Architecture Style Selection using Statistics of Quality Attributes to Reduce Production Costs

Hamidreza Hasannejad Marzooni[1], Homayun Motameni[2], and Ali Ebrahimnejad[3]
[1]Department of Computer Engineering, Babol Branch, Islamic Azad University, Babol, Iran
[2]Department of Computer Engineering, Sari Branch, Islamic Azad University, Sari, Iran
[3]Department of Mathematics, Qaemshahr Branch, Islamic Azad University, Qaemshahr, Iran

Abstract: *As time goes by and software systems grow in complexity and size, there is an increasing need for software architecture as an important tool in software design. Designing an appropriate architecture is necessary in producing a high-quality software, which also suits stakeholders. In order to design the desired high-quality software program, style-based architectures can be used. That is, with the selection of appropriate style architecture, we will get an ideal architecture for design. With the same attitude in this research, using a statistical computational algorithm, we have attempted to select the appropriate software architecture style to meet stakeholders' requirements. In meeting Non-Functional Requirements (NFRs) of stakeholders, increase of one NFR does not increase the others necessarily, and they may be at odds with each other, thus the best quality for all cannot be achieved. In the designing stage of an ideal software, we must take into account the production and maintenance costs as well as a trade-off between stakeholders' desired needs. The proposed algorithm structure involves a method using Gamma Probability Distribution Function (PDF). In a way that, a statistical estimate for each present style is created, and finally in the design of the software, the best style (based on the mentioned statistical estimate) is used for meeting the stakeholder's needs. The method not only creates NFRs in the software program, but also gives importance to production and maintenance costs. This requires that the qualitative data of the problem be converted into quantitative data. It will be fully described in the introduction to the algorithm. In order to verify the validity of the proposed algorithm, the resulted architecture style ranking will be compared with the results of alternative methods namely Analytic Hierarchy Process (AHP) and A Lightweight Value-based Software Architecture Evaluation (LiVASAE). The results confirm the applicability of the proposed algorithm and moreover it has less time complexity with respect to other methods.*

Keywords: *Software architecture style, non-functional requirements, curve fitting, gamma method.*

Received September 13, 2019; accepted June 18, 2020
https://doi.org/10.34028/18/4/3

1. Introduction

Since not so long ago, discussions have revolved around technology and its effect on life, to the extent that life without technology is seen unacceptable by humans. Although mankind has kept up with the speed of technology, technology has in some cases turned into a disruptor or destroyer causing havoc on the positive quality of life. All-round improvement in the quality of life in today's advanced societies is seen as a main issue. Given the considerable advances in information technology and the use of various software programs, one of the existing methods for improving the quality of life is this very technology, to the extent that it has brought sciences as different as architecture and construction, meteorology and crisis management, financial issues, factories and industries control system, vehicles' central control system, medical sciences and more under its direct control, bringing us to this conclusion that life without information technology will be impossible.

Given the special importance of software programs in human life, the science of software engineering is an essential need [40], which should be incorporated throughout the life cycle of producing a software program including analysis, architecture development, design, implementation, verification and maintenance phases [35, 41]. Modern societies also depend crucially on complex software systems which provide help in maintaining and satisfying stakeholders' goals and their inevitably changing needs. Therefore, the existence of a software program in the form of software architecture is a necessity in order to meet such demands.

Providing a complex, large-scale, distributed software engineering environment, the ability to quickly evaluate and improve software engineering practices can be a key differentiator of the market. Practices that shorten the development cycle, cost-effectively improve quality, and align the software with customer needs, leaving a direct impact on the business value provided by the company [47]. Therefore, software architecture is a basis for any kind of software system and a necessary mechanism for raising the software quality and gaining access to quality attributes [15]. The most important factor in ensuring the quality of software program is its architectural sustainability [42, 43]; throughout the life cycle of software production, its architecture endures. The architecture should be designed so that it leads to maintaining customer value in the short and long terms, thus bringing more architectural technical Debt (DBT) to the software structure.

Several methods have been proposed for better designing of software architecture, some of which will be referred to below. This paper presents a formal linear programming optimization model for the Non-Functional Requirements (NFRs) framework with regard to operationalization selection. Affleck *et al*. [3] used a formal linear programming optimization model for the NFRs framework with regard to the choice of operation. Decisions about software architecture depend on system failures. In his paper, Quaglia [37] investigated software diversity based on software diversity in the field of advanced simulation systems with the aim of improving the time taken to produce simulation outputs. They suggested using High-Level Architecture (HLA).

Sievi-Korte *et al*. [44] presented a paper in which the potential of Genetic Algorithms (GAs) is examined in the design of automated software architecture, assuming that the software architecture is made of certain patterns. For software development, the theory of constraints can also be used. The algorithm was provided by Ribeiro *et al*. [39] So far, various methods for providing the quality features of the software provided, such as the Jelinski Moranda (JM) model, are often used in software reliability [30]. Fieberg and Conn [18] also used the hidden Markov method, which presents the parameters of the Non-Homogeneous Poisson Process model (NHPP) to identify the software development defect detection process. The use of a Markov switching process allows us to identify non-uniform variations in the extent to which defects are found. This would better reflect the industrial application development environment [38]. The paper describes a systematic review of academic and industrial literature regarding architectural patterns and architectural tactics for micro services by Lefranc [29].

Moreover, software rejuvenation can be used for software maintenance. “Software rejuvenation” is a proactive technique intended to reduce the probability of future unplanned outages due to aging. Castelli *et al*. [14] analysed a rejuvenation policy based on prediction, demonstrating that it can further increase system availability and reduce downtime cost. This will instead increase stakeholders' trust. Building trust from the quality attributes would encourage the developer to induce these quality attributes in the development life cycle and produce a system whose foundation will be the stakeholders [8]. In architecture design, recognizing stakeholders' needs, the conditions of a problem and managing the concepts of architecture throughout the software life cycle play key roles in a project's success [5]. With designing an appropriate architecture, dependence of the software's quality on the code of a program decreases significantly, since the software's quality depends on the architecture model which faces model-based development [33]. Requirement Management (RM) is a fundamental activity which reduces errors, delay in software preparation and overrun costs [17]. The quality attributes of a software system are, to a large extent, determined by the decisions taken early in the development process. Best practices in software engineering recommend the identification of important quality attributes during the requirements elicitation process, and the specification of software architectures so as to satisfy these requirements. Over the last few years, the software engineering community has studied the relationship between quality attributes and the use of particular architecture styles and patterns [36].

These generally include points, strategies and methods which can be useful in designing and selecting the appropriate structure for software architecture.

Software reuse has been recognized as an attractive idea with an obvious payoff to achieve a faster, better and cheaper software program. One important component in designing reusable object-oriented software is design patterns. Design patterns describe a commonly recurring structure of communicating components that solve a general design problem in a particular context. An important property of design patterns is that they are independent of a particular application domain and programming paradigm. As a result, design patterns facilitate the reuse of software architecture, even when other forms of reuse are infeasible [2]. For instance, Béjar *et al*. [10] proposed an architecture style, a pattern, for Spatial Data Infrastructures (SDIs). This style provides a tool and a shared vocabulary to help system architects design these infrastructures, and facilitates the exchange of knowledge about them.

The new complicated requirements demand solution to newly arisen problems. Kalistratov [26] addressed the problem of wireless monitoring of the Megacities through simulating the propagation of a radio signal. Hadizadeh and Tanghatari [21] considered the problem of increasing processor efficiency using new parallel processing design of MIPS-Based Series. Jahanirad and Karam [24] used the Built-in Self-Test (BIST) based approach to test new configurations parallel proposed processing chips. These complicated new multi-processor, multi-task usages of software call for modern software styles, the efficiency of which should be evaluated using new functional and NFRs as the metric of evaluation. In this respect, style selection would be of ultimate importance to them.

There are several frameworks and middleware which result in savings in software implementation and production process. Some of them have been variously presented and used for certain systems and their capabilities have been proven [1]. The accurate selection of a set of such frameworks can prevent applying unwelcome changes when completing the desirable architecture known as software architecture style. This research mainly attempts to select an appropriate style when designing a software system with a determining role in leading to success. This study aims to present a style-based software architecture model and hence, investigates how the financial software architecture style of a relatively big meteorological organization involving 390 people as the case study is selected. In this case study, the appropriate software architecture style is carried out on Data Centred (DC), Data Flow (DF), Virtual Machines (VM), Remote Procedure Call (RPC), Object-Oriented (OO) and Layered (L) styles using the proposed algorithm. Some quality attributes of these styles, which will be of importance in this system, have been obtained using estimation questionnaire method and its completion by experts. The questionnaire output is used in the proposed algorithm in order to compare these styles with stakeholders' requirements and the selection of a model based on an ideal style [13]. In fact, an ideal style refers to a style relatively meeting the stakeholders' needs. Various algorithms have been presented in this regard, each of which suffering from certain shortages and defects mentioned in the second section.

This research presents a new algorithm based on mathematical reasoning and a statistical method, which selects a style able to meet stakeholders' NFRs with minimum amount of money spent within the shortest possible time. This introduces a simple, yet effective, method for selecting an optimum software architecture style, meeting the stakeholders' requirements with the least operation cost. It is itself an easy-to-design and -implement method (similar to AHP- and matrix-based methods) having low time complexity (like Modelling and Formal Methods) with low implementation cost, while achieving results comparable with what is obtained using complex methods. The paper has been arranged as follows. the studied methods are briefly described in section 2, the proposed algorithm is described step-by-step (and draft questionnaires are presented) in section 3, the Gamma distribution function and parameters estimation method are shortly described in section 4, discussions on the results of applying the proposed algorithm and compare with other methods are in section 5 and finally the section 6 covers the conclusions.

2. Related Works

One of the main subjects in designing a software architecture based on styles selection is the appropriate style. The term architecture style was first introduced by Perry and Wolf [34]. Garlan and Shaw [20] introduced software architecture styles and drew comparisons between them by providing several examples. Different research projects have presented different methods for the analysis and selection of styles. Bosch et al. presented an algorithm called arch designer, in which the prioritization and assignment of quality attribute weights have been used as criteria in selecting the most appropriate software architecture model or style [12]. In this algorithm, when the number of candidate styles and that of NFRs increases, the size of matrix grows. As a result, the number of calculations increases and leads to a reduction in efficiency.

Furthermore, Jabali *et al*. [23] used AHP algorithm based on the density of data for selecting a software architecture style or model, in which the implementation has not been conducted and the results have not been tabulated. Wang and Yang [46] also presented an algorithm based on AHP for style selection. Chun Yong Chong *et al*. [15] offered a fuzzy AHP-based algorithm in an effort to identify quality attributes and rank them based on their priorities. Kim *et al*. [27] proposed a Lightweight Technique for Software Architecture Evaluation (LiVASAE) based on arithmetic mean and AHP. AHP has a hierarchical one-way structure. This means that when ranking and selecting the best choice, the criteria list is assumed to remain unchanged. If the choice is to affect the criteria list, for example, by introducing new attributes of a candidate style, the output of calculations and as a prior results and ranking of the selected appropriate style will no longer be valid, so the problem needs to be reconsidered from the scratch. Considering such complexity, time and costly process of AHP-based algorithms, their applicability in real use is under question.

The correlation coefficient is another method that has been drawn upon in various papers for evaluating architectural models or styles. However, the problem with all these methods may involve the long distance and parallelism of the attributes. For example, the correlation coefficient between DM attributes and a candidate style may be around 1, but each of DM attributes can be 100 times of the style's attributes [6, 7, 16, 28].

Fiondella and Gokhale [19] used Uncertainties in model parameters for importance assessment of a software system. Using methods based on the investigation of a model can also guide us in selecting architecture styles. In order to reach that goal, the optimal model is investigated for each style and the best one is selected. Thus, we can find a very large transfer matrix for each style. Jegourel *et al*. [25] proposed an algorithm based on a statistical estimation method for preventing the instability in the transfer matrix and ultimately reducing its size. In addition, SAT-based learning model has been proposed as a preventive method by Ivančić *et al*. [22] in order to avoid abnormalities. These methods also reduce the size of the transfer matrix. Thus, there are various algorithms for selecting an appropriate architecture style. There are various ways to check and improve the quality of software. For example, formal methods have become the recommended methodology in critical software engineering. In formal confirmation, a system must be identified with a specific formula such as Petri Net networks, automata, and process algebras that require formal expertise and may be complicated especially with large systems. Mkaouar *et al*. [32] proposed a model for a real-time work model using the Linear No-Threshold model (LNT) language, describing how to use it to integrate a formal confirmation phase into an Architecture Analysis and Design Language (AADL) - based development process. It can be compared with the proposed algorithm. The reason for comparing both of these methods is the use of statistical parameters.

Each architecture analysis and selection method contain both advantages and disadvantages that is reflected briefly in Table 1. Since each method does have a known algorithm structure, thus advantages and disadvantages of each are recognizable.

Table 1. Related works.

Structure of Algorithm	Presented by	Disadvantages	Advantages
AHP-based	Chun *et al*. [15] Kim *et al*. [27] Jabali *et al*. [23]	High time complexity High Sensitivity to criteria list High memory needs	Easy to design and apply
Matrix-based	Bosch and Bengtsson [12]		
Modelling and Formal Method	Fiondella and Gokhale [19] Jegourelc *et al*. [25] Ivančić *et al*. [22] Mkaouar *et al*. [32]	Costly in designing stage	Low time complexity

Considering behaviour and structure of each method based on Table 1. Those with AHP-based and Matrix-based structures have higher time complexity and memory needs and are dependent on the input NFRs, but their design and usage are easier. But if the modelling and formal methods used for analysis and selection of architecture, albeit their complex and high cost, time complexity order will reduce. Therefore, each of the different structures assumes known advantages and disadvantages.

3. Newly Proposed Method

As mentioned earlier, this research aims to select a software architecture style out of candidate styles using Gamma probability function. The Gamma probability function will be briefly described along with input attributes, with the probability function being first fitted based on some expert opinions on the quality attributes of each architecture style.

Then, for each style and given the stakeholders' qualitative requirement, the percentile corresponding to the experts' opinion is obtained and its chart is given. At the end, the style with a lower percentile chart is chosen. The lower this percentile, the cheaper and partly faster it will be. Evidently, we can achieve the stakeholders' qualitative level by employing an average programming team that reach the minimum levels of the selected style.

Figure 1 briefly displays the steps of implementing the appropriate architecture style selection.

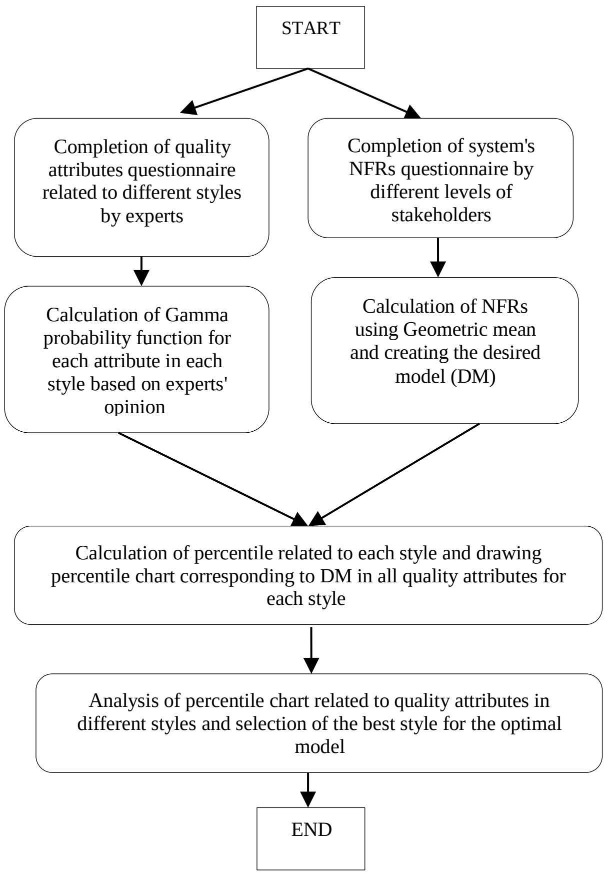
START
Completion of quality attributes questionnaire related to different styles by experts
Completion of system's NFRs questionnaire by different levels of stakeholders
Calculation of Gamma probability function for each attribute in each style based on experts' opinion
Calculation of NFRs using Geometric mean and creating the desired model (DM)
Calculation of percentile related to each style and drawing percentile chart corresponding to DM in all quality attributes for each style
Analysis of percentile chart related to quality attributes in different styles and selection of the best style for the optimal model
END

Figure 1. Schematic diagram of the proposed algorithm for selecting software architecture style.

- *Step* 1: At first, a questionnaire concerning the stakeholders' needs will be prepared, with the stakeholders in different levels giving each requirement a value from 1 to 8 according to Table 2.

Table 2. Questionnaire of Desired Model (DM).

Non-functional requirement	**Negative**	**Relatively negative**	**No effect**	**Low**	**Medium**	**High**	**Relatively high**	**Very high**
	1	2	3	4	5	6	7	8
Performance								
	⋮		⋮		⋮			
Reliability								

- *Step* 2: At this step, the stakeholders are given a weight for their every requirement based on their field of activity in regard to each NFR. To obtain the desired model, all stakeholders' requirements should be taken into account. Each quality attribute is of different importance from the stakeholders' point of view. For instance, a quality attribute like security, from one stakeholders' view, may be of high importance, preoccupying his/her mind, while from another one's view, it may be of no or little importance. Thus, software components can give rise to several kinds of architectural mismatches when assembled together in order to form a software system [9]. Quality attributes can be converted into quantitative ones using a variety of methods. One of these methods is the interval scale which has been used in [4] and is shown in Table 3.

To develop the desired model, qualitative needs are calculated for each stakeholder using the above-mentioned method. We then obtain the numerical average, which is the attribute's value in the Desired Model (DM). As mentioned earlier in this paper, in order to design a software system for a big organization, a questionnaire based on the quality attributes in Table 2 was filled in by the members of this company. The DM was calculated after averaging. It is also possible to give a certain weight to each attribute's value of importance from the personnel's viewpoint based on their field of activity; however, giving weight has not been considered for this example. After the questionnaires are received, the Geometric mean for each attribute in each style is used. Finally, the obtained value of each attribute is rounded to 1.0, which is seen in the estimated DM (Table 3).

Table 3. Questionnaire of Desired Model (DM).

NFR	Performance	Security	Modification	Reusability	Scalable	Portability	Reliability
DM	8	5	7	8	6	4	6

- *Step* 3: In order to quantitatively calculate the NFRs of the candidate styles, a scale-interval-based questionnaire is used, with 10 chosen software engineers having been provided with a table for the candidate styles like Table 4. After the questionnaires are received, the Geometric mean is used for each NFR in each style. At the end, the obtained value of each attribute is rounded to 1.0, the value of the quality attributes for the existing styles being observed in Table 4.

Table 4. Questionnaire for NFRs-Styles strength assessment.

Non-functional requirement	Negative	Relatively negative	No effect	Low	Medium	High	Relatively high	Very high
	1	2	3	4	5	6	7	8
Performance								
	⋮		⋮		⋮			
Reliability								

- *Step* 4: At this step, a Gamma statistical function fitting is calculated for each attribute of each style based on experts' opinions.
- *Step* 5: After the Gamma function is calculated, the percentile corresponding to the given quality attribute in the DM can be calculated and extracted.
- *Step* 6: At the last step of this algorithm, the appropriate software architecture style satisfying the stakeholders' requirements with minimum cost is selected after the results and the chart are examined.

In addition to giving a brief description of the Gamma statistical function, the steps of implementing the algorithm will be explained in detail.

4. Gamma Distribution

As mentioned previously, after the DM is calculated and the questionnaire is completed by the experts, the fitting for each style should be carried out through the Gamma probability distribution function [45]. The Gamma probability distribution function is one of the continuous probability distributions used in the problems related to optimization, with a shape parameter α, a scale parameter β and a Gamma function $\Gamma_{(\alpha)}$ obtained using Equation (1). If α is a natural number, then the Gamma distribution is equal to the sum of the random variable α with the exponential distribution based on parameter 1/β.

$$\Gamma_{(\alpha)} = \int_0^\infty t^{\alpha-1} e^{-m}\, dm \tag{1}$$

However, in order to increase the accuracy of the results, the 3-parameter Gamma distribution for the starting point was used according to Equation (2), with the Gamma probability distribution function being fitted to each quality attribute of each style (e.g., reliability in OO style) based on Equation (2) in the fourth step of the algorithm. i.e., the calculation of the Gamma probability distribution function. It should be noted that since the Gamma distribution covers only the positively skewed data, an appropriate change of variable has been used to estimate the probability function, the attributes of which have been shown in Table 5.

Table 5. Input attributes of 3-parameter Gamma distribution for the starting point.

Input attribute	Description
X	Qualitative variable of the case study
X_{max}	Reported maximum value in the case study
X_{min}	Reported minimum value in the case study
Delta	Positively skewed data are equal to the reported minimum distance to the drawing point of Gamma function and negatively skewed data are equal to the reported maximum distance to the drawing point of data
α	Shape parameter of Gamma probability density function
β	Scale parameter of Gamma probability density function
S	Skewness of observational data

$$PDF_{Gamma(\mathrm{X};\alpha,\beta,X_{min},X_{max},\mathrm{Delta},\mathrm{S})} = \begin{cases} \dfrac{\beta^{\alpha}(X-X_{min}+Delta)^{(\alpha-1)}e^{-\beta(X-X_{min}+Delta)}}{\Gamma_{(\alpha)}}; \\ S \geq 0\ and\ X > (X_{min} - Delta) \\ \dfrac{\beta^{\alpha}(X_{max}+Delta-X)^{(\alpha-1)}e^{-\beta(X-X_{min}+Delta)}}{\Gamma_{(\alpha)}}; \\ S < 0 and\ X > (X_{max} + Delta) \\ 0\,;\ otherwise \end{cases} \tag{2}$$

$$S = E\left[\left(\frac{X-\mu}{\sigma}\right)^3\right] = \frac{\frac{\sum_{i=1}^{N}(X_i{}^3)}{N} - 3\mu\sigma^2 - \mu^3}{\sigma^3},$$
$$\mu = \frac{\sum_{i=1}^{N} X_i}{N}, \sigma = \sqrt{\frac{\sum_{i=1}^{N}(X_i - \mu)^2}{N-1}}$$

In addition, the cumulative probability function for the function above is as follows:

$$CDF_{Gamma(\mathrm{X};\alpha,\beta,X_{min},X_{max},\mathrm{Delta},\mathrm{S})} = \int_{-\infty}^{X} PDF_{Gamma(\mathrm{X};\alpha,\beta,X_{min},X_{max},\mathrm{Delta},\mathrm{S})}\,dx \tag{3}$$

And attributes are estimated as follows:

$$\mathrm{Y} = \begin{cases} \mathrm{X}\text{-}(\mathrm{X}_{\mathrm{min}}\text{-Delta}); \mathrm{S} \geq 0 \\ (\mathrm{X}_{\mathrm{max}} + \mathrm{Delta})\text{-}\mathrm{X}\,; \mathrm{S} < 0 \end{cases} \tag{4}$$

$$A = Ln(\overline{X}) - \overline{Ln(X)} = LN\left(\frac{\sum_1^N X_i}{N}\right) - \left(\frac{\sum_1^N (Ln(X_i))}{N}\right);$$
$$\hat{\alpha} = \frac{1}{4A}\left(1 + \sqrt{1 + \frac{4A}{3}}\right); \hat{\beta} = \frac{\overline{X}}{\hat{\alpha}};$$

$$\mathrm{Delta} = \mathrm{Inverse\_Gamma}\left(\frac{1}{N+1}; \hat{\alpha}, \hat{\beta}\right) \tag{5}$$

Equation (5) is a recursive one used for calculating Delta, by which new values of $\hat{\alpha}, \hat{\beta}$ and Delta can be regarded as attributes of the intended variable probability function with approximation for one recursive calculation, where *N* indicates the number of questionnaires completed by experts. In order to calculate Inverse-Gamma function, the estimation using numerical calculations methods is used.

In the fifth step of the algorithm, given the probability function for each quality attribute related to each programming style, the percentile corresponding to the expected qualitative level in the DM of each quality attribute (called Z) can be obtained for the probability function of each style and quality attribute using Equation (6).

$$\mathrm{Percentile}_{(\mathrm{z};\alpha,\beta,\mathrm{X}_{\mathrm{min}},\mathrm{X}_{\mathrm{max}},\mathrm{Delta},\mathrm{S})} = 100 * \begin{cases} 1\,;\mathrm{S}<0, z>(\mathrm{X}_{\mathrm{max}}+\mathrm{Delta}) \\ 0;\mathrm{S}\geq 0,\mathrm{z}<(\mathrm{X}_{\mathrm{min}}-\mathrm{Delta}) \\ \mathrm{CDF}_{\mathrm{Gamma}(\mathrm{z}-\mathrm{X}_{\mathrm{min}}+\mathrm{Delta};\alpha,\beta,\mathrm{X}_{\mathrm{min}},\mathrm{X}_{\mathrm{max}},\mathrm{Delta},\mathrm{S})}; \\ \mathrm{S}\geq 0,\mathrm{z}\geq(\mathrm{X}_{\mathrm{min}}-\mathrm{Delta}) \\ 1-\mathrm{CDF}_{\mathrm{Gamma}(\mathrm{X}_{\mathrm{max}}+\mathrm{Delta}-\mathrm{z};\alpha,\beta,\mathrm{X}_{\mathrm{min}},\mathrm{X}_{\mathrm{max}},\mathrm{Delta},\mathrm{S})}; \\ \mathrm{z}\leq(\mathrm{X}_{\mathrm{max}}+\mathrm{Delta}) \end{cases} \tag{6}$$

It should be noted, however, it is likely that the desired qualitative level in the DM in one or several NFRs are not fulfilled by the highest level of a similarly intended quality attribute in one programming style. In this case, the number 100 is reported for the corresponding percentile. Moreover, if the desired qualitative level in the DM in one or several NFRs can be fulfilled for the lowest level of a similarly intended quality attribute in one programming style, the number 0 is reported for the corresponding percentile.

5. Analysis of Exam Results

This research firstly carried out probability function fitting on the experts' opinions about the quality attributes of each architecture style. Then, the percentiles corresponding to each DM were calculated based on the probability functions obtained from those opinions (the 20th percentile of a software quality attribute means this attribute in a certain software style has a low quality; this hypothetical software has a low reliability in comparison to other object-oriented programs). Now, if it is assumed that the experts have given the reliability of the object-oriented style a score of 8 from 6 to 8, it can be assumed that the case study has obtained a score of 6.2 with a low quality in terms of reliability in the hypothetical object-oriented style. Now, with a special mixture of the percentiles of software styles quality attributes, we want to satisfy users' need to their desired software qualitative levels DM. It should be noted that the lower the percentile number corresponding to each quality attribute in a style, the lower the costs of producing software program with this style will necessarily be. In other words, if a style automatically ensures the high quality of an attribute, a certain level of quality in the DM, with a lower cost (or using an average programming team with a lower wage); percentiles lower than a quality attribute, can be fulfilled. At this step, the chart of the calculated percentiles for the DM in each style has been shown in Figure 2.

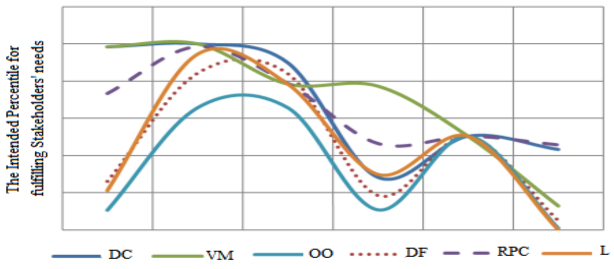

Figure 2. The Intended Percentile for fulfilling stakeholders' needs.

Data Flow and Layered styles rank second and RPC and VM styles achieve the lowest corresponding to the DM's expected NFRs. To test the validity of the proposed algorithm (Gamma distribution), Case Study is evaluated using the LiVASAE Technique algorithm and the results are shown in Table 6.

Table 6. Input attributes calculate *ROI* of styles according to the LiVASAE technique.

NFR	DM	DC	DF	VM	RPC	OO	L
Modification	**7**	**5.25**	**7.25**	**4.875**	**5**	**7.625**	**7.375**
Reusability	**8**	**5.375**	**7.375**	**5.25**	**5.75**	**7.875**	**6**
Performance	**8**	**7.125**	**6.875**	**7.5**	**7.625**	**7.625**	**7.5**
Scalability	**6**	**6.375**	**6.875**	**5**	**5.25**	**7.25**	**6.375**
Reliability	**6**	**7.5**	**6.25**	**5.875**	**5.875**	**7.625**	**7.125**
Portability	**4**	**5.125**	**5.5**	**6.25**	**5.375**	**7.375**	**6.5**
Security	**5**	**7**	**6.375**	**7.125**	**6.5**	**6.875**	**7.375**
$\prod_{i=1}^{n} NFR_i$	**322560**	**344870.5**	**553819.6**	**251094.9**	**236230.9**	**1283343**	**722628.3**
$ROI = \frac{\prod_{i=1}^{n} NFR_i}{\prod_{i=1}^{n} NFR_i\,(DM)}$	**-----**	**1.069167**	**1.716951**	**0.778444**	**0.732363**	**3.978619**	**2.240291**

In Table 6, according to the ROI, the LiVASAE Technique considers the OO style as the best choice to meet the needs of the stakeholders. Table 6 and Figure 2 show a comprehensive comparison between three algorithms based on the response rate of styles for the case study. Kim *et al*. [27] used the Arithmetic mean method to calculate the NFRs of different architectures (styles), Jabali *et al*. [23] used ordinary AHP algorithm while we used Gamma distribution method to smooth out the NFRs of each style. According to Table 7, the results show the validity of the proposed algorithm, with the exception that the proposed algorithm is better seen by using the probability distribution functions (uncertainty) for each of the NFRs.

Table 7. Comparison of the results from LiVASAE technique, ordinary AHP and proposed gamma method.

Style	**LiVASAE Technique Rank**	**Ordinary AHP Rank**	**Gamma distribution Rank**
OO	1	1	1
L	2	2	2
DF	3	4	3
DC	4	3	4
VM	5	6	6
RPC	6	5	5

Furthermore, the corresponding time complexity of the methods is shown and compared with each other according to Figure 3.

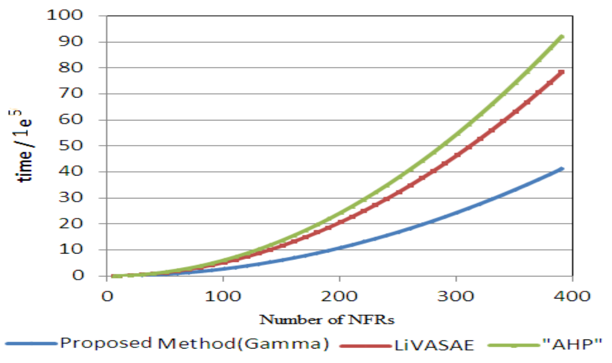

Figure 3. Time complexity of evaluated methods.

Considering the Figure 3, we can say that increasing number of NFRs has the potential to improve comparison of methods with each other. By different number of NFRs, we can execute each algorithm and based on the results, one can say that the AHP (proposed method) has the highest (lowest) time complexity in selection of software architecture style.

6. Conclusions

This paper examined a certain mixture of optimal quality attributes in a relatively organization employing 390 people as the case study is selected. Six quality attributes by the architect (modifiability, reusability, performance, scalability, reliability and portability) in six software styles (DC, DF, VM, RPC, OO, and L) were investigated and probability functions were fitted on them. The percentiles corresponding to each quality attribute in the DM were calculated. Using the graphical visual comparison and drawing the points on the chart, the styles with the lowest percentiles (corresponding to the lowest implementation cost) were able to satisfy users' non-functional needs. The rankings of styles were found to be respectively as follows: OO, L, DF, DC, RPC, and VM.

The results show that OO-developed software is able to satisfy users' needs with the smallest percentile corresponding to its quality attributes. This means that an ordinary object-oriented program, with a medium or even weak level of quality, e.g. one written by a semi-skilled programmer, is as efficient in satisfying users' needs as a layered-based program developed by a skilled programmer (having spent more time and money on it). In other words, a certain combination of hypothetical user's DM quality attributes may be fulfilled by a sub-optimal OO-style program with minimal cost, instead of a costly one developed in layered style, which is not optimal regarding user's DM.

Various papers have suggested novel strategies to implement OO-style, e.g., Librecherr and Xiao [31] extended the object-oriented programming paradigm to a structure called adaptive programming. The Object-Oriented style was tested on some case study [11].

It is of prime importance to survey the DM before selection of style architecture, to select the most applicable one. The suggested algorithm has the following advantages and disadvantages:

1. 1st advantage: since this method reorders candidates based on their correspondence with the DM, thus it can be a tool in selection of a set of styles in designing a multi-morphologic architecture. In other words, similar styles (ranking in reordering) come in the same set.
2. 2nd advantage: the method needs low memory and less time complexity. In fact, in contradiction with the Modelling, AHP and Matrix based methods, in this method applying with the variation in order and number of NFRs and also increased number of styles and stakeholders, the subject space does not change much. The Figure 3 in the previous section, shows the time complexity against other methods and verify this advantage.
3. 3rd advantage: it's design is easy, in a way that similar to AHP and Matrix based methods, it can be executed with a few mathematical and statistical relations and does not have complexity and design as in the modelling method.

Main disadvantage: as other methods, the proposed method is not free from disadvantages. The main disadvantage is related to the time of study and recognition of the subject. In other words, if a parameter in the analysis time of the system and architecture design is overlooked, at the end all design computations must be repeated from their beginning. Since this method is a selected one, with the changes in the DMs, the measure for selection differs. So, the comprehension of the subject and assessment of the stakeholders' considered NFRs should be done with most care and delicacy.

References

[1] Abowd G., Allen R., and Garlan D., "Formalizing Style to Understand Descriptions of Software Architecture," *ACM Transactions on Software Engineering and Methodology*, vol. 4, no. 4, pp. 319-364, 1995.

[2] Admodisastro N. and Palaniappan S., "A Code Generator Tool for the Gamma Design Patterns," *Malaysian Journal of Computer Science*, vol. 15, no. 2, pp. 94-101, 2002.

[3] Affleck A., Krishna A., and Achuthan N., "Non-Functional Requirements Framework: A Mathematical Programming Approach," *The Computer Journal*, vol. 58, no. 5, pp. 1122-1139, 2015.

[4] Albin S., *The Art of Software Architecture: Design Methods and Techniques*, John Wiley and Sons, 2003.

[5] Arcelli D., Cortellessa V., and Di Pompeo D., "Performance-Driven Software Model Refactoring," *Information and Software Technology*, vol. 95, pp. 366-397, 2018.

[6] Astudillo H., "Five Ontological Levels to Describe and Evaluate Software Architecture," *Revista Facultad de Ingeniería-Universidad de Tarapacá*, vol. 13, no. 1, pp. 69-76, 2005.

[7] Babamir S. and Khabazian M., "Evaluation of Qualitative Requirement Analysis in Software Architecture," *in Proceedings of the International Conference of IT Knowledge*, Mashhad, 2007.

[8] Bedi P. and Gaur V., "Trust Based Prioritization of Quality Attributes," *The International Arab Journal of Information Technology*, vol. 5, no. 3, pp. 223-229, 2008.

[9] Bernardo M., Ciancarini P., and Donatiello L., "Architecting Families of Software Systems with Process Algebras," *ACM Transactions on Software Engineering and Methodology*, vol. 11, no. 4, pp. 386-426, 2002.

[10] Béjar R., Latre M., Nogueras-Iso J., Muro-Medrano P., and Zarazaga F., "An Architectural Style for Spatial Data Infrastructures," *International Journal of Geographical Information Science*, vol. 23, no. 3, pp. 271-294, 2010.

[11] Binder R., "Testing Object Oriented Software: A Survey," *Software Testing, Verification and Reliability*, vol. 6, no. 34, pp. 125-252, 1996.

[12] Bosch J. and Bengtsson P., "Assessing Optimal Software Architecture Maintainability," *in Proceedings of 5th European Conference on Software Maintenance and Reengineering*, Lisbon, pp. 168-175, 2001.

[13] Busch A., Fuch D., and Koziolek A., "PerOpteryx: Automated Improvement of Software Architectures," *in Proceedings of IEEE International Conference on Software Architecture Companion*, Hamburg, pp. 162-165, 2019.

[14] Castelli V., Harper R., Heidelberger P., HunterS., Trivedi K., Vaidyanathan K., and Zeggert W., "Proactive Management of Software Aging," *IBM Journal of Research and Development*, vol. 45, no. 2, pp. 311-332, 2001.

[15] Chong C., Lee P., and Ling C., "Prioritizing and Fulfilling Quality Attributes for Virtual Lab Development Through Application of Fuzzy Analytic Hierarchy Process and Software Development Guidelines," *Malaysian Journal of Computer Science*, vol. 27, no. 1, pp. 1-19, 2014.

[16] Clements P., Bass L., Garlan D., Ivers J., Little R., Nord R., and Stafford J., *Documenting Software Architectures*, Addison Wesley, 2007.

[17] Ebad S., "Towards Measuring Software Requirements Volatility: A Retrospective Analysis," *Malaysian Journal of Computer Science*, vol. 30, no. 2, pp. 99-116, 2017.

[18] Fieberg J. and Conn P., "A hidden Markov Model to Identify and Adjust for Selection Bias: An Example Involving Mixed Migration Strategies," *Ecology and Evolution*, vol. 4, no. 10, pp. 1903-1912, 2014.

[19] Fiondella L. and Gokhale S., "Importance Measures for Modular Software with Uncertain Parameters," *Software Testing, Verification and Reliability*, vol. 20, no. 1, pp. 63-85, 2009.

[20] Garlan D. and Shaw M., *Advances in Software Engineering and Knowledge Engineering*, World Scientific Publishing Company, 1994.

[21] Hadizadeh A. and Tanghatari E., "Parallel Processor Architecture with a New Algorithm for Simultaneous Processing of MIPS-Based Series Instructions," *Emerging Science Journal*, vol. 1, no. 4, pp. 226-232, 2018.

[22] Ivančić F., Yang Z., Ganai M., Gupta A., and Ashar P., "Efficient SAT-Based Bounded Model Checking for Software Verification," *Theoretical Computer Science*, vol. 404, no. 3, pp. 256-274, 2008.

[23] Jabali F., Sharafi S., and Zamanifar K., "A Quantitative Algorithm to Select Software Architecture by Trade off between Quality Attributes," *Procedia Computer Science*, no. 3, pp. 1480-1484, 2011.

[24] Jahanirad H. and Karam H., "BIST-based Testing and Diagnosis of LUTs in SRAM-based FPGAs," *Italian Journal of Science and Engineering*, vol. 1, no. 4, pp. 216-225, 2017.

[25] Jegourel C., Legay A., and Sedwards S., "Command-Based Importance Sampling for Statistical Model Checking," *Theoretical Computer Science*, vol. 649, pp. 1-24, 2016.

[26] Kalistratov D., "Wireless Video Monitoring of the Megacities Transport Infrastructure," *Civil Engineering Journal*, vol. 5, no. 5, pp. 1033-1040, 2019.

[27] Kim C., Lee D., Ko I., and Baik J., "ALightweight Value-based Software Architecture Evaluation. Eighth," *in Proceedings of the International Conference on Software Engineering Artificial Intelligence, Networking, and Parallel Distributed Computing*, Qingdao, pp. 646-649, 2007.

[28] Kim J. and Garlan D., "Analyzing Architectural Styles with Alloy," *in Proceedings of Workshop on the Role of Software Architecture for Testing and Analysis*, Portland, pp. 70-80, 2006.

[29] Lefranc G., "NETE Review of Architectural Patterns and Tactics for Micro Services in Academic and Industrial Literature," *IEEE Latin America Transactions*, vol. 16, no. 9, pp. 2321-2327, 2018.

[30] Lian Y., Tang Y., and Wang Y., "Objective Bayesian Analysis of JM Model in Software Reliability," *Computational Statistics and Data Analysis*, vol. 109, pp. 199-214, 2017.

[31] Lieberherr K. and Xiao C., "Customizing Adaptive Software to Object-Oriented Software Using Grammars," *International Journal of Foundations of Computer Science*, vol. 5, no. 2, pp. 179-208, 1994.

[32] Mkaouar H., Zalila B., Hugues J., and Jmaiel M., "A Formal Approach to AADL Model-Based," *International Journal on Software Tools for Technology Transfer*, vol. 22, no. 2, pp. 1-29, 2019.

[33] Pérez J., Ramos I., Carsí J., and Costa-Soria C., "Model-Driven Development of Aspect-Oriented Software Architectures," *Journal of Universal Computer Science*, vol. 19, no. 10, pp. 1433-1473, 2013.

[34] Perry D. and Wolf A., "Foundations for the Study of Software Architectures," *ACM Software Engineering Notes*, vol. 17, no. 4, pp. 40-52, 1999.

[35] Phillips D., Mazzuchi T., and Sarkani S., "An Architecture System Engineering and Acquisition Approach for Space System Software Resiliency," *Information and Software Technology*, vol. 94, pp. 150-164, 2018.

[36] Pinto M. and Fuentes L., "Modelling Quality Attributes with Aspect-Oriented Architectural Templates," *Journal of Universal Computer Science*, vol. 17, no. 5, pp. 639-669, 2011.

[37] Quaglia F., "Software Diversity-Based Active Replication as An Approach for Enhancing the Performance of Advanced Simulation Systems," *International Journal of Foundations of Computer Science*, vol. 18, no. 3, pp. 495-515, 2007.

[38] Ravishanker N., Liu Z., and Ray B., "NHPP Models with Markov Switching for Software Reliability," *Computational Statistics and Data Analysis*, vol. 52, no. 8, pp. 3988-3999, 2008.

[39] Ribeiro S., Schmitz E., Alencar A., and Silva M., "Literature Review on the Theory of Constraints Applied in the Software Development Process," *IEEE Latin America Transactions*, vol. 16, no. 11, pp. 2747-2756, 2018.

[40] Salama M. and Bahsoon R., "Analyzing And Modelling Runtime Architectural Stability for Self-Adaptive Software," *Journal of Systems and Software*, vol. 133, pp. 95-112, 2017.

[41] Schaefer L., Rabiser R., Clarke D., Bettini L., Benavides D., Botterweck G., Pathak A., Trujillo S., and Villela K., "Software Diversity: State of the Art and Perspectives," *International Journal on Software Tools for Technology Transfer*, vol. 14, no. 5, pp. 477-495, 2012.

[42] Sharma A., Kumar M., and Agarwal S., "A Complete Survey on Software Architectural Styles and Patterns," *The International Conference on Eco-Friendly Computing and Communication Systems*, vol. 70, pp. 16-25, 2015.

[43] Sharma T. and Spinellis D., "A Survey on Software Smells," *Journal of Systems and Software*, vol. 138, pp. 158-173, 2018.

[44] Sievi-Korte O., Koskimies K., Mäkinen E., "Techniques for Genetic Software Architecture Design," *The Computer Journal*, vol. 58, no. 11, pp. 3141-3170, 2015.

[45] Venters C., Capilla R., Betz S., Penzenstadler B., Crick T., Crouch S., Nakagawa E., Becker C., and Carrillo C., "Software Sustainability: Research and Practice from A Software Architecture Viewpoint," *Journal of Systems and Software*, vol. 138, pp. 174-188, 2018.

[46] Wang Q. and Yang Zh., A Method of Selecting Appropriate Software Architecture Styles, Quality Attributes and Analytic Hierarchy Process. University of Gothenburg, 2012.

[47] Woodward E., Bowers R., Thio V., Johnson K., Srihari M., and Bracht C., "Agile Methods for Software Practice Transformation," *IBM Journal of Research and Development*, vol. 54, no. 2, pp. 3-12, 2010.

Hamidreza Hasannejad Marzooni received B.S. degree in Mazandaran University of Science and Technology and M.S. degree in Computer Engineering- Software Engineering from Sari Islamic Azad University 2011 and 2013. Respectively. He is studying Computer Engineering-Software Engineering in Babol Islamic Azad University. His current research interests include Software Architecture Styles, Numerical Analysis, and Majority Voting.

Homayun Motameni serves as an Professor at the Computer Department, Sari Branch, Islamic Azad University. He obtained his PhD in Computer Engineering-Software Engineering in the Department of Computer at the Islamic Azad University (Sciences and Research Branch), Tehran, Iran. He received his M.S. degree in Computer Engineering-Machine Intelligence from Islamic Azad University-Science and Research Branch and BS from the Shahid Beheshti of Tehran University in Computer Engineering-Software Engineering, Iran. He is on the editorial board of the Journal of Soft Computing and Information Technology (JSCIT) and Director-in-Charge of the journal of advances in computer research (JACR). His research interests include Software Engineering, https://scholar.google.com/citations?view_op=search_authors&hl=en&mauthors=label:formal_methods_petri _netModel Checking, Requirements Engineering, performance evaluation.

A**li Ebrahimnejad** serves as an Associate Professor at the Mathematics Department, Qaemshahr Branch, Islamic Azad University. He obtained his PhD in Applied Mathematics in the Department of Mathematics at the Islamic Azad University (Sciences and Research Branch), Tehran, Iran. He received his BS from the Mazandaran University, Iran. He is on the editorial board of the International Journal of Fuzzy System Applications (IJFSA), Annals of Fuzzy Mathematics and Informatics (AFMI), International Journal of Information and Decision Sciences (IJIDS), Iranian Journal of Optimization (IJO), International Journal of Strategic Decision Sciences (IJSDS) and International Journal of Enterprise information Systems (IJEIS). His research interests include operations research, network flow, data envelopment analysis and fuzzy optimization.