Journal of Advances in Computer Research
Quarterly ISSN: 2008-6148
Sari Branch, Islamic Azad University, Sari, I.R.Iran
(Vol. 11, No. 1, February 2020), Pages: 101-111
www.jacr.iausari.ac.ir

Selecting an Architecture Style Using Fuzzy Cubic Spline on in Style-based Systems

Hamidreza Hasannejad Marzooni[1], Homayun Motameni✉[2], 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
Hamidreza.hasannejad.m@gmail.com; motameni@iausari.ac.ir; a.ebrahimnejad@qaemiau.ac.ir

Received: 2019/08/29; Accepted: 2020/09/16

Abstract

An architecture shows to what extent a system meets the needs of the stakeholders, so designing a desired architecture produces very high-quality software tailored to the needs of the stakeholders. Thus, finding a software architecture model is quite critical, and to achieve this goal, a suitable architecture style model must be used. In this regard, attempts are made to deploy the behavior of the candidate software styles as well as the stakeholders' desired model as functions graphs, so that the desired architecture style model graphs, which meet the maximum needs of the stakeholders, are selected by comparison. Fuzzy Cubic Spline is used in the structure of the proposed algorithm, which requires turning qualitative data into quantitative data. To evaluate the proposed approach, a controlled experiment is also conducted. The proposed approach is compared with an analytic hierarchy process-based approach (AHP), TOPSIS and PTFG (Prioritizing using tensor and fuzzy graphs) in the experiment. The results analysis demonstrates that our approach needs to less time complexity and is much easier to use and highly accurate.

Keywords: *Software Architecture Style, Non-Functional Requirements, Fuzzy Curve Fitting, Fuzzy Spline*

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 [1], which should be incorporated throughout

the life cycle of producing a software program including analysis, architecture development, design, implementation, verification and maintenance phases [2]. 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 [3].

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 [4]. 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. The most important factor in ensuring the quality of software program is its architectural sustainability [5]; 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. used a formal linear programming optimization model for the NFRs framework with regard to the choice of operation [6]. Decisions about software architecture depend on system failures. 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.

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 [7]. 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 study was conducted in attempt to present a style-based software architecture model. In order to achieve that objective, we explored how to select the style of financial software architecture in a relatively small company as a case study involving three stakeholders: manager, computer engineer and accountant. In this case study, the appropriate software architecture style was facilitated through the proposed algorithm from among Remote Procedure Call (RPC), Data Flow (DF), Data Center (DC), Virtual Machine (VM), Object Oriented (OO) and Layered (L) styles. A few of the qualitative parameters of such styles crucial in this system have been obtained through questionnaire estimation completed by experts. This study proposed a new algorithm based on mathematical argument, from which the style with the greatest consistency was selected with the NFRs of the stakeholders. The main advantage of this method over its counterparts lies in its direct calculation independent of the number of candidate styles or NFRs. Therefore, the most robust results ever will be obtained. The paper has been arranged as follows. Each one of the studied methods are briefly described in

Section 3. The methods are implemented on the data of the studied problem simultaneously. The outcome and results of software architecture style selection by means of Majority Voting are presented. Furthermore, the algorithms are compared with each other in terms of the time required for running and the results obtained from running each one algorithm. The details will be described later in this paper.

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 in 1992 [8]. In 1994, Garlan and Shaw [9] 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 [10]. 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, Hoseini Jabali et al. [11] 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 et al. [12] also presented an algorithm based on AHP for style selection. Chun Yong Chong et al. [3] offered a fuzzy AHP-based algorithm in an effort to identify quality attributes and rank them based on their priorities. Kim et al. [13] 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 [14-17].

Fiondella et al. [18] 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. Cyrille Jegourel et al. [19] 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 Franjo Ivančić et al. [20] 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. [21] proposed a model for a real-time work model using the LNT language, describing how to use it to integrate a formal confirmation phase into an 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.

Misaghian et al. [22] used an algorithm abbreviated as PTFG and achieved the same degree of correction with less calculation demand as sophisticated Multi-Criteria Decision Making (MCDM) algorithm of technique of ordered preference by similarity to ideal solution (TOPSIS), which evaluates the alternatives based on cost (relative distance from the positive ideal alternative) and benefit (relative distance from the negative ideal alternative) [23]. Again, changing the number of architecture style s and/or the number of NFRs in TOPSIS and PTFR will oblige total recalculation and there would be no guarantee for achieving the same prioritization ranking, nor there would be results comparable against the case of lowest number of Styles/NFRs, just as mentioned in the case of AHP. The result of strength/weakness analysis of introduced alternative methods have been summarized in Table 1.

Table 1 -SWOT table of alternative algorithms

Algorithm	Related Works	Weakness	Strength
AHP-based	• Chun [3] • Kim [13] • Hoseini jabali [11]	• High time complexity • High Sensitivity to criteria list	• Easy to design and apply
Matrix-based	• Bosch [10]	• High memory need	
Modeling & Formal Method	• Jegourel [19] • Ivančić [20] • Mkaouar [21] • Misaghian[22]	• Costly in designing stage	• Low time complexity

This research presents a new algorithm based on mathematical reasoning and a curve fitting 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. It will be described in more details below.

This algorithm consists of two methods: cubic spline and fuzzy logic. In its description, we first need to briefly introduce the fuzzy cubic spline and the Delphi fitting method. Then, we will present the algorithm and calculate the results.

3. Proposed algorithm

The main purpose of this study is to select a software architecture style from among the candidates using the fuzzy cubic spline fitting method. The core of the newly proposed algorithm consists of two methods of cubic spline fitting and fuzzy logic. Therefore, we need to initially introduce cubic spline fitting and fuzzy Delphi prior to providing a description of the proposed algorithm.

3.1 Cubic spline

In curve fitting methods, higher-degree polynomials might be adopted to achieve more accurate results in problems. High-degree polynomials not only increase the number of computational operations, but also render the findings uncertain due to potential rounding errors. The piecewise technique is employed to keep the degree of interpolation polynomials low and to achieve the desired accuracy in approximation problems.

By partition of interval [a, b] into sub-intervals [x_{i-1}, x_i] and approximating the function through low-degree polynomials in each sub-interval, we can enhance the accuracy while preventing the oscillating nature of high-degree polynomials. One of these methods involves a three-point, equidistance, piecewise, linear interpolation known as cubic spline, which follows Equation (1).

$$S(x) = \frac{1}{6h}(x_i - x)[(x_i - x)^2 - h^2]M_{i-1} + \frac{1}{6h}(x - x_{i-1})[(x - x_{i-1})^2 - h^2]M_i + \frac{1}{h}[(x_i - x)f_{i-1} + (x - x_{i-1})f_i] \ , 1 \le i \le \text{n} \quad (1)$$

Each coefficient of (M_i) in Equation (1) is calculated through Equation (2).

$$M_{i-1} + 4M_i + M_{i+1} = \frac{6}{h^2}[f_{i+1} - 2f_i + f_{i-1}] \ , 1 \le i \le n-1$$

I. $M_0 = M_n = 0$

II. $M_0 = M_n \ , M_1 = M_{n+1} \ , f_0 = f_n \ , \ f_1 = f_{n+1}$

III. $2M_0 + M_1 = \frac{6}{h}\left[\frac{f_1 - f_0}{h} - f_0\right] \ , \ M_{n-1} + 2M_n = \frac{6}{h}\left[f_n - \frac{f_n - f_{n-1}}{h}\right]$ (2)

3.2 Fuzzy Delphi

Delphi is a robust process based on a group communication structure adopted in cases where incomplete, unreliable knowledge is available with the aim of achieving consensus among experts. In the classical Delphi method, the expert opinions are expressed in the form of definite numbers, whereas experts use their mental competencies to express opinions, indicating the possibility of uncertainty prevailing in this situation. The probability of uncertainty is compatible with fuzzy sets. Hence, it is better to obtain data in the form of natural language from experts and analyze it using fuzzy sets. The advantage of the fuzzy Delphi method is the integration of each opinion to reach a group agreement. Given the broad application and ease of calculation in the triangular method, the fuzzy Delphi calculation has been demonstrated in Equation (3).

$$aij = (cij, , gij)$$
$$cij = min(bijk) \ , k = 1, \dots . , n$$
$$dij = (\Pi \ ijk)^{\frac{1}{n}}, k = 1, \dots . , n$$
$$gij = max(bijk) \ , k = 1, \dots . , n \quad (3)$$

Equation (3) shows the relative importance of parameter *i* over parameter *j* from the viewpoint of *K*th expert, c as such, c*ij* and gij represent the lower and upper bounds of the respondent opinions and the geometric means of the respondents' opinions, respectively. In fact, this tool helps convert the expert opinions from the questionnaire into a triangular fuzzy number.

3.3 Overview of newly proposed algorithm

In this research, architecture styles will be employed to design software architecture. A suitable architecture is expected to fulfill the requirements of stakeholders to some extent. Therefore, selecting the appropriate style to achieve a good architecture can affect the quality of the software. The newly proposed algorithm consists of four steps and is able to select the appropriate style. This research intends to design a style-based architecture for financial software used in a small company. As previously mentioned, there are three stakeholders in this company with a very high degree of importance, as well as six architecture style candidates for seven non-functional requirements (quality attribute) to be examined. Figure 1 illustrates a schematic view of the new algorithm for designing a suitable architecture in order to maximize response to these qualitative attributes.

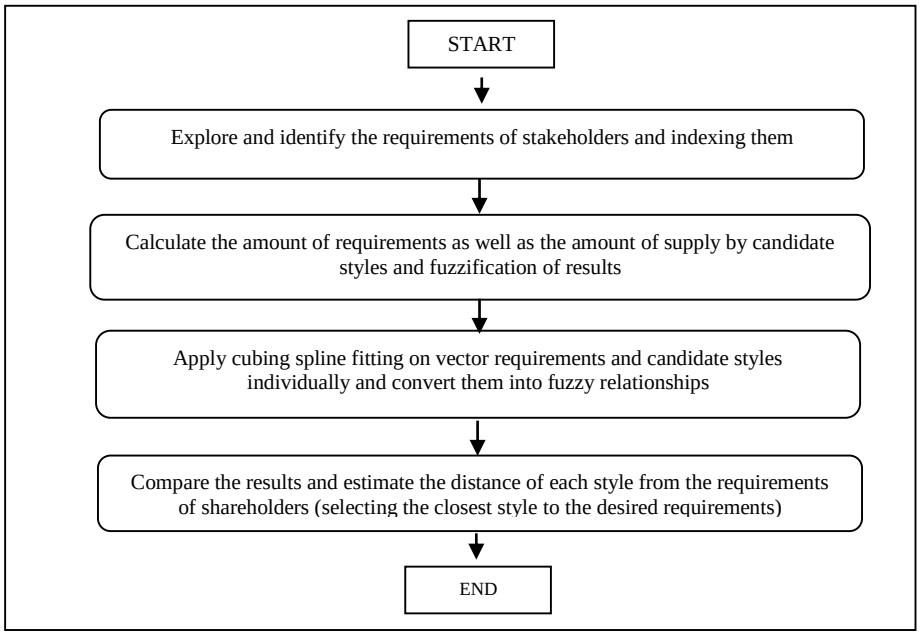

Figure 1. Software architecture style selection algorithm using fuzzy cubic spline

Step One: First, the non-functional requirement of stakeholders is specified. A fuzzy index will be assigned to each pair of more closely related requirements used as fuzzy x_is in the computational equations.

Step two: At this stage, according to the non-functional requirement, a questionnaire is prepared and presented to stakeholders to complete. There is also another questionnaire about candidate styles handed to software engineering experts to complete. After receiving the completed questionnaires and applying the fuzzy Delphi method, the questionnaires each of the requirements vector table as well as the styles table will be calculated fuzzily.

Step three: In this step, the cubic spline fitting is calculated for the candidate styles and the requirements vector. In this procedure, The x_is are defuzzified as the value of the non-functional requirement index while the $f(x_i)$s is its value (i.e. geometric mean of that requirement by experts in the styles and stakeholders in the DM). After calculating the cubic spline fitting in different partition intervals (between each of the two requirements) using fuzzy relationships, we will achieve a fuzzy cubic spline fitting. Table 2 displays an example of calculating a fuzzy spline fitting for a VM style.

Table 2. Calculate of Fuzzy Cubic Spline of VM

Calculation of the expected fuzzy value	Partition	Dependent parameter pair	Fuzzy index parameter pair $(\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3})$
$(y_1, y_2, y_3) = -.0509(x_1, x_2, x_3)^3 + 2.0512(x_1, x_2, x_3)^1 + 13.9225 = (-97.0289, -127.6089, -113.3484)$	p_0	(0,1,2)	Performance-Security
$(y_1, y_2, y_3) = .0833(x_1, x_2, x_3)^3 + .9700(x_1, x_2, x_3)^2 - 3.6624(x_1, x_2, x_3)^1 + 12.6158 = (-77.113, -105.2077, -138.8594)$	p_1	(2,3,4)	Security-Modification
$(y_1, y_2, y_3) = .09212(x_1, x_2, x_3)^3 - .0305(x_1, x_2, x_3)^2 + .0957(x_1, x_2, x_3)^1 + 8.15 = (256.2890, 313.628, 379.1967)$	p_2	(4,5,6)	Modification-Reusability
$(y_1, y_2, y_3) = .0707(x_1, x_2, x_3)^3 + .5222(x_1, x_2, x_3)^2 + 1.0791x_1, x_2, x_3{}^1 + 8.9564 = (320.4158, 381.2504, 449.4924)$	p_3	(6,7,8)	Reusability-Scalable
$(y_1, y_2, y_3) = -.1389(x_1, x_2, x_3)^3 - .322(x_1, x_2, x_3)^2 + .2995(x_1, x_2, x_3)^1 + 8.4644 = (-431.5962 - 528.2806, -638.11)$	p_4	(8,9,10)	Scalable-Portability
$(y_1, y_2, y_3) = -111(x_1, x_2, x_3)^3 + 1.3396(x_1, x_2, x_3)^2 - 3.7498(x_1, x_2, x_3)^1 + 10.9196 = (-83.6, -118.5424, -160.7956)$	p_5	(10,11,12)	Portability-Reliability

According to Table 2, we insert the value of each (x_1, x_2, x_3) in the fuzzy equation in that partition so as to obtain(y_1, y_2, y_3).

Step four: This is the last step of the algorithm. Thus, in both the DM fuzzy spline fitting and the fuzzy spline fitting, each of the candidate styles is obtained after the fuzzified x_is and $f(x_i)$ is placed. After calculating (y_1, y_2, y_3) in each partition, the value of distance between each of the candidate styles and DM is calculated using Equation (4) [24].

$$D_{2,\frac{1}{2}}(\tilde{A}, \tilde{B}) = \sqrt{\frac{1}{6}\left[\sum_{m=1}^{3}(b_m - a_m)^2 + (b_2 - a_2)^2 + \sum_{m\in\{1,2\}}(b_m - a_m)(b_{m+1} - a_{m+1})\right]} \quad (4)$$

Finally, the style with greatest similarity (least numerical distance) with DM is selected.

In fact, this distance is calculated in each partition and the mean of these distances in different partitions is the final distance of each style and DM as shown in Table 3.

Table 3. Distance between DM and each Style

Partition ↓ / Candidate Styles →	DF	VM	L	OO	RPC	DC
p_0	115.24	154.03	156.52	27.33	33.98	115.24
p_1	26.03	17.95	15.43	2.15	6.25	18.75
p_2	139.39	241.38	295.57	108.37	204.79	394.42
p_3	521.26	441.84	573.52	109.23	412.31	486.391
p_4	191.49	254.54	83.19	138.29	290.64	18.48
p_5	2.46	3.94	7.87	7.44	8.48	25.3
Distance between each style and the desirable model	165.9783	185.6133	188.6833	65.46833	159.4083	176.4302
Ranking of each style in fulfilling the requirements of stakeholders	3	5	6	1	2	4

According to Table 3, OO is the best style for the optimal model with the greatest responsiveness to the desired requirements of stakeholders. Moreover, RPC and DF are ranked second and third, respectively.

4. Analysis and review of results

In this research, we first examined and summarized the requirements of the stakeholders. Then, these requirements as well as the candidate styles are modeled using the cubic spline fitting method. In the end, each of these styles is compared against DM and the best style is selected. In order to evaluate the newly proposed algorithm and compare the output results against those of its counterparts from AHP, TOPSIS and PTFG algorithms, we used a computer equipped with an Intel Core i5 processor with 2.3 GHz of processing capacity, 500 GBs of Hard Disk, 4.0 GBs of RAM and an installed copy of Microsoft Windows 7 so as to evaluate and compare performance through MATLAB as the programming environment.

Table 2. The Priority ranking of selection of architecture styles by different algorithms

Algorithm	**Priority of Selection**	**Algorithm Mean Run Time (m seconds)**
AHP	OO,RPC,DF,DC,L,VM	2850.8
TOPSIS	RPC,OO,DF,DC,VM,L	1990.5
PTFG	OO,RPC,DC,DF,VM,L	1465.1
Proposed Algorithm	OO,RPC,DF,DC,VM,L	1212.9

The results are shown in Table 4, indicating the superiority of our proposed algorithm over all of its three counterparts in the sense of lower processing time requirement (in accordance with our prior expectation). Moreover, the results suggest that the priority ranking resulted from AHP and PTFG are identical. The reason behind the substitution of the first and second ranked architecture style s in TOPSIS can be the high similarity of these two styles (OO and RPC). Thus, the NFR weighting schemes need to be performed with special care, just as the case for the AHP algorithm. To compare the

proposed algorithm with the previous algorithms, the number of NFRs can be increased and the response time of each algorithm can be examined. The graphs have been displayed in Figure 2.

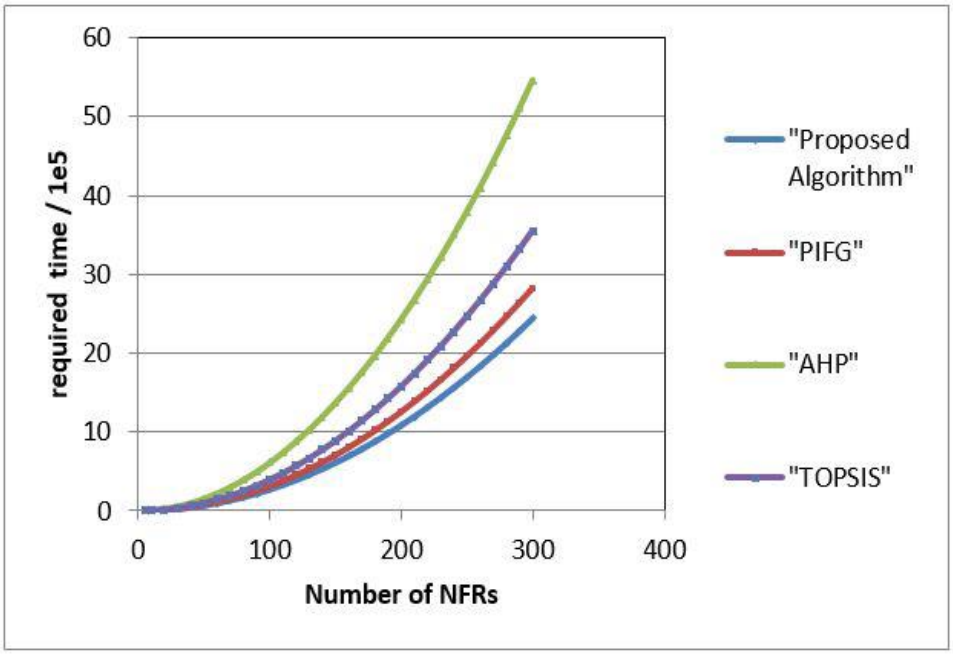

Figure 2. The required time for running each method according to the Number of NFRs tested

According to Figure 2 and Table 4, as well as the results, the proposed algorithm of this paper runs faster than other methods for small number of NFRs and the PIFG is ranked 2. Moreover, the results show that as the number of NFRs increases, the newly proposed algorithm will be the fastest while AHP will be the slowest method of all.

5. Conclusion and Future Works

This paper adopted an algorithm based on Fuzzy logic methods and Curve fitting together to select a software architecture style as a new step taken in software architecture. Based on Table 4 and Figure 2, we can conclude that: Compared to the other algorithms, the first and foremost advantage of the proposed algorithm comes from its lower time complexity. Secondly, it enables the selection of appropriate styles in a way that if new styles are suggested as the candidate, the new will be compared to the last selected style and the proposed algorithm selects the most appropriate one, while providing a superior qualification compared to the ASCC method. The last advantage is related to the size of the inputs. As the algorithm is an interpolation-based method, with increase in NFRs, the accuracy of calculations increases, while calculations volume does not grow. However, it is notable that with larger case studies in size, increasing the number of NFRs, applying weight-based methods is preferred because of its more efficient features compared to AHP, TOPSIS and PTFG algorithms.

While having the aforementioned advantages, the main disadvantage of the proposed algorithm (a common disadvantage in all algorithms of Table 4, is that after adding a

new NFR, the aforementioned results are no more valid and the algorithm needs to be rerun before the new results become usable.

In future studies, researchers can analyze the polymorphic styles and apply the proposed algorithm to select the software architecture styles on them. Moreover, using this algorithm, one can create new polymorphic styles or modify them more easily to make them more efficient depending on the requirements of the stakeholders. However, other interpolation methods such as B-Spline and Fuzzy B-Spline could be examined and appropriate styles may be selected accordingly.

References

[1] M. Salama, R. Bahsoon, Analysing and modelling runtime architectural stability for self-adaptive software, Journal of Systems and Software, 133 (2017): 95-112.

[2] D.M. Phillips, T.A. Mazzuchi, Sh. Sarkani, An architecture system engineering and acquisition approach for space system software resiliency, Information and Software Technology, 94 (2018): 150-164.

[3] C.Y. Chong, S.P. Lee, T.Ch. Ling, 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, 27.1 (2014): 1-19. doi: https://ejournal.um.edu.my/index.php /MJCS/article/view/6790

[4] E.V. Woodward, R. Bowers, V.S. Thio, K. Johnson, M. Srihari, C.J. Bracht, Agile methods for software practice transformation, IBM Journal of Research and Development, 54.2 (2010): 3-1. doi**:** 10.1147/JRD.2009. 2038749.

[5] T. Sharma, D. Spinellis, A survey on software smells, Journal of Systems and Software, 138 (2018): 158-173.

[6] A. Affleck, A. Krishna, A. Narasimaha, R. Achuthan, Non-Functional Requirements Framework: A Mathematical Programming Approach, The Computer Journal, Vol. 58, Issue. 5, pp. 1122–1139, 2015. doi:10.1093/comjnl/bxu027.

[7] G.D. Abowd, R. Allen, D. Garlan, Formalizing style to understand descriptions of software architecture, ACM Transactions on Software Engineering and Methodology (TOSEM), 4.4 00190(1995): 319-364. doi: https://doi.org/10.1145/2 26241.226244.

[8] D.E. Perry, A.L. Wolf, Foundations for the Study of Software Architectures, ACM Software Engineering Notes, 17.4 (1999): 40-52.

[9] D. Garlan, M. Shaw, An Introduction to Software Architecture, Advances in Software Engineering and Knowledge Engineering, Series on Software Engineering and Knowledge Engineering, World Scientific Publishing Company, 2 (1994): 1-39.

[10] J. Bosch, P. Bengtsson, R. Smedinga, Assessing Optimal Software Architecture Maintainability, Fifth European Conference on Software Maintainability and Reengineering, 2000.

[11] F. Hoseini Jabali, S. M. Sharafi, K. Zamanifar, A Quantitative Algorithm to Select Software Architecture by Tradeoff between Quality Attributes, Procedia Computer Science, 3 (2011): 1480-1484.

[12] Q. Wang, Zh. Yang, A method of selecting appropriate software architecture styles: Quality Attributes and Analytic Hierarchy Process, University of Gothenburg/ Department of Computer Science and Engineering, 2012.

[13] Ch. Kim, D. Lee, L. Ko, J. Baik, ALightweight Value-based Software Architecture Evaluation. Eighth, ACIS International Conference on Software Engineering Artificial Intelligence, Networking, and Parallel/ Distributed Computing, IEEE Xplore, (2007): 646-649. **doi:** 10.1109/SNPD.2007.507.

[14] H. Astudillo, Five Ontological Levels to Describe and Evaluate Software Architecture, Department de Informatics, Universidad Technical Federico Santa Maria Avda, España1680, Valparaiso, Chile, 2004.

[15] S. Babamir, M. Khabazian, Evaluation of qualitative requirement analysis in software architecture, Mashhad, Iran: International conference of IT Knowledge, 2007.

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

[17] D. Garlan, K. J. Soo, Analyzing architectural styles with Alloy, In Workshop on the Role of Software Architecture for Testing and Analysis (ROSATEA), 2006.

[18] L.N. Fiondella, S. Gokhale, Importance measures for modular software with uncertain parameters, Software Testing, Verification & Reliability, 20.1 (2009): 63-85. doi: 10.1002/ stvr.420.

[19] C. Jegourel, A. Legay, S. Sedwards, Command-based importance sampling for statistical model checking, Theoretical Computer Science, 649 (2016): 1-24.

[20] F. Ivančić, Z. Yang, M.K. Ganai, A. Gupta, P. Ashar, Efficient SAT-based bounded model checking for software verificat-ion, Theoretical Computer Science, 404.3 (2008): 256-274.

[21] H. Mkaouar, B. Zalila, J. Hugues, M. Jmaiel, A formal approach to AADL model-based, software engineering.International Journal on Software Tools for Technology Transfer, (2019): 1–29. doi: https:// doi.org/10.1007/s10009-019-00513-7.

[22] N. Misaghian, H. Motameni, M. Rabbani, Prioritizing interdependent software requirements using tensor and fuzzy graph. Turkish Journal of Electrical Engineering & Computer Science, 27.4 (2019): 2697-2717. doi:10.3906/elk-1806-179.

[23] C.L. Hwang, K. Yoon, Methods for multiple attribute decision making, Multiple attribute decision making. Springer, Berlin, Heidelberg, (1981): 58-191.

[24] A. Tajdin, I. Mahdavi, N. Mahdavi-Amiri, B. Sadeghpour-Gildeh, Computing a fuzzy shortest path in a network with mixed fuzzy arc lengths using α-cuts, Computers & Mathematics with Applications 60.4 (2010): 989-1002.