Journal of Advances in Computer Research
Quarterly ISSN: 2008-6148
Sari Branch, Islamic Azad University, Sari, I.R.Iran
(Vol. 11, No. 4, November 2020), Pages: 73-93
www.jacr.iausari.ac.ir

A Framework for the Empirical Forecasting of the Software Quality

Rezaali Mosayyebi✉[1], Alieh Peykar[2]
1) Department of Computer Engineering, Sari Branch, Islamic Azad University, Sari, Iran
2) Master's Degree in Computer Software Engineering, High School Teacher in Bandar-e-Turkmen, Golestan Province, Iran
rmosayyebi@tci.ir; peykar@iausari.ac.ir

Received: 2020/09/30; Accepted: 2020/12/13

Abstract

Before the beginning of the software development project that is based on contract between software user and developer company, the user company provides the RFP including project domain current situations and constraints, and all conditions such as desired initial system requirements specification as well as requesting documents of developer companies capabilities to ask the developer companies for their proposals to fulfill the requirements and competition with each other. This paper renders an evaluation model of proposals including non-functional requirements (NFRs). It is very important, mainly focusing on the NFRs since the system architecture and quality greatly depend on the NFRs. Within this framework, the NFRs existed in the RFP are weighed and are traced in the proposals cause evaluating respected quality attributes coverage and satisfaction level. The model helps the user company and its consultants compare the proposals and choose qualitative desired proposal. As a case study, 6 telecommunication projects RFPs were evaluated by the proposed model. As a result, we have confirmed that the model can forecast the software quality in practical environments and indicate the effect of each quality attribute coverage and satisfaction level for agreement between software user and developer company to trading price and quality off.

Keywords: *Request for Proposal (RFP), Non-Functional Requirement (NFR), Requirement Evaluation, Quality Attributes Evaluation and Evaluation Model.*

1. Introduction

Software evaluation, as well as its quality, has been presented with different methods so far. Some of these methods evaluate running or implemented software, while others do their job during design, analysis, or requirements identification by developers. The evaluation of non-functional requirements (NFR), and the quality attributes of the software that corresponds to it, has been done with different models so far, and we have used them for request for proposals (RFP), with its special framework and model, and used it for case studies on software contracts for telecommunication projects. Determining the coverage and depth of quality attributes for software quality and price estimation is important before its development. It would be the basis for bargaining and agreement between user and developer of software with a specific price and quality, and also an appropriate ranking of bidder's proposals in an RFP based software competition.

The necessity of this issue is obvious in all organizations and companies that develop their required software through RFP, in inquiries and tenders; Due to the novelty of the software industry in comparison with many other industries, and also the rationality of this product in comparison with other physical products, as well as its specific breadth and complexity, the proposed tool for measuring it is very rudimentary and it is safe to say that there is no consensus on any of these tools. Therefore, the importance of this research is that with this volume of purchase and sale in software industry, there is no definitive criterion for weighing and measuring it. The framework and model presented in this research is a practical and experienced solution for organizations and companies. The foundation of this research can be used as an adaptive method for initial pricing of each software. The advantage of this method is that it can be used at the beginning of the project and can be done with the initial identification of the applicant's requirements. In addition, it does not need to decompose the system for FP[1]or SNAP[2]-based computing and has no weaknesses. We should use metrics to evaluate a software product; Product metrics are dynamic and static; Dynamic metrics are directly related to software quality attributes but static metrics have an indirect relationship with quality attributes. In this research, we use both metrics and compare them. We show that in practical environments, by using both of these metrics, a suitable index can be given to project managers and evaluators. In addition, we use the corresponding dynamic metrics to validate static metrics.

2. Literature Review

Software evaluation is a fundamental process in quality management. The quality management process and software development are closely related. Quality management activities including quality assurance, quality planning, and quality control need to be assessed within themselves. For example, to control software quality, we have two methods including quality review and automatic software estimation, that both end with evaluation. On the other hand, it is necessary to consider improvement prior to quality management. Quality management has various aspects that need to be addressed. Whether we consider quality the user satisfaction, or the compliance of a software product with a requirement [3], validation will not be complete unless we measure this satisfaction and compliance, because satisfaction, conformity, and similar concepts cannot be described as binary and must be considered fuzzy.

Requirements are defined by the International Institute of Business Analysis (IIBA) as a condition or capability required by stakeholder/stakeholders to solve a problem or achieve a goal. While the need is a high level abstraction of the needed requirements. In other words, need is the final result or goal (why are we doing this?). Business needs identify and define why a change to an organizational system or capabilities is required [4]. Understanding the purpose of the project is very crucial to be able to elicit and analyze requirements. When we know why this system is being built, what the business need is for it, and what business impact it has, we will easily be able to ask the right questions when eliciting requirements and prioritizing them. This will in turn help us reject requirements that don't satisfy the need. In some articles, inaccurate understanding of the

[1] Function Point.

[2] Software Non-functional Assessment Process

above issues has led to incorrect results. For example, increasing sales and customer acquisition have been considered as non-functional requirements [5], as a result, there is a misconception about, which functional requirement this non-functional imaginary requirements are related. While both are a need at the business level, not a requirement! To meet each of these two needs, a number of requirements are needed, some functional and some non-functional. In order to improve each of sub-processes, new requirements are raised recursively, and by recognizing each of them, it may be necessary to comply with various requirements.

The software applicants or consultants should be able to estimate the software project, especially in terms of price estimation. Unfortunately, the standard of the project management plan or other standards in [1] does not provide specific guidance in this regard, but in [2] it is viewed in a completely abstract way. Whereas the industry desperately wants a practical way to determine the quality of software and the appropriate price estimation. In the case of software that has not been developed yet, estimation should be made that usually uses FP and similar concepts such as OP[3], UCP[4], etc. The FP guide was provided by IFPUG[5]as the IFPUG Counting Practices Manual in [6] and is standardized in ISO as ISO / IEC 20926: 2009 [7] [8].

A function point or FP is like a chunk of software [6] which expresses the user's point of view towards the software, not the programmer's point of view. In [9], some reasons for using FP in pricing were mentioned; furthermore the benefits of FP in new projects and improvements are also expressed. In [9], various types of FP models including explanatory, predictive, and prescriptive are discussed, and non-functional requirements items are referred to as challenges until a suitable solution is found by software measurement institutes.

There are basically two methods of price estimation in general: process-oriented and consequential-oriented; In the first, the price is set on the basis of inputs or the same sources that can be provided by the supplier; But in the latter, the price is determined based on the output, for example, the size of the software exchanged, which is measured in FP or LOC.

The IFPUG method has no solution for measuring software improvement projects, for this reason, IFPUG SNAP [6] is mostly used; Many also use the NESMA [7] software improvement project measurement method, which includes weights for adding, deleting, and modifying FPs [10]. These methods are used in various ways in Brazil [11], Australia [12], Italy, South Korea, Finland, and many other countries.

Since 1976, when Boehm first introduced a well-defined and clear framework for evaluating software quality issues [13] until 1990, when the NFR's framework was discussed [14] and especially in the last decade, extensive studies on NFRs are underway; non-functional requirements exist wherever there is a need, such as business level, applications, infrastructure, etc. In some references, such as [15] and [16], the approach to non-functional requirements is confined to the scope of the infrastructure, including servers, storages, and network equipment, and software such as operating systems, firmware, and other operations management and control software.

[3] Object Point.
[4] Use Case Point.
[5] International Function Point Users Group.
[6] Software Non-functional Assessment Process.
[7] Netherlands Software Metrics Users Association.

In [15] six major groups of non-functional requirements were presented, in the same reference, comparison with ISO 9126 illustrated in table 5-2-1-1. The two also have differences and overlaps, which naturally arise from the point of view. It should be noted that in [15] more attention is paid to the field of infrastructure. In [16] the concept of granulation is expressed. Each quality attribute is associated with several features that can be obtained through software engineering [2], in other words, by using software engineering methods to create each feature, the software product is affected by quality attributes related to that feature. In developing information systems, a common understanding of the developer and the applicant about non-functional requirements is vital. The reason for the failure of many systems is the lack of attention to this important issue; To the extent that one of the 10 major risks in requirements engineering is the lack of attention to non-functional requirements in projects [17]. Mostly its knowledge is available to designers, but since there is no formal structure for its implementation, it is unfortunately rarely considered in software design [18]. Determining the desired level of non-functional requirements for applications is created by step - by - step filtering, agreement and trade off. In [19] a simple conceptual model is demonstrated to determine the level of non-functional requirements in applications.

Initial requirements actually form the main basis of the system. References [1], [2], and [20] focus on the necessity of mentioning requirements in the RFP. After incorporating the requirements of the problem into the RFP, it is necessary to evaluate the quality of the problem proposing, as well as the proposed solutions. This is why both evaluating the quality of the RFP and evaluating the technical proposals that meet the requirements of the RFP are so important. There have been many studies on the evaluation of RFPs in terms of covering the needs of users, which can be used with a slight difference for the received proposals. Of course, there have been no studies on the evaluation of proposals, because the RFP document and the proposal document are assumed to be equivalent with a bit of tolerance. Hence, the prevailing assumption is that what is requested in the RFP should normally be accepted as the same in the proposal. But if we pay attention to the behavior of ourselves and companies in the software product development contract procedure, we see that in practice, acceptance /rejection of the proposal is not like a Boolean proposition, but compromises on parameters such as different capabilities and their quality according to price. Non-functional requirements are particularly important in both the RFP and the proposal. Because the system architecture and the quality of the software product, which is manifested in cases such as response time, security, etc., depend on them.

FSM [8] or FS is not suitable for non-functional requirements items, it should be experienced from different types of projects, or used COCCOMOII that is the successor of COCOMO [9] and is claimed to be better suited for estimating modern software development projects. In some countries, such as Brazil, some suppliers use FP-specific non-functional feature mapping tables for simplicity [9]. In some references, a method for non-measurable items was presented in the form of documentation activities and tests [9], [10]. Some references also use a coefficient with FP to include non-functional requirements [21] [22], [23].

Evaluation using FP was suggested after Alan Albrecht's paper of IBM, which first proposed it in 1979 [22] and then in 1984 [21]. The rules and standards then were

[8] Functional Size Measurement.

[9] Constructive Cost Model.

developed in the IFPUG group [23]. In [23], the comparison is very interesting, between what Alan Albrecht developed in 1979 and 1983. In the latter article, written four years after former, Albrecht considers the quality factor to be 10% higher and has increased the number of quality characteristics he wants; IFPUG also went on to give more importance to NFRs by introducing a new metric for a measurement called SNAP.

The assessment of non-functional requirements in the SNAP, expressed in terms of categories and subcategories in [8], has been in completing the FP deficiency; Non-functional requirements in FP, which were presented in Albrecht's 1979 paper, but did not fully reflect the effort - and ultimately the cost - to implement non-functional requirements in FP, therefore increased the adjustment factor, but still, FP, does not take into account the efforts in different phases, therefore in [8], the discussion is about to show the correlation of the obtained SNAPs with the effort and price of the software product. Basically, FP only measures the amount of data flow and storage in the software and does not consider other requirements that need effort. Therefore, for completing FP, and to increase the quality and accuracy of software evaluation, IFPUG introduced the SNAP method for assessing the size of non-functional requirements to the industry and the research community [8]. Completed Version 2.1 of this FP, entitled Assessment Practices Manual, was presented in [24] by the SNAP project team at IFPUG; The results of the beta test to measure and correlate the SNAP with the amount of work effort are shown in [8]. The results of this test are very important; For example, to calculate the cost of building a house, $100 per square meter, if we consider its area, the cost is obtained by multiplying these two factors. In order to calculate the required cost of the software with FP, an analyst uses [6] to extract the amount of data and storage requirements, through the number of FPs according to Table 2-2, and finally by calculating the total number of FPs, which it was also stated in [25] that the necessary cost can be obtained. The number of obtained FPs is correlated with the required cost. This was the method first used by Allen Albrecht and showed the correlation of cost- FPs in [22] and [21]. With the success of various organizations using this method and sharing their organizational data, commercial software estimation tools, which used FP and other productivity indicators (such as software language, programming team skills, used project management tools, etc.), were developed to help the software applicant and could estimate the necessary development cost better [8]; By studying [6], we find that the debility of FP eventually return to the view of the data flow and its storage!. In other words, the same lack of attention to other non-functional requirements; While we have variety of non-functional requirements; In the case of building a house above, the cost of construction certainly varies according to the choice of mosaic, ceramic, high-quality parquet floors, etc. The purpose of the SNAP is to include such considerations, not seen in [6]. For this reason, in [8], various categories and subcategories have been considered. Considering all respects of the above non-functional requirements are intended to provide a comprehensive view - not just the aspect and view of the data flow and storage - of the software product.

Ref [20] presented a simple evaluation model of NFRs included in the RFP, mainly focusing on the user maintenance and operation issues. This model consists of NFR categories, NFR metrics, description level grading and weight to each NFR. To quantify the linguistic expressions that are expressed in fuzzy, the values of 0, 2, 3, 4, 5 were also used according to the level of coverage (not mentioned, mentioned vaguely, mentioned, specified without value, clear with value). Point field in Table 1 is adjusted based on aforementioned.

3. Research Methodology

Consider the stable system S_0 that changes to S_1 unstable state because of its need to develop. If the development decision is approved, the requirements of Req_a will be recognized in response to N_a needs. If all the requirements of Req_a are applied correctly, the system will reach a stable state of S_a, which will be the first development of the S_0 system. This is illustrated in the following figures:

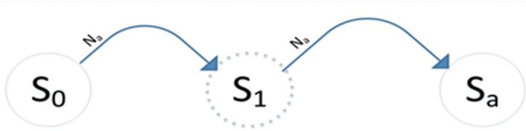

Figure 1. Unstable state among two stable states

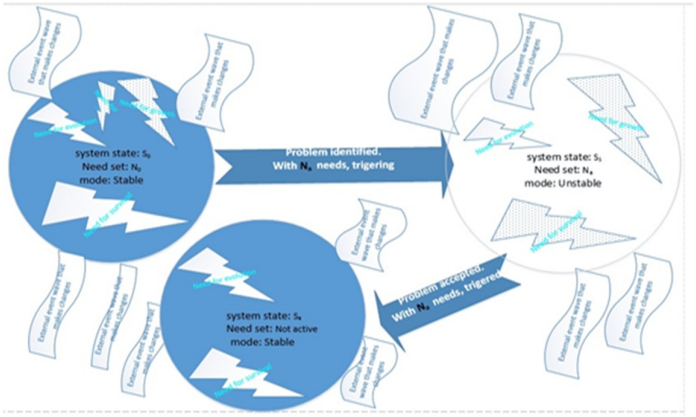

Figure 2. Wave of events and changes stimulus development

In order to move from the software corresponding to S_0 current system (As-Is), which has the set of N_a requirements, to the software corresponding to S_a future system (To-Be), the satisfaction of requirements of Req_a is necessary. The RFP document contains the S_0 description, the S_a description, and the Req_a (or at least N_a) set. Different developers using the mentioned factors, have defined requirements, development and maintenance stages according to their models, and finally provide their own model of S_a.

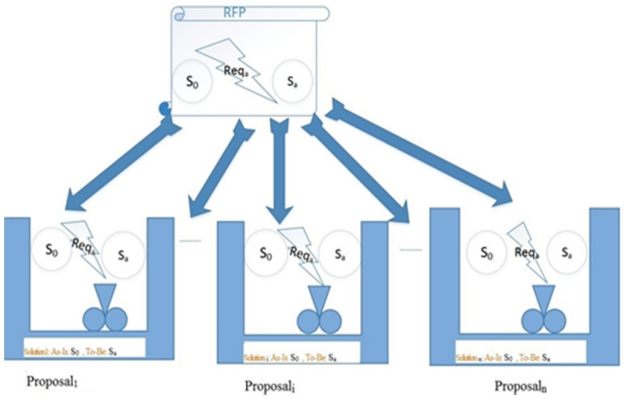

Figure 3. Different proposals of developers for a specific RFP

If the RFP document does not have a Req_a, the various developers will not consider the corresponding set of requirements and responses in their proposal, hence the price estimate will be independent of the issue, or each considers a set of requirements from its point of view. In latter case, the comparison of proposals and pricing, has no fixed criteria and the analogy will be different, but if the RFP document has Req_a, which is comprehensive and restrictive, we want to compare different developers based on their proposed solution, of course, based on their proposed models of S_a, So two related works need to be done to determine the quality, which we will summarize first and then in details:

- First, ensuring that the RFP is comprehensive and restrictive, especially in non-functional requirements (completeness of expression or coverage) and measuring it.
- Second, determining the quality of the software features corresponding to the non-functional requirements (in-depth) in each of the various components of the proposed solution. This should be specified in the proposals of the developers and should be compromised in the meetings between them and the applicant.

Software project management starts with a set of activities called project planning. Before the project can be started, the project manager and the software team must estimate the project and determine the required resources and the expected time spent from beginning to end. Software planning involves estimating the time, effort, money, and resources needed to build a particular software system. After the project area was determined and the problem was broken down into smaller issues, software managers use the project history data and their professional experience to make estimations for each of the issues. The final estimations are also adjusted according to the complexity of the problems. Software development is expensive and costly, consequently exceeding the estimated cost can be dangerous for developers. Usually many factors, such as human, technical, environmental, and political factors, can be effective in estimating the final cost. In order to obtain reliable cost and effort estimations, various options are in software engineering resources. The effectiveness of most of these methods depends heavily on

the historical data they use. To estimate the project, especially when the project is complex, the method of breaking problems into a set of smaller issues is used.

Decomposition is done in two ways: problem decomposition and process decomposition. Before making an estimation, an estimation of the software size must be made. Because project estimation accuracy depends on software size accuracy, this estimation is the first project plan challenge. This estimation is done in four ways: fuzzy logic, FP, standard components, and changes. Problem-based estimation is done in two ways: LOC and FP; Decomposition using LOC done with an emphasis on software operations and estimation using FP done with an emphasis on information domain properties. In the process based estimation, the decomposition is based on the tasks required to complete the software process framework. The use case estimation is considered useful, but it has a debate due to deficiencies in standardization. Software estimation models use empirically derived formulas to predict the amount of effort as a function of LOC[10]or FP. Usually, because empirical data was obtained from a limited number of previously completed projects, no estimation model is appropriate for all projects and environments. An estimation model should be developed to reflect the specific conditions of the project. These models are usually extracted by analyzing the data of historical projects and are performed with person-month as the dependent variable and LOC, FP, or object-dependent variables as the independent variables. For example, the COCOMO model or constructive cost, which is an example of a static estimation model, or the COCOMO ‖ and software equation models, which are dynamic estimation models. Unfortunately, none of the above methods are very useful for a software applicant company who is just starting in RFP and wants to have a practical price estimation or to compare the proposals of developer companies.

On the other hand, various and different non-functional requirements can be identified in different parts such as product, project, process. In the software of different business areas, some of them are more important and some are less important. Due to the limited credits and priorities of the projects, we must focus on some of them and ignore some. While some of the non-functional requirements automatically have convergence or divergence. We consider the maximum expressing coverage of non-functional requirements in RFP according to the corresponding characteristics of the top software in that business domain (or with the best RFP in that domain if available) and the distance between these two in each non-functional requirement indicates RFP defects, Which should be considered and its coverage should be strengthened. This action is very important in ranking proposals, otherwise the bidder may mention non-functional requirements in his proposal that are not relevant to the business area at all. On the other hand, the score that each proposed plan receives in response to any non-functional requirement should not exceed the share of it in the RFP document. Because it is the applicant company that has determined which non-functional requirements are more important, according to its priorities (credit, time, strategic decisions, security, etc.). In other words, the score of each software quality attribute, its share in the whole project, and ultimately its depth, will be contained in the RFP document. This causes the applicant company, regarding its credit cost, to gain software related to its priorities. Obviously, if

[10] Line of Code.

the goal is to achieve software higher than the existing scale (national or global) in that class, the existing ceiling should not be considered .

The greater the coverage of functional requirements at a given domain level, the larger the size of the solution. Larger size affects the price, but in the research assumptions, we have assumed complete coverage of the functional requirements in the received proposals. In other words, the solutions were assumed to be complete and of the same size, and we focused on comparing non-functional requirements.

We calculate the coverage of non-functional requirements for each of the functional requirements of the domain. As this spectrum of coverage of non-functional requirements for functional requirement increases, the quality of the solution of that functional requirement becomes more transparent and dimensional, and more aspects of the quality of the solution become apparent - these aspects are called qualitative characteristics in the final product. The more the development of the non-functional requirements in the depth increases, the degree of quality of the functional requirement increases. Note that non-functional requirements have many dimensions and each non-functional requirement is related to a functional requirement. Therefore, in product quality rating, the weight of each dimension of non-functional requirements at the level of a functional requirement, as well as the weight of that functional requirement at the solution level should be considered. Even the weight of the DM[11] views must be taken into account.

In view of this research, non-functional requirements can be proposed at any level and in any functional requirement. In other words, we consider the range of non-functional requirements to be very wide. As we define the issues of product quality, project, business processes, and other constraints and requirements of different levels in this area. However, the weight of non-functional requirements related to an functional requirement - which may itself be low in the whole project - is naturally lower than the weight of non-functional requirements of the same kind, for example in the field of infrastructure, etc. Here we turn to the software aspects. Apart from infrastructure, other factors that have a great impact on the weighting of non-functional requirements are the weight of this type of requirements in objects that are at a higher level in terms of granulation, such as business and its various functions, project, software product, processes and other requirements and constraints that have a great deal of control over a large part of the objects of a system. That is why in this research, for the convenience of conducting a case study, we confine ourselves to a higher level of grading and omit less important non-functional requirements. For simplicity, we may not consider all non-functional requirements in practical applications, and consider important items that play a more prominent role in the quality and cost of applications. See the comprehensive RFP and proposals evaluation model in Table 1.

To evaluate the Coverage of non-functional requirements we have:

$$Score_k(m_i) = \text{5-grade of coverage evaluation of } m_i * \text{ weight of } m_i \quad (1)$$

Note: Weight can be obtained with a questionnaire from the companies using that software. Score(M), Score(H) are calculated based on weighted average.

[11] Decision Maker.

Table 1. Coverage evaluation table

Number of mentioned requirements corresponding to **maintenance** and **operation**				Number of NFR pages or clauses within RFP					Number of RFP pages or clauses	
. . . . .				. . . . .					. . . . .	
NFR	**High Level**	**Middle Level**	**NFR Metrics**	**Grade**	**W**	**RFP** Clauses	**point**	**Score(m)**	**Score(M)**	**Score(H)**
Operation requirements	Ready for Operation	Operation Test	**Acceptable Fault Occurrence Frequency at Start of Operation**		6.0					
		Operation Start condition	**Test Case Density**		2.6					
	Evaluation of system Operation	Operation Easiness	**Minimum of User Interaction Operation**		1.9					
			Easiness of User Interaction Operation		1.9					
		Availability Factor	**Average Availability Factor**		5.3					
			Online System Availability Factor		1.0			–		
		Availability Factor of Quality	**Success ration of Batch Processing**		6.2					
			Response Time		3.7					
			Throughput		3.6					
			Stoppage Time		1.3					
	Operational Monitoring	Condition of Abnormal Detect	**Detecting Time**		1.7					
			Access Control		5.4					
		Error Handling	**Data Loss Avoidance**		1.9					
	Countermeasure of Failure	Countermeasure of Failure	**Backup Method**		3.6					
		Disaster Countermeasure	**Wide Area Disaster Measure**		1.0					
Maintenance Requirement	Maintenance Productivity	Understand the Problem and modification analysis	**Logic Acquisition**		4.3					
			Log Storage Period		1.0					
		Maintenance Easiness	**Malfunction Trace Tool**		1.9					
			Test tool		1.9					
			Maintenance Document		4.9					
	User Support	User Support	**Working Hours**		1.3					
			Window Hours		1.3					
			Training Frequency		1.3					
			Target Audience		1.3					

There is another table such as Table 1, instead for qualitative assessment. In comparison, this includes the depth of quality specifications related to maintenance and operation. It has instead Qpoint, QScore(m), QScore(M) and QScore(H) titles that evaluates amount of quality characteristics corresponding to maintenance and operation for a system. Qpoint field is adjusted based on Table 2 and so we calculate all other calculations such aforementioned Table 1.

Table 2. Relationship between measured value and the major levels of grading

major levels of grading	range of major values	The exact determined value Adjustment
Great	15-20	full satisfaction
Well	10-15	
Medium	5 - 10	Range
Weak	0 - 5	
		Dissatisfaction

In addition to the formulas mentioned in [23], in which the $Score_k(m_i)$ function describes the coverage of non-functional requirements, by defining a new function $QScore(NFR_i)$ to evaluate the quality of non-functional requirements we have:

$$QScore_k\left(m_i\right) = 4 - grade\ quality\ evaluation\ of\ m_i * weight\ of\ m_i \quad (2)$$

Note: The domain of the $QScore_{fr}(NFR_i)$ is equivalent to the level of an functional requirement, and the domain of $QScore_p$ is equivalent to the total level of the project p.

We also have for any non-functional requirement at a functional requirement level:

$$Score_{fr}\left(NFR_i\right) = \frac{\sum_{m \in NFR_i} Score\left(m\right)}{\left|NFR_i\right|} \quad (2)$$

$$QScore_{fr}\left(NFR_i\right) = \frac{\sum_{m \in NFR_i} QScore\left(m\right)}{\left|NFR_i\right|} \quad (3)$$

Where $|NFR_i|$ is the number of sub metrics for NFR_i. At the level of a DM we have:

$$Score_{DM}\left(NFR_i\right) = \frac{\sum_{J=1}^{|FR|} \mathbf{W\_FR}_j * \left(\sum_{m \in NFR_i} Score\left(m\right) / \left|NFR_i\right|\right)}{\sum_{J=1}^{|FR|} \boldsymbol{W\_FR}_j} \quad (4)$$

$$QScore_{DM}\left(NFR_i\right) = \frac{\sum_{J=1}^{|FR|} \mathbf{W\_FR}_j * \left(\sum_{m \in NFR_i} QScore\left(m\right) / \left|NFR_i\right|\right)}{\sum_{J=1}^{|FR|} \boldsymbol{W\_FR}_j} \quad (5)$$

Where | FR | is the total number of functional requirements for which, non-functional requirements are addressed.

If there are various DMs with different weights, each of which has an opinion on the weight of FR_i and NFR_i, then the above calculation at the first project level (P) should be completed as follows:

$$Score_P\left(NFR_i\right)=\frac{\sum_{j=1}^{|DM|}\mathbf{W\_DM}_j * Score_{DM}\left(NFR_i\right)}{\sum_{j=1}^{|DM|}\mathbf{W\_DM}_j} \tag{6}$$

$$QScore_P\left(NFR_i\right)=\frac{\sum_{j=1}^{|DM|}\mathbf{W\_DM}_j * QScore_{DM}\left(NFR_i\right)}{\sum_{j=1}^{|DM|}\mathbf{W\_DM}_j} \tag{7}$$

Where | DM | is the total number of DMs that meet the non-functional requirements in a particular functional requirement.

Non-functional requirements must be calculated for each of the functional requirements and the lower the level of granulation and abstraction the more accurate, that it is suitable if there is a mechanized calculation. However, because of simplicity and manual work we assume the functional requirements with the same weight, we consider the calculation for the most efficient part, i.e. the high-level of granulation (system level) and common for the total functional requirements. It should be noted that if the first case is used, the below formula should be used.

$$QScore_{DM}\left(NFR_i\right)=\frac{\sum_{J=1}^{|FR|}\mathrm{W\_FR}_j *\left(\sum_{m\in NFR_i}\mathbf{Q}Score\left(m\right)/\left|NFR_i\right|\right)}{\sum_{J=1}^{|FR|}W\_FR_j} \tag{8}$$

This means that for each functional requirement, the formula $Qscore_{fr}\left(NFR_i\right)=\frac{\sum_{m\in NFRi}\mathbf{Q}Score\left(m\right)}{\left|NFR_i\right|}$ must be calculated; in other words the evaluation table must be calculated for each functional requirement, then for each functional requirement, the above calculated amount must be multiplied by the weight of the functional requirement, hence instead of being calculated at the level of an functional requirement, it was calculated at the more general level (DM or project level). This is important because a non-functional requirement may have a high weight in itself but is dependent on a functional requirement, which has negligible weight throughout the project, so this is addressed with the above formula. In addition, the denominator of the relation, $\sum_{J=1}^{|FR|}\boldsymbol{W\_FR}_j$, was used for normalization. The above explanation is also true for DMs at the first project level, which use the formula $QScore_P(NFR_i)=\frac{\sum_{j=1}^{|DM|}\mathbf{W\_DM}_j * \mathrm{Q}Score_{DM}\left(NFR_i\right)}{\sum_{j=1}^{|DM|}\mathbf{W\_DM}_j}$. This means that for each DM, an evaluation table must be calculated for each functional requirement.

Development of the proposed model:

- We do not restrict the model to just two non-functional requirements.
- We do not limit the quality perspective only to the perspective of the user company.
- In [23], the weight allocated to the metrics of each non-functional requirement in the model is obtained with respect to the acceptability of that metric in the general questionnaire in [29], and this was to have a common basis for all measurable projects .However, to compare the proposals of a project and a particular RFP, DMs can alter the weight with trading off respect to the needs and conditions of applicant's company.

4. Evaluation of research results

RFPs from 6 projects were selected as a case study.

i. Evaluation of the coverage and the degree of static quality of the relevant proposals were done according to Tables 1, 2 and the above. The final summary of the calculations shown in the following tables:

Table 3. Final summary of the coverage calculations in low level metrics for 6 systems

NFR Metrics	Score(m)					
	FIX CRM	PAYAM	INF118	BILLING	ADSL CRM	AAA system
Acceptable Fault Occurrence Frequency at Start of Operation						
Test Case Density						
Minimum of User Interaction Operation				9.5		
Easiness of User Interaction Operation		3.8	3.8	9.5		
Average Availability Factor		26.5	26.5			
Online System Availability Factor	5	5	5	5		5
Success ration of Batch Processing						
Response Time		18.5	18.5	18.5		
Throughput				18	18	18
Stoppage Time	6.5	6.5	6.5	6.5		
Detecting Time	8.5		8.35	8.5		
Access Control		10.8		21.6		16.2
Data Loss Avoidance		3.8	3.8	9.5		
Backup Method	7.2	14.4	7.2	14.4	7.2	7.2
Wide Area Disaster Measure			5		3	3
Logic Acquisition		12.9	8.6	17.2	8.6	8.6
Log Storage Period				4		
Malfunction Trace Tool	7.6	3.8	3.8	3.8		
Test tool				3.8		

Maintenance Document	9.8	9.8	14.7	19.6		
Working Hours	6.5	6.5	6.5		6.5	6.5
Window Hours		6.5	6.5		6.5	6.5
Training Frequency		5.2	2.6	6.5	5.2	5.2
Target Audience	3.9	3.9	3.9		3.9	3.9

Table 4. Final summary of the coverage calculations in middle level metrics for 6 systems

Middle Level	Score(M)					
	FIX CRM	PAYAM	INF118	BILLING	ADSL CRM	AAA system
Operation Test						
Operation Start condition						
Operation Easiness		1.54	1.54	9.5		
Availability Factor	2.5	15.57	15.57	2.5		2.5
Availability Factor of Quality	1.62	6.25	6.25	10.75	2.5	2.5
Condition of Abnormal Detect	4.25	5.4	4.17	15.05		8.1
Error Handling		3.8	3.8	9.5		
Countermeasure of Failure	7.2	14.4	14.4	14.4	7.2	7.2
Disaster Countermeasure					3	3
Understand the Problem and modification analysis		6.45	4.3	10.6	4.3	4.3
Maintenance Easiness	5.8	4.53	6.17	9.07		
User Support	2.6	5.52	4.88	1.62	5.52	5.52

Table 5. Final summary of the coverage calculations in high level metrics for 6 systems

High Level	Score(H)					
	FIX CRM	PAYAM	INF118	BILLING	ADSL CRM	AAA system
Ready for Operation						
Evaluation of system Operation	1.44	7.54	7.54	8.38	2.25	2.88
Operational Monitoring	2.83	4.87	4.05	13.2		5.4
Countermeasure of Failure	3.6	7.2	7.2	7.2	5.1	5.1
Maintenance Productivity	3.48	5.5	5.42	9.68	1.72	1.72
User Support	2.6	5.52	4.88	1.62	5.52	5.52

Table 6. Degree of selected NFRs coverage for 6 systems

Selected NFR	Score(NFR)					
	FIX CRM	PAYAM	INF118	BILLING	ADSL CRM	AAA system
Operation requirement	1.81	1.93	5.64	8.07	1.88	3.29
Maintenance Requirement	3.09	5.40	5.18	6.10	3.41	3.41

Table 7. Final summary of the quality calculations in low level metrics for 6 systems

NFR Metrics	QScore(m)					
	FIX CRM	**PAYAM**	**INF118**	**BILLING**	**ADSL CRM**	**AAA system**
Acceptable Fault Occurrence Frequency at Start of Operation						
Test Case Density						
Minimum of User Interaction Operation				28.50		
Easiness of User Interaction Operation				19		
Average Availability Factor		21.2	21.2			53
Online System Availability Factor	10	4	4	18		10
Success ration of Batch Processing						
Response Time		55.50	55.5	55.5		
Throughput				36	7.2	18
Stoppage Time	13	19.5	19.5	23.40		
Detecting Time	17			25.50		
Access Control		21.6	21.6	54		27
Data Loss Avoidance		19	19	19		
Backup Method	18	54	54	43.20	18	18
Wide Area Disaster Measure					5	5
Logic Acquisition		17.2	17.2	21.5	8.6	21.5
Log Storage Period				5		
Malfunction Trace Tool	9.5	5.7	5.7	9.5	28.5	28.5
Test tool				9.5		
Maintenance Document	24.5	14.7	14.7	49		
Working Hours	19.5	19.5	19.5		19.5	19.5
Window Hours		19.5	19.5		19.5	19.5
Training Frequency		13	13		19.5	19.5
Target Audience		13	13		6.5	6.5

Table 8. Quality calculations in high level metrics for 6 systems

High Level	QScore(H)					
	FIX CRM	PAYAM	INF118	BILLING	ADSL CRM	AAA system
Ready for Operation						
Evaluation of system Operation	2.88	12.52	12.52	10.75	0.9	10.12
Operational Monitoring	5.67	13.53	13.53	32.83		9
Countermeasure of Failure	9	27	27	21.6	11.5	11.5
Maintenance Productivity	6.8	7.32	7.52	18.9	7.42	10
User Support	4.88	16.25	16.25		16.25	16.25

Table 9. Degree of selected NFRs quality for 6 systems

Selected NFR	QScore(NFR)					
	FIX CRM	PAYAM	INF118	BILLING	ADSL CRM	AAA system
Operation requirement	3.87	12.99	12.99	21.47	2.01	8.73
Maintenance Requirement	5.94	11.40	11.40	10.50	11.34	12.78

ii. To estimate their dynamic quality, we also used a data model in which the raw information of that was prepared from the following principles:

• Recorded historical data about business systems such as history of defects, failures, start and end times of failures, etc. For instance, the status of failure samples of the studied systems in a period of about eight months, from the first day of the timeline to its end, can be seen in the following figure:

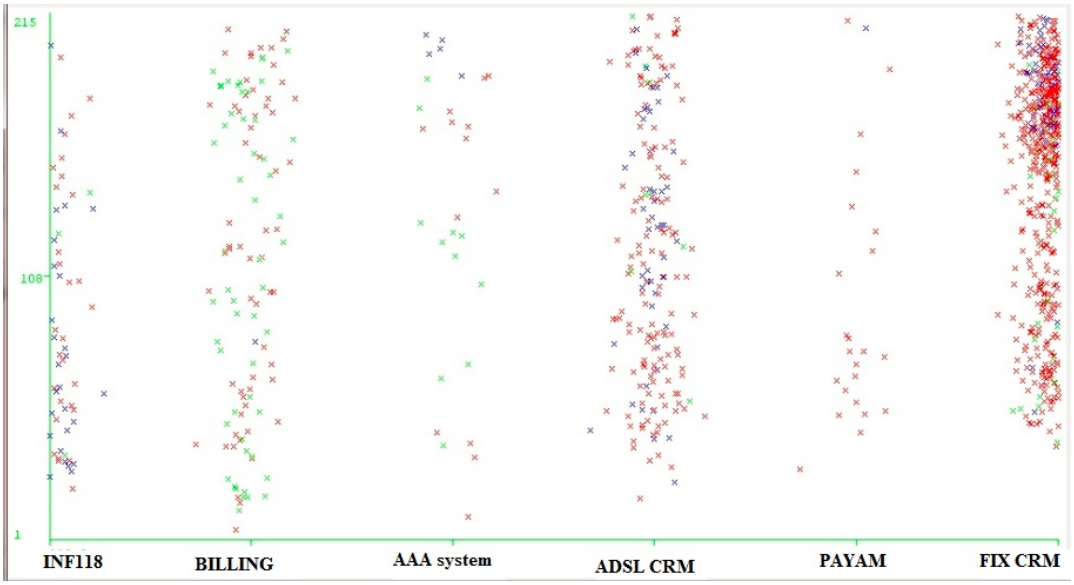

Figure 4. Samples of failures in the 6 studied systems

• Report on tickets received from end users or sent to software development companies, from the internal telecommunication ticketing system, as well as ticketing systems of software development companies.
• Reporting letters, announcing the urgency of the problem, etc., from the automation system.
• Possible warnings, confiscation of guarantees, announcement of dismissal, etc. from the field of procurement.

iii. We loaded the combined table containing the characteristics of the two static and dynamic states, resulting from the previous two steps, into WEKA. We want to investigate the relationship between static and dynamic qualitative attributes. We used two dynamic attributes, Abnorm_C (N) and Dur_min (N), which were calculated in the previous section of the data for each software as follows:

$$\text{Dur\_min (N)} = \frac{\text{The total sum of failure time}}{software\ size}$$

$$\text{Abnorm\_C (N)} = \frac{number\ of\ failure}{software\ size}$$

First, we try to show the correlation of Dur_min (N) with static qualitative attributes. We use the Linear Regression function in WEKA, the following results are obtained by the model maker:

- Model I: First, we include only the highest level characteristics i.e., Score (NFR1), Score(NFR2), QScore(NFR1) و QScore(NFR2)- which corresponds

to the amount of coverage and quality of operation and maintenance requirements respectively- in modeling, which yields the following result:

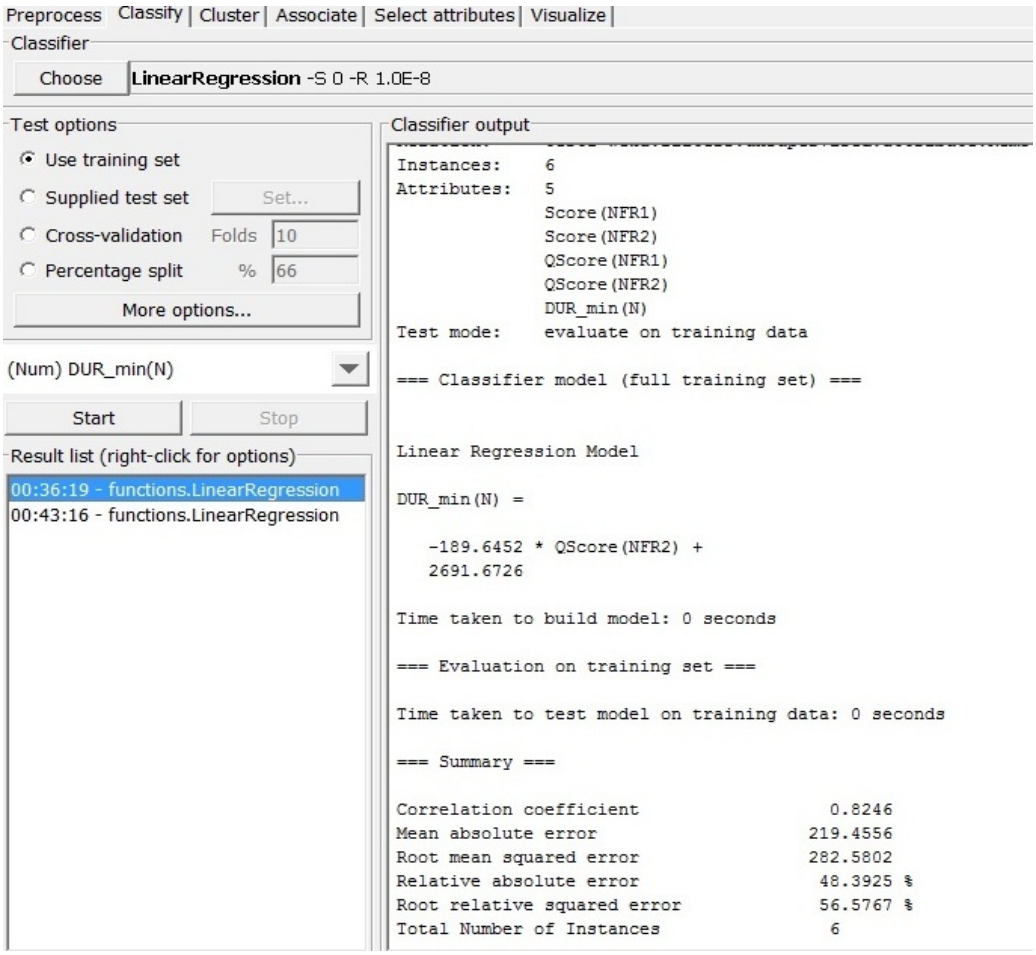

Figure 1. Model 1 with regression

- Model II: In the following experiment, we try to increase the accuracy, and discover that which lower-level criteria exactly are involved in this correlation. Thus, at this time we use the following features:

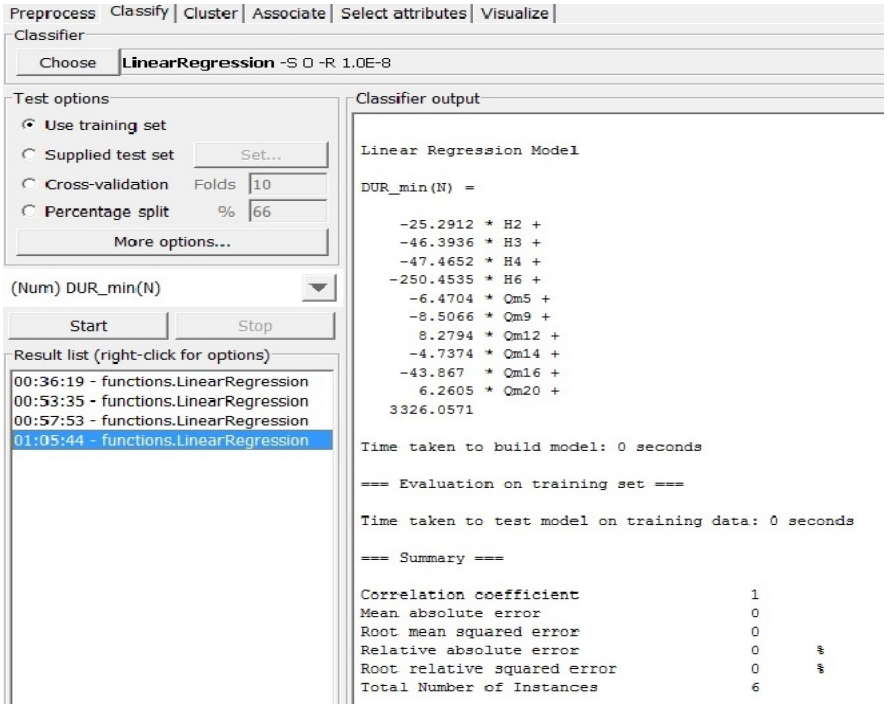

Figure 1. Model 2 with regression

- ✓ H1-H6: H1 to H4 are the high-level metrics of NFR1 in the coverage section, H5 and H6 are the high-level metrics of NFR2 in the same section.
- ✓ Qm1-Qm24: Which are low-level metrics of NFR1 and NFR2 in terms of quality.

In the above experiments, we tried to analyze the correlation between static and dynamic qualitative attributes. For the static quality features studied, we selected two attributes of operation and maintenance. We decomposed each of these requirements into sub-attributes at three levels of abstraction. Finally, we used metrics to measure each of the metrics at each of the three levels of abstraction. We also considered measures for the dynamic state of the software in the operating environment, for which we selected about 1000 failure records from the list of actual failure events.

Because software failure depends on their size, we used the FP Rate ratio to normalize the number and duration of software failure. For this reason, we used two dynamic attributes, Abnorm_C (N) and Dur_min (N), which are the normalized form of the total number and failure duration of the software.

We proved the correlation between static and dynamic qualitative features by constructing a linear regression model using the Linear Regression modeling function at the level of the studied software.

Regression line coefficients and their impact rate, help us to understand what we need to reinforce in the RFP and proposal in order to have a software with more suitable dynamic state, for example in terms of duration and number of failures. Although we do

not expect that MSE[12]=0 and the correlation coefficient which obtained 100%, are still valid for samples with a large number of software. However the existence of a correlation is undeniable and cannot be conclusively stated when it is not examined in practice with large actual samples.

5. Conclusion

The comprehensive model for evaluating RFPs and proposals presented in this study works better than of Albrecht in [21], [22] and SNAP methods because it emphasizes the real requirement and also the various aspects and perspectives by assigning weight for the model. It has done the calculation of quality with the most accurate mode - of course, according to the selected granulation - and not with purely statistical and general coefficients. While Albrecht focuses only on data flow and storage in the software structure and SNAP refers to fixed and limited categories and aspects. The two mentioned methods for validation, also used the correlation of the proposed model with the amount of effort, while we used the correlation of the presented static model with the actual dynamic state of the software for the validation of our model.
Our static model, complemented by the RFP data and proposal, expresses the desired status of the software - what has been agreed and should be - and indicates a measure of committed quality, but FP, and even SNAP, measure the developer structure, whereas part of them may be useless, and was a waste of effort. On the other hand, the dynamic model, which we used to validate our static model, shows the real state of the software - what it is. In other words, our proposed model actually compares the current state with the desired one and presents a qualitative standard deviation.
In summary, the ranking of proposals with a static quality model provides an estimate of the quality of the software in the actual operating environment. Therefore, it can be used for measurement and pricing.

References

[1] Supreme Informatics Council, "Nematon: Software Standards", 1385.

[2] Technical Commission of Informatics Council, "Software Engineering Standard", 1387.

[3] Software Engineering Center and Information-Technology Promotion Agency of Japan, "Non-Functional Requirements Grades Usage Guide [Description Manual]", 2010.

[4] Julian Sammy," Needs and Solutions, Requirements and Designs

Business Analysis Core Concept Model", IIBA, 2018.

[5] Renata Guizzardi, Feng-Lin Li and Alexander Borgida, "An Ontological Interpretation of Non-Functional Requirements," Foundations for Software Evolution, 2016.

[6] International Function Point Users Group (IFPUG),"Counting Practices Manual (now in version 4.3)," Princeton Junction, New Jersey, USA, 2009.

[7] ISO/ IEC 20926:2009 Software and Systems Engineering, "Software Measurement," IFPUG Functional Size Measurement Method 2009, retrieved from <http://www.iso.org, 2012.

[8] Tichenor Charley, "A New Software Metric to Complement Function Points The Software Non-functional Assessment Process (SNAP) ,",CrossTalk-july, 25th anniversary issue, 2013.

[9] Aguiar Mauricio, "When metrics mean business," TI Métricas, 2015.

[12] Mean squared error.

[10] Netherlands Software Metrics Users Association, "Function Point Analysis for Software Enhancement – Guidelines," Version 1.0, NESMA, 2001.

[11] IFPUG, “unpublished internal reports," IFPUG Princeton Junction, 2012.

[12] Government of Victoria, Australia, " southernSCOPE Reference Manual, Version 1,"Government of Victoria, Victoria, Australia, http://www.egov.vic.gov.au/pdfs/SSRefManual.PDF, 2000.

[13] Boehm B. et al, "Quantitative Evaluation of Software Quality" in Proceedings of the 2nd IEEE International Conference on Software Engineering, 1976.

[14] L. Chung, B. Nixon, E. Yu, and J. Mylopoulos, "Non-functional Requirements ," Softw Eng, 2000.

[15] Software Engineering Center and Information-Technology Promotion Agency of Japan, "Non-Functional Requirements Grades Usage Guide [Description Manual]", 2010.

[16] Software Engineering Center and Information-Technology Promotion Agency of Japan, "Non-Functional Requirements Grades Usage Guide [Usage Manual]", 2010.

[17] Lawrence B, Wiegers K and Ebert C, "The top risk of requirements engineering,"IEEE Softw 18(6):62, 2001.

[18] Di Noia, T., Mongiello, M., Nocera, F. et al," A fuzzy ontology-based approach for tool-supported decision making in architectural design," Knowl Inf Syst, 2018.

[19] Matinee Kiewakanya and Pornsiri Muenchaisri,"Measuring Maintainability in Early Phase using Aesthetic Metrics," 4th World Scientific and Engineering Academy and Society, 2005.

[20] Saito Yasuhiro, Monden Akito and Matsumoto Kenichi, "Evaluation of Non Functional Requirements in a Request For Proposal," in Seventh International Conference on Software Process and Product Measurement, IEEE, 2012.

[21] Albrecht, A. J., and Gaffney, "Software Function,Source Lines of Code, and Development Effort Prediction:A Software Science Validation,"J. E, IEEE Trans. Software Eng. SE-9, 639-648, 1983.

[22] Albrecht, A. J, "Measuring application development productivity," in Proceedings IBM Applications Development Symposium, Monterey, California, October 14-17, 1979.

[23] Alain Abran and Pierre N. Robillard, " Function Points:A Study of Their Measurement Processes and Scale Transformations ," J. SYSTEMS SOFTWARE, Elsevier Science Inc, 1994.

[24] International Function Point Users Group (IFPUG), "Software Non-functional", 2012.

[25] Dekkers and Carol.Musings,"About Software Development", retrieved from <http:/ /caroldekkers.blogspot.com,November 5, 2012.

[26] Ministry of Economy, Trade and Industry and Mitsubishi Research Institute Inc, "Product Quality Metrics WG Activities in 2010 Software Metrics Advancement Project”, 2010.

[27] Rezaali Mosayyebi and Homayun Motameni, "A practical approach for evaluating the software projects based on a Request for Proposal (RFP) focusing on Non Functional Requirements", the First International Conference on new Achievements in Science and Technology,2017.

[28] M. Akbari, "An efficient genetic algorithm for task scheduling on heterogeneous computing systems based on TRIZ," Journal of Advances in Computer Research, vol. 9, pp. 103-132, 2018.

[29] M. Hosseini, "A new Shuffled Genetic-based Task Scheduling Algorithm in Heterogeneous Distributed Systems," Journal of Advances in Computer Research, vol. 9, pp. 19-36, 2018.

[30] B. Barzegar, S. Habibian, and M. Fazlollah Nejad, "Heuristic algorithms for task scheduling in Cloud Computing using Combined Particle Swarm Optimization and Bat Algorithms," Journal of Advances in Computer Research, vol. 10, pp. 83-95, 2019.