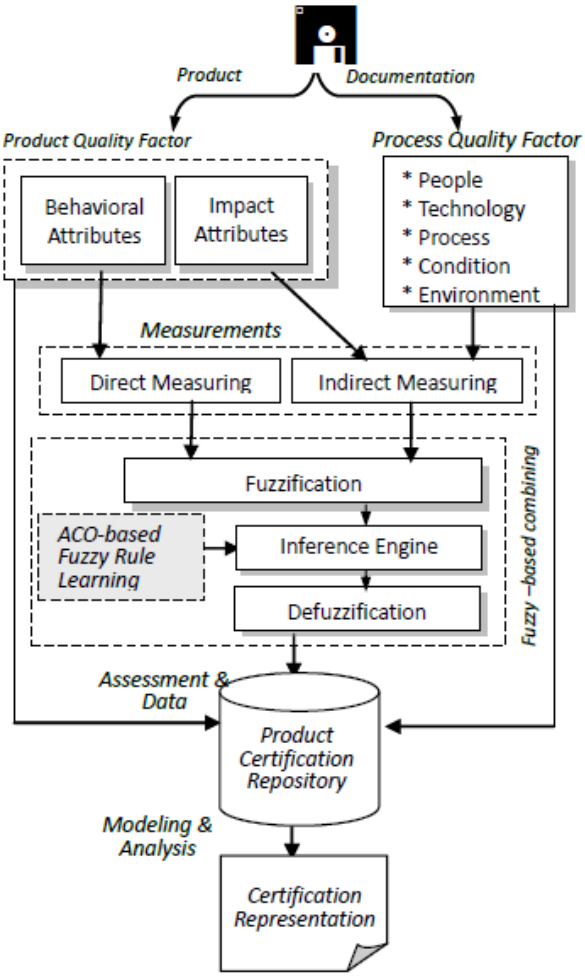

Fig. 1. Proposed software certification model.

2.1. Software Quality Factor

Given the candidate software product to be assessed that is prepared to be delivered to clients, information on the development process is collected by reviewing all manufactured articles produced during development stages. The first component of the proposed model is the software quality factor. It defines what to be measured in this model. Users may select their motivating quality attributes to meet their clerical requirements, which offer flexibility in the certification exercise. The model's quality factor includes process quality factor (*PQ*) and product quality factor that divides into behavioral attribute (*BQ*) and impact attribute (*IQ*).

The process quality factor identifies factors that affect the quality of the software process in practice. These factors are [7-9]: 1) Process (*Pr*): measures the durability of the basic process activities; development, management and supporting; 2) People(*Pe*): measures in terms of team's skill, experience, and knowledge;

3) Environment (*E*): measures the comfortability and safety concerns in the workplace; 4) Technology (*T*): measures in terms of procedure, tools, and techniques, and 5) Condition (*C*): the aspect of quality that measures the time delivery and budget. Formally, the software quality based on the development process factor is a function of the previously introduced factors, plus an error term, ε that represents a quality portion in which the utilized factors can't define:

$$PQ = f(Pr,Pe,E,T,C) + \varepsilon . \quad (1)$$

The behavioral attribute is defined as the external quality feature of the software. It is derived from ISO 9126 model such as functionality, usability, portability, integrity, etc. [5]. In this model, each attribute is made up several metrics that show the measurement aspects of the attribute. Note that, the quality features defined in ISO model are generic attributes that require some customization for a particular case. The quality of software based on behavioral attribute can be formulated as (A_i is the set of *n* product attributes):

$$BQ = f(A_i) + \varepsilon, \ 1 \le i \le n. \quad (2)$$

The impact attribute refers to the human side of quality towards the product and measures the conformity of the software to the user requirements. These attributes are important to balance the quality model between the technical assessment of software and human factor. The impact attributes are decomposed into two distinct sub-attributes: (1) user perceptions (*Up*), where the metrics include assessing of popularity, performance, environmental adaptability, and trustworthiness; (2) user requirements (*Ur*), in which the metrics contain user acceptance and satisfaction. The impact factors defined in this model are based on findings from previous empirical study discussed in [8], [20]. This attribute can be written as:

$$IQ = f(Up,Ur) + \varepsilon , \quad (3)$$

in summary, the total formulation of the quality of a software product is characterized as a linear equation as follow:

$$Q_{total} = PQ + BQ + IQ + \varepsilon . \quad (4)$$

2.2. Measurements Module

Because the software development process activities' documentation is usually written in human readable format, the metrics that have been defined for process quality factor require manual analysis. For that reason, the measurements used for both impact attributes and process quality metrics are Likert scale of 1 to 3 based on joint outlook among assessment team members, including user, developer and independent assessor (must be an expert in software engineering and software quality). Likert scale is defined as something that we measure the satisfaction based on perception (questionnaire) [7]. This technique presents a set of attitude statements and subjects are asked to express agreement or disagreement of a three-point scale. The scale used in this certification model is recommended to 1=low; 2=medium; 3=high.

On the subject of the behavioral attribute, the available model focuses on the correctness, complexity, efficiency, reliability, completeness and modularity for the performance of management information system, since they have the biggest impact on whether or not the system would be successful. These characteristics appear in all models and therefore, are considered as essential and vital. In general, selection criteria for product quality requirements include tradeoff between the quality characteristics and their priorities.

Another selection criterion is the ability to collect the required data for the metric. Note that, in assessing the existing quality product, it is useful to examine the history of software components in actual use. The following equations (see Table 1) are used to compute the above mentioned software product quality metrics [2], [3], [7], [11]. To satisfy the requirement of the membership function used in the fuzzy logic-based software assessment, the values of each attribute measurement (direct and indirect) need to be transformed into range [0,1] with Gaussian normalization method (see [16] for details). The beauty of this model is that software owner or the stakeholder has flexibility and authority to choose relevant weight values to reflect the the business requirements and constraints. It is normal in a business environment that in some situation certain quality attributes are more important than the others.

Table 1. Behavioral Attribute Metrics

Metric	Equation	Description
Correctness	$DD = \frac{\sum_{i=1}^{L} D_i}{KSLOC}$	Computed via Defect Density (*DD*),where D_i is the total number of defects during the *i*-th code inspection, *L* equals the total number of inspections to date, and *KSLOC* indicates the number of source lines of executable code (in thousands).
Complexity	$C_{program} = \sum_{i=1}^{m} DE_i + m$	Where DE_i is the decision count (count of branching, condition and loop control statements) for the *i*-th module and *m* is the total number of software's modules.
Defect Removal Efficiency(*DRE*)	$DRE = Er / (Er + Dr)$	Employed to measure efficiency, where *Er*= number of errors found before software delivery, and the *Dr*= number of errors found after software delivery.
Reliability (*RE*)	$RE = \left(\frac{MTBF}{MTTR}\right)\%$	The degree to which a software system behaves robustly over time. *MTBR* represents the mean time between failure, and *MTTR* stands for mean time to repair.
Completeness	$CE = 100 \times \left(1 - \frac{A_{existing}}{A_{tot}}\right)$	The degree to which the software components implement all required capabilities and is determined by the use of calculation of cause/effect completeness (*CE*). $A_{existing}$ = number of remaining requirements, and A_{tot} = total number of identified requirements.
Modularity	$Coupling = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} item_{ij}}{m \times n}$	The way the software component is decomposed into sub-component. Data coupling is the sharing of data via parameter lists, $item_{ij}$ is the sum of number of input and output items shared between components *i* & *j* and *n* symbols the number of components in the software product.

2.3. Ant Colony 0ptimizer for Fuzzy Rules Design

This module describes the main part of the proposed model, which integrates both of process and product quality attributes via ACO-based fuzzy domain. Inside the fuzzy logic approach, the first step is to transform the discrete values into continuous one, this process is called fuzzification. These are then manipulated in the fuzzy domain by an inference engine based on knowledge base (rule base and database) supplied by domain experts. Finally the process of translating back fuzzy members into single values is named defuzzification [15]-[17]. This unit considers all quality attributes' measurements obtained from the previous step as inputs and provides a crisp value of certification level as output using the ACO-based rule base. All input values can be classified into fuzzy sets as low (L), medium (M) and high (H). In order to fuzzify the inputs, the trapezoidal function is chosen. In addition the output variable (quality level) is classified into fuzzy sets as poor, basic, good and excellent using the triangular membership function.

It has been observed that the fuzzy systems produce a large rule set, thus affecting the interpretability of the system. Also the generated rule base is complex, thus making the overall inference system complex. One way to improve the interpretability of a fuzzy model consists of trying to identify rules as general as

possible; so that each rule covers the highest number of examples and, this way, the size of the rule base diminishes. However, the goal of finding the optimal set of such general rules is not an easy task. It is possible to extend the syntax of the rules by associating more than one label to each input variable in the antecedent of the rule and by using other relational operators different from the usual equal–to operator. These rules called compound rules *R* denoted as [21]:

$$R^i : X_1 \, op_i \, LX_1^i, \ldots . X_n \, op_n \, LX_n^i \rightarrow LY^i , \tag{5}$$

here, $op_i \in \{=, \leq, \geq \div\}$ ($\div$ means between); each LX^i represents both the membership function and the linguistic label of the i^{th} value, LX^i is one label if $op_i \in \{\leq, \geq\}$, and two labels if $op_i \in \{=, \div\}$. Analogously LY^i is the output fuzzy domain.

In this section, an ACO algorithm is utilized that tries to search for the best transformation of a fuzzy model described initially as a set of single rules in a set of compound rules. ACO is a meta-heuristic algorithm inspired by the behavior of real ants, and in particular how they forage for food with shortest route between their nest and a source of food. This is done using a chemical substance called pheromone trails, which ants deposit whenever they travel, as a form of indirect communication. The ACO is working as follows [22]:

Algorithm 1: ACO Procedure

1. Initialize pheromone trails;
2. **Repeat**
3. Each ant is positioned on a starting node
4. **Repeat**
5. Each ant applies a *state transition rule* to incrementally build a solution and a local *pheromone-updating* rule;
6. **Until** all ants have built a complete solution Global *pheromone-updating rule* is applied
7. **Until** terminating condition is reached

To apply ACO algorithms to the problem of finding the optimal set of general fuzzy rules, the following steps have to be performed [23], [24]:

1) *Problem Representation*: To apply ACO algorithms to the *FRL* problem, it is convenient to see it as a combinatorial optimization problem with the capability of being represented on a graph. Herein, there exist two types of edges that the ants will travel:
   (a) <*i*,0,0> allowing the ant to include the *i*-th initial rule in the set of compound rules to form the *i*-th compound rule R_i. The work presented in this paper uses the method introduced in [19] to construct the set of initial rules. The basic idea is to divide the input space into a fuzzy grid and take the rules with a maximum certainty degree in every input fuzzy region of this fuzzy grid as follows :

$$\begin{aligned} &\exists e_i = \left(x_1^i, \ldots . x_n^i, y^i\right) \in E \quad , LX_{1,i}, \ldots . . LX_{1,n} \neq 0 , \\ &R_{LY_k}^i : LX_{1,i}, \ldots \ldots . . LX_{n,i} \rightarrow LY_k , \\ &k = \arg \max_{j=1 \ldots t} w\left(R_{LY_j}^i\right), \end{aligned} \tag{6}$$

where *w* is a certainty measure based on $E = \{e^1, \ldots . e^t\}$ that represents a set of *t* examples. If no example

covers a region no rule will be selected; if several rules take the maximum certainty degree, one of them will be selected randomly.

(b) <*i*,*j*,*k*> allowing the ant to add to the *i*-th compound rule , R_i, of the *k*-th label ,$LK_{j,k}$,from the fuzzy domain of the *j*-th input variable (i.e. amplifying a compound rule by adding one label to one of its premises).

In this case, each ant will be randomly located in an initial rule (no more than one ant for each initial rule), so that this initial rule will be included in the set of compound rules of the ant. Subsequently, the ant will select one step among the feasible transitions at each ant state.

2) *Define the way of assigning a heuristic preference*: the heuristic function provides a way to guide the search to the paths containing a *good* accuracy issue (e.g. the number of positive initial rules the step covers) and a good interpretability subject (e.g. the width of the covering gained with the step). The heuristic function is defined as follows:

$$\eta = \gamma \frac{R_{cov}}{R_{max}} + (1-\gamma)\frac{R_{pos}}{R_{max}},$$

$$R_{max} = \begin{cases} \prod_{l \neq j} p_l & \text{if the step is } \langle i,j,k \rangle \\ \prod_l p_l \Big/ \max_l \times p_l & \text{if the step is } \langle i,0,0 \rangle \end{cases}, \quad (7)$$

R_{cov} is the number of the input regions of the fuzzy grid covered by the amplification zone and R_{pos} is the number of the input regions with only positives initial rules, R_{max} is the maximum number of input regions covered by the amplification zone, *P* is the rule's premises. Finally, the parameter γ controls the influence of both accuracy and interpretability in the final value. In this case, each premise in the antecedents of the rules will be described by the ant with a set of labels associated with the corresponding input value.

3) *Establish an appropriate way of initializing the pheromone*: the pheromone trails refer to information shared by all the ants in the algorithm that stores the past experience collected by all of them about the good /bad *selection* in the steps. In this case, the initial amount of pheromone deposited in each arc of the graph is computed as:

$$\tau_0 = \frac{1}{2\ N_v * N_r}, \quad (8)$$

N_r is the number of initial rules and N_v denotes the number of input variable. In this paper, a key issue in the pheromone deposit strategy is the definition of the interpretability level of a compound level. Therefore, the pheromone updating in each step *s* will be:

$$\tau_s = (1-\rho)\tau_s + \sum_{k=1}^{M} \Delta\tau_s^k, \quad (9)$$

where $\rho \in (0,1)$ is the pheromone evaporation rate that affects to all the arcs in the graph by which less accurate trails are gradually removed, *M* is the number of ants and $\Delta\tau_s^k$ is the amount of deposited pheromone on the arc associated with a step <*i*,0,0> involved in the construction of the final compound rule, R_i, by the *k*-th ant that is calculated as:

$$\Delta\tau_s^k = \frac{R_{cov} * C(R^i)}{N_{cr}},$$

$$C(R^i) = \frac{\sum_j p_j - c(p_j)}{\sum_j p_j}, \tag{10}$$

N_{cr} is the number of compound rules in the final solution, $C(R^i)$ refers to the compactness of the rule, and $c(p_j)$ is the complexity cost of each premise that defined as:

$$c(p_j) = \begin{cases} 0 & if\ void\ premise \\ 2 & if\ (=\{LX_{j,k}\}) \\ 2 & if\ (<\{LX_{j,k}\}) \\ 2 & if\ (>\{LX_{j,k}\}) \\ 3 & if\ (\div\{LX_{j,k_1}, LX_{j,k_2}\}) \\ l+1 & if\ (=\{LX_{j,k_1}, \ldots\ldots, LX_{j,k_2}\}) \end{cases}. \tag{11}$$

4) *Define the fitness function*: the fitness function establishes the quality of a solution. The measure will be the mean *square* error:

$$MSE(R_k) = \frac{1}{2|E|}\sum_{e_l \in E}\left(y^l - F_k\left(x^l\right)\right)^2, \tag{12}$$

with $F_k\left(x^l\right)$ being the output obtained from the rule generator by the ant K, when receives the input x^l of the example e_l, and y^l being the known desired output. The closer to zero the measure is, the better the solution is.

Once the previous components have been defined, an ACO algorithm has to be given to solve the problem. In this respect, the original ant system version proposed in [23] has been implemented, which applies a random proportional rule for selecting each step and whose pheromone deposit mechanism is run once the solution is completed. An example of the solution obtained by the proposed ACO algorithm for a training set with 20 examples is the following rule base, where the parameters were set as M=5, γ =0.25, ρ =0.1, the number of cycles was bound to 20, and the corresponding run finishes if all the ants in a cycle construct the same set of compound rules.

$$R^1 : \text{if } x_1 \le \{\text{M}\}, \text{and } x_2 = \{\text{M}\} \rightarrow Basic$$

$$R^2 : \text{if } x_1 \le \{\text{M}\}, \text{and } x_2 = \{\text{H}\} \rightarrow Good$$

$$R^3 : \text{if } x_1 = \{\text{L}\}, \text{and } x_2 = \{\text{L}\} \rightarrow Poor$$

$$R^4 : \text{if } x_1 = \{\text{M}\}, \text{and } x_2 = \{\text{L}\} \rightarrow Basic$$

$$R^5 : \text{if } x_1 = \{\text{H}\}, \text{and } x_2 = \{\text{H}\} \rightarrow Exellent$$

$$R^6 : \text{if } x_1 = \{\text{H}\}, \text{and } x_2 \div \{\text{L}, \text{M}\} \rightarrow Good$$

In general, it can be observed that the fuzzy model is described, in average, with a low number of rules when compared with the initial fuzzy model and these rules have a low complexity. All the above rules are entered and the rule base is created; a rule will be fired depending on particular sets of inputs. In this case,

using the rule viewer, the output is observed for a particular set of inputs using the Matlab fuzzy logic toolbox. The configuration of the utilized fuzzy system is shown in Table 2.

The interpretability of fuzzy models is one of their key features. Due to it, this feature must be preserved and enhanced instead of being relegated to a secondary place. With this aim, an ACO algorithm has been proposed that finds good descriptions by means of compound rules of fuzzy models initially expressed with conventional single rules. The construction graph allows representing each solution as a sequential addition of labels to the premises in the antecedent, and from the cooperative behavior of the ants good combinations of compound rules emerge.

Table 2. Inputs and Outputs for Fuzzification

Type	Description
System	Name="Assessment", Type="Mamdani", Number of Input =2 , Number of Output=1 , Number of Rule=6, Defuzzification Method="Centroid"
Input 1	Name="Avg(direct measuring attributes)", Range=[0 1], NumMFs=3=3, MF1="Low" : "TrapMF"[0, 0, 0.3, 0.4] MF2="Medium": "TrapMF" [0.3, 0.4,0 6,0.7] MF3="High" : "TrapMF" [0.6, 0.7, 1, 1]
Input 2	Name="Avg(Indirect measuring attributes)", Range =[0 1] , NumMFs=3, MF1="Low" : "TrapMF"[0, 0, 0.3,0.4] MF2="Medium": "TrapMF" [0.3, 0.4, 0.6,0.7] MF3="High" : "TrapMF" [0.6, 0.7, 1, 1]
Output	Name="Quality", Range =[0 1] , NumMFs=4, MF1="Poor" : "TriMF"[0, 0.25,0.5] MF2="Basic" : "TriMF" [0.25, 0.5,0.75] MF3="Good" : "TriMF" [0.5, 0.57, 0.9] MF3="Excellent": "TriMF" [0.75, 0.9, 1]

2.4. Certification Presentation

This module presents certification-mapping procedure to obtain the associated certification level of the software product. The certification level is used to picture the overall performance of software process development practices. The certification level is determined by the crisp value c returned from defuzzification operation. The proposed model adopts the certification approach in [8] for identifying and naming of the certification level. This level is shown in Table 3. It is important to note that the ranking of the utilized certification level is flexible and does not fix to the mentioned states.

Table 3. Ranking of Certification Levels

Crisp Value	Certification Level	Certification Status	Description
$0.9 \le c \le 1$	4	Excellent	The software satisfies all quality criteria
$0.75 \le c < 0.9$	3	Good	Satisfactory quality
$0.5 \le c < 0.75$	2	Basic	Acceptable quality
$0 \le c < 0.5$	1	Poor	Unsatisfactory quality

2.5. Certification's Data Warehouse

The final component is the repository, which stores all information and results from assessment and certification exercises. This data is helpful for upcoming analysis and enhancement. Herein, the procedure

of evaluation is applied in a combination of the following approaches: 1) First party evaluation is related to internal product and process assessment; 2) Second party evaluation, is associated with acceptance evaluation on product before delivery; and 3) Third party evaluation, is the independent evaluation by an independent body or a testing laboratory [25].

3. Evaluation of the Model

This section verifies the proposed certification model feasibility using real case study. The case study was carried out with a large governmental organization (Alexandria University-Egypt) and one of the systems in the university was selected to be assessed and certified. The selected system is a Management Information System (MIS) that is considered steady and being used broadly. The assessed system is a large system and operates in each Faculty in the University. During the assessment and certification exercise, the data were collected through multiple data gathering techniques, which include document reviewing, interviewing and observing. The data collection was conducted through a mutual outlook judgment among members in the team, which consists of the independent assessor, developer and users.

Based on the assumption that the assessment and certification can be repeated several times during the life span of the products; two assessments were carried out in two different periods to check the stability of the selected attributes to assess the software. The interim period between the assessments is about 6 months to evaluate the behavioral attribute metrics under diverse conditions. Following the data collection, data analysis was carried out and the results are shown in Table 4. In this case, the result of quality for the direct' measuring attributes is in range between 75-80 %, while the result of the indirect' measuring attributes is 70%. The total quality score of this product is computed by averaging the attribute scores through the fuzzy inference engine. The total score of this product is 78.6 %. In this case, this score is mapped into the certification level to obtain the relevant certification status of this product. This shows that the software was developed and implemented with enough involvement of users and management, therefore the quality aspects is considered satisfactory and sufficient.

Table 4. Summary of Case Study Certification

Criteria	
Sector	Business Management
Software	Information System
Development Approach	Out-Source
Duration of Use	Two year
Results of Assessment and Certification	
Quality Score	78.6 /100
Certification Level	3
Certification Status	Good

In general, the application of the case study has demonstrated the feasibility and practicality of the proposed software's assessment model. The model provides beneficial information to the developers, owners as well as the stakeholders on the quality status of the system. Furthermore, this model has been developed to work with current requirements for certification and quality issues. The experience of this research reveals the potentials in supporting ACO based fuzzy logic technique as an alternative way to merge different attributes instead of attributes' weighting approaches [8],[12],[25]. In which, the weight values are assigned by the owner of the product based on its organizational requirements and expectations. A significant advantage of the suggested model is that it allows users to choose and design their implementation model of certification which fit with their organization's requirements and expectations.

The chart in Fig. 2 demonstrates the results in both assessments and illustrates the quality attribute's

scores in more detail. The first assessment study is labeled in straight line while the second assessment study is labeled in a dotted line. In Kiviat chart the points fall in outer layer are considered better than the points in the inner layers. Each attribute is represented by axis and scores are plotted at the limits between 0-100%. Kiviat graph can be used to easily identify attributes that need attention in this process. Attribute that fall on the limit's outer layer is considered better quality compares to attributes at inner layers of this graph.

It is clearly shown in the chart that most attributes of the first assessment: correctness, modularity, completeness, usability, and complexity fall in the inner layer compared to the second assessment. This shows that the quality performance of the attributes is enhanced from the first to the second assessment. Whilst the other attributes: efficiency, reliability and integrity have degraded. This degradation may come from several problems such as system hangs and downs for some period of time. The recovery time normally long and it ranges from several minutes to one or two days. This problem causes the low score in efficiency and reliability. Furthermore, integration with other systems operating that causes difficulty to the users to switch from one system to another. The developed certification model does not only offering a mechanism for the certification process, but also providing an alternative mechanism for monitoring of quality and continuous improvement of software quality throughout its life span.

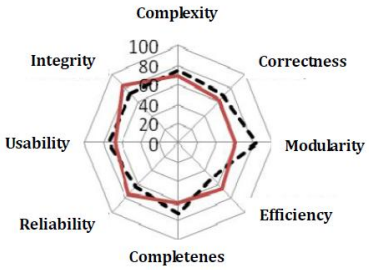

Fig. 2. Kiviat chart of the system quality attributes.

There are at least three factors to be considered that influenced the certification level of a candidate software product. The three factors are the operation period in the environment of the candidate, second the weight factors of attributes of the candidate product, and third, the number of rules derived from ACO algorithm. The studies show that the longer the operating period of the software the better result of quality and certification level can be achieved. Clearly, this is true because the software has been updated and corrected accordingly and necessarily by the developers. This relates to the issue of maturity of the software. On the other hand, if the certification exercise is conducted periodically over some time intervals, an unexpected result may be seen because of the aging of the software. The second aspect that influences the result is the weight factors of the product. Without assigning weights factors to quality attributes, the results may indicate different level of certification. Thus, this model accommodates fuzzy logic–based weight factors for all attributes with different level of importance to reflect individual business requirements.

The third aspect that impacts the result is the number of compound rules for fuzzy rule base that can be extracted without leading to inconsistencies or malformations in the final fuzzy model. An inconsistency is produced when a compound rule covers negatively an initial rule (i.e., the compound rule has a different consequent than some of the initial rules it covers). A malformation will be provoked when two or more compound rules with the same consequent overlap in the input space, since they are providing —partially or totally— a redundant information. In general, the support of each compound rule can be defined as measuring the coverage of training patterns.

4. Conclusion

This work towards improving effort in the software certification process, particularly in outlining a system to assess software and determine the software quality status. The proposed model is derived from state-of-the-art certification systems with enhanced features and capabilities that considers both qualitative and user requirement aspects. The persuasive feature of this model is obtained through the hybridizing fuzzy inference tool with ant colony optimization, meta-heuristic engine to fuse different certification levels of software; process and product level.

The advantages of this model compare to other approaches are: (1) remove and eliminate bias assessment and unfairness evaluation by including fuzzy logic tool; (2) an ant colony optimization based local searcher is employed to find the optimal set of compound fuzzy rules to improve the interpretability of a final fuzzy model; (3) the utilized quality factors are derived from the up-to-date ISO model with additional characteristics to accommodate other aspects of software quality requirements;(4) it improves the involvement of participants of various people in the certification process; (5) it enables easy assessment and certification exercises and offers better guidance and procedures; and finally (6) the model able to handle multiple assessment and certification exercises easily and efficiently.

The application on case study has illustrated the practicality and feasibility of the model. Besides, the empirical results prove that the fuzzy-based integrated measures of quality shows a strong co-relation to the assessment accuracy with an aim of supporting certification process. The future model of certification has an intelligence capability and is able to improve itself in the environment. The next strategy is to enhance the model and practice in e-government applications.

References

[1] Yahaya, J., Deraman, A., & Hamdan, R. (2008). Software quality from behavioral and human perspectives. *International Journal of Computer Science and Network Security*, *8*(*8*), 53-62.

[2] Gennaro, G., Lagelle, D., & Schäbe, H. (2001). Software product evaluation and certification. *Proceedings of the International Conference of Data Systems in Aerospace* (pp. 6-26). Netherlands.

[3] Heck, P., Klabbers, M., & Eekelen, M. (2010). A software product certification model. *Software Quality Journal*, *18*(*1*), 37-55.

[4] Kan, S. (2003). *Metrics and Models for Software Quality Engineering.* Addison-Wesley.

[5] Azevedo, R., & Belchior, A. (2005). ISO 9001:2000 certification model. *Clei Electronic Journal*, *8*(*1*), 7-27.

[6] Voas, J. (1998). The software quality certification triangle. *Journal of Defense Software Engineering*, *5*(*1*), 12-14.

[7] Yahaya, J., Deraman, A., & Baharom, F. (2009). Software certification from process and product perspectives. *International Journal of Computer Science and Network Security*, *9*(*3*), 222-231.

[8] Yahaya, J., Deraman, A., & Hamdan, A. (2008). SCFM_PROD: A software product certification model. *Proceedings of International Conference on Interaction & Communication Technologies from Theory to Applications* (pp. 1-6). Syria.

[9] Heck, P. (2006). A maturity model for software product certification. *Proceedings of International Workshop on Software Certification* (pp. 17-28). Canada.

[10] Taylor, C., Foss, J., & Rinker, B. (2002). Merging safety and assurance: the process of dual certification for software. *Proceedings of the International Conference on Software Technology* (pp. 790-811). USA.

[11] Yahaya, J., Deraman, A., & Hamdan, A. (2007). A case study in applying software certification model by product quality approach. *Proceedings of International Conference on Electrical Engineering and Informatics* (pp. 706-709). Indonesia.

[12] Heck, P., Klabbers, M., & Eekelen, M. (2010). A software product certification model. *Software Quality*

Journal, 18(*1*), 37–55.

[13] Yahaya, J., Deraman, A., & Hamdan, A. (2007). Software product certification model: Classification of quality attributes. *Proceedings of the First Regional Conference of Computational Science and Technology* (pp. 436-440). Malaysia.

[14] Denney, E., & Fischer, B. (2005). Software certificate and software certificate management systems. *Proceedings of the International Conference on Software Certificate Management* (pp. 1-6). USA.

[15] Seth, K., Sharma, A., & Seth, A. (2009). Component selection efforts estimation — A fuzzy logic based. *International Journal of Computer Science and Security*, *3*(*3*), 210-215.

[16] Aggarwal, K., Singh, Y., & Puri, M. (2005). Measurement of software maintainability using a fuzzy model. *Journal of Computer Sciences*, *1*(*4*), 538-542.

[17] Kalpana, A., & Jeyakumar, A. (2010). Fuzzy logic based software process improvisation framework for Indian small scale software organization. *International Journal of Computer Science and Engineering*, *2*(*3*), 852-859.

[18] Atoufiand, B., & Hosseini, H. (2010). Bio-inspired algorithm for fuzzy rule-based system. *Journal of Advanced Knowledge: Model, Application and Research*, *1*(*1*), 126-159.

[19] Cassillas, J., Cordon, O., & Herrera, F. (2000). Learning fuzzy rules using ant colony optimization algorithms. *Proceedings of the Second International Workshop on Ant Algorithms* (pp. 13-21). Belgium.

[20] Heck, P. (2009). A Software product certification model for dependable systems. *Software Quality Journal*, *18*(*1*), 37-55.

[21] Carmona P., & Castro, J. (2005). Interpretability enhancement of fuzzy modeling using ant colony optimization. *Proceedings of the first International Workshop on Genetic Fuzzy System* (pp. 148-153). Spain.

[22] Azar, D., & Vybihal, J. (2011). An ant colony optimization algorithm to improve software quality prediction models: Case of class stability. *Information and Software Technology*, *53*(*4*), 388-393.

[23] Thangavel, K., & Jaganathan, P. (2007). Rule mining algorithm with a new ant colony optimization algorithm. *Proceedings of the international Conference on Computational Intelligence and Multimedia Applications* (pp. 135-140). India.

[24] Nobahari, H., & Pourtakdoust, S. (2005) Optimization of fuzzy rule bases using continuous ant colony system. *Proceedings of the First International Conference on Modeling, Simulation and Applied Optimization* (pp.1-6). UAE.

[25] Heck, P. (2009). A Software product certification model for dependable systems. *Software Quality Journal*, *18*(*1*), 37-55.

Saad M. Darwish received the B.Sc. degree in statistics and computer science from the Faculty of Science, Alexandria University, Egypt in 1995. He held the M.Sc. degree in information technology from the Institute of Graduate Studies and Research (IGSR), Department of Information Technology, University of Alexandria in 2002. He received his Ph.D. degree from the Alexandria University for a thesis in image mining and image description technologies. He is the author or coauthor of 50 papers publications in prestigious journals and top international conferences. He has served as a reviewer for several international journals and conferences. His research and professional interests include image processing, optimization techniques, security technologies, database management, and machine learning. Since Feb. 2012, he has been an associate professor in the Department of Information Technology, IGSR.