Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 91 (2016) 599 – 608

Procedia Computer Science

Information Technology and Quantitative Management (ITQM 2016)

SDMF: Systematic Decision-making Framework for Evaluation of Software Architecture

Nitin Upadhyay[a],*

[a]Information Technology and Operations, Goa Institute of Management, Goa India

Abstract

The software architectural decisions are crucial and critical to the success of a software project life cycle. The set of relevant design decisions affects the quality of the software architecture. In this paper, a systematic decision-making framework is proposed by considering management and organizational factors and design goals/parameters that affect software architecture (SA) and integrating it with the technique for order preference by similarity to ideal solution (TOPSIS) to evaluate and select the quality software architecture. An illustrative case study is also mentioned to show the applicability of the proposed framework. The framework suggested in the paper should enable an architect and other key stakeholders of the software architecture to efficiently identify, evaluate and select the software architecture.

© 2016 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).
Peer-review under responsibility of the Organizing Committee of ITQM 2016

Keywords: Decision-making; software architecture; design decisions; pattern; software quality; TOPSIS

1. Introduction

Software Architecture (SA) plays a critical role in realizing the quality of the software product. The software product stakeholders are increasingly more concerned about the quality of the software product to satisfy various functional and non-functional requirements. The researchers, decision makers, managers, practitioners and product owners have identified that SA of the software system help in understanding and managing large and complex software systems [1-2] and in constraining the quality attributes [3]. A quality SA is important to achieve a high-quality software system, both regarding development and long-term maintainability. Since SA plays a critical role in realizing software quality attributes, it has become a paramount task to evaluate SA about desired quality requirements as early as possible in the software development life cycle. The SA evaluation deals with the problem of assessing and selecting the potential SA, from the pool of alternatives SA candidates, that is capable of realizing required quality requirements [4]. The detection and fixing of possible errors and faults later in the software development life cycle contribute to enormous risks and costs. Thus, an early evaluation of SA plays a significant role in understanding software quality and associated potential risks [5]. Researchers have developed many methods to evaluate quality related issues at the

* Corresponding author. Tel.: +91-083225366751; fax: +91-0832-2366710.
E-mail address: upadhyay.nitin@gmail.com

1877-0509 © 2016 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).
Peer-review under responsibility of the Organizing Committee of ITQM 2016
doi:10.1016/j.procs.2016.07.151

SA level. The conventional qualitative and quantitative SA evaluation techniques [1-2], [4] majorly focus on the limitations of the quality in SA. Most of the available methods focus mainly on assessment of single quality attribute [6]. For example, a modifiability analysis of an SA is achieved through Architecture-Level Modifiability Analysis (ALMA) [7]; Scenario based Architecture Analysis Method (SAAM) [8] is used to analyze the modifiability attribute of the SA quality.

To achieve business value and stakeholders' requirements satisfaction, it is of utmost important to evaluate SA considering multi-attribute quality analysis. Thus, there is dire need of a systematic SA evaluation framework that considers management and organizational factors and designs goals/parameters that drive the quality of the SA. Moreover, provide results quantitatively for the purpose of benchmarking and ranking of the software architectures.

The remainder of this paper is as follows: Section 2 provides the literature review of the related work. Section 3 describes the systematic decision-making framework for SA evaluation. Section 4 presents an illustrative case study that shows the utility of the proposed framework. Finally, Section 5 concludes the paper.

2. Literature Review

One of the most vital features of SA evaluation method is the number of quality attributes that a process can handle for the evaluation. Most of the methods available in the literature focus mainly on a single quality attribute analysis for SA assessment. For example, a modifiability analysis of an SA is achieved through Architecture-Level Modifiability Analysis (ALMA) [7]; modifiability quality attribute has also been analyzed through Scenario based Architecture Analysis Method (SAAM) [9]; Active Reviews for Intermediate Design (ARID) [10]; Cost-Benefit Analysis Method (CBAM) [11], [6] focuses on Costs, Benefits, and Schedule Implications; Scenario-based Architecture Level UsabiliTy Analysis (SALUTA) [12] focuses on Usability of the SA; SAAM for Complex Scenarios (SAAMCS) [13] addresses the Flexibility attribute; Extending SAAM by Integration in the Domain (ESAAMI) [14] target to analyze modifiability attribute; Aspectual Software Architecture Analysis Method (ASAAM) [15] focuses on modifiability analysis; maintainability analysis is achieved through Architecture-Level Prediction of Software Maintenance (ALPSM) [16]; performance evaluation of enterprise architecture [17]; efficiency evaluation by developing executable model [18] and serviceability analysis of service-oriented architecture [19].

There is a lack of work on the systematic evaluation of SA considering multiple quality attributes quantitatively. Currently, only two techniques Scenario-Based Architecture Reengineering (SBAR) [20] and Architecture Tradeoff Analysis Method (ATAM) [8] focus on multiple quality attributes analysis. However, SBAR focus on only development and operational related quality attributes analysis [20] and thus limit the overall assessment of the SA towards achieving business value. ATAM specifically focuses on utility tree and scenarios to make the quality analysis. In case the scenarios are not mapped precisely and consistently then the overall assessment becomes the far-reaching goal. None of the methods mentioned in the available literature consider management and organizational factors and design goals/parameters that drive the quality of the SA for the evaluation purpose.

One of the widely used decision methods is Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) which is the focus point of usage in varied disciplines [21-25]. This technique functions on the concept that the chosen alternative should have the shortest Euclidean distance from the positive ideal solution and farthest from the negative ideal solution. Generally, TOPSIS is a technique where decision maker has to perform the evaluation of the finite number of decision alternatives under the finite number of criteria. The underlying philosophy of TOPSIS is that the selected option remains at the shortest distance, in a geometrical sense, w.r.t the ideal solution and longest distance from the worst solution. The purpose of the analysis is to rank the alternatives in an order of preference.

3. Systematic Decision-making framework (SDMF)

The process of SA evaluation comes under multi-criteria decision-making problem and to attain such a process of SA evaluation a systematic decision-making framework is proposed. The framework considers management and organizational factors and design goals/parameters for software architecture evaluation and also utilizes TOPSIS technique to restructure the complex domains composed of diverse internal and external factors in the SA evaluation process. Figure 1 presents the framework.

In the framework, two-phase structure - Phase I and Phase II is proposed. Phase I deal with the identification of various management and organizational factors and design goals/parameters that drives and affect the quality of SA. This phase comprises four major concerns: Planning and Objective, Timescale and effort, Assumptions and Constraints and Domain scope. Phase II deals with an application of TOPSIS method to select and rank SA. The output generated in each phase becomes input to the subsequent phase. Each phase also receives feedback from subsequent phases. The Phases execution are iterative and is completed when the ranking is acceptable to the decision committee based on the set goal and criteria. The feedback helps evaluators, designers, decision makers and other stakeholders of the project to tune, tailor and customize the SA as per requirements

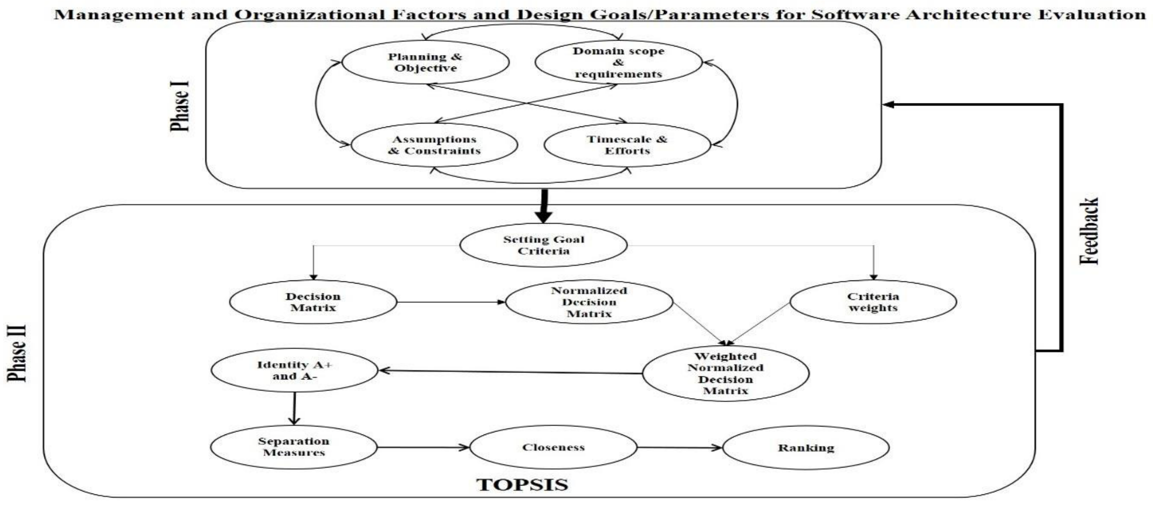
Management and Organizational Factors and Design Goals/Parameters for Software Architecture Evaluation
Phase I
Planning & Objective
Domain scope & requirements
Assumptions & Constraints
Timescale & Efforts
Feedback
Phase II
Setting Goal Criteria
Decision Matrix
Normalized Decision Matrix
Criteria weights
Weighted Normalized Decision Matrix
Identity A+ and A-
Separation Measures
Closeness
Ranking
TOPSIS

Fig. 1. Systematic Decision-making Framework

3.1. Phase I

In this phase, the main concern is to identify various organizational and management factors that affect SA. The phase comprises four steps: Planning and Objective, Timescale and effort, Assumptions and Constraints and Domain scope. The description of 4-step procedure is as follows [26]:

Step 1: Planning and Objective

In this step, an establishment of the plan and objective of the SA evaluation as per stakeholders' point of view is accomplished. It helps in identifying the critical goals that has to be accomplished for the successful completion of the SA evaluation.

Step 2: Time scale and effort

In this step, the milestones for achieving (pre)defined tasks is agreed upon. More specifically, the resources, evaluation team and staff are identified.

Step 3: Assumptions and constraints

In this step, the underlying assumptions and constraints (if any) for the SA evaluation are determined. The execution of assessing available infrastructure, support and environment and the identification of various contract and legal issues is recommended. Budget and staff and/or evaluation team training issues are also sorted out.

Step 4: Domain scope and requirements

In this step, domain scope, definition, and requirements (quality requirements) are identified.

3.2. Phase II

In this phase, the quality criteria and alternative identified from the Phase I is inputted to phase II. The TOPSIS is utilized in this phase to select and rank the SA. Below are the steps for doing the overall computation to select and rank the potential SA from the pool of alternatives.

Step 1

This step produces a decision matrix of SA evaluation criteria and SA alternatives based on the information available regarding the SA evaluation problem. In case, the number of SA alternatives is *M,* and the number of SA evaluation criteria is *N*, then the decision matrix having an order of *M* × *N* is represented as follows:

$$D_{M\times N} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ \dots & \dots & \dots & \dots \\ a_{M1} & a_{M2} & \dots & a_{MN} \end{bmatrix} \tag{1}$$

where an element a_{ij} of the decision matrix $D_{M\times N}$ represents the actual value of the *i*th alternative regarding *j*th decision criteria.

Step 2

In this step, the decision matrix is converted to a normalized decision matrix so that the scores obtained in different scales become comparable. An element r_{ij} of the normalized decision matrix *R* is calculated as follows:

$$r_{ij} = \frac{a_{ij}}{\left[\sum_{i=1}^{M} (a_{ij})^2\right]^{0.5}} \tag{2}$$

Step 3

To get the weighted normalized matrix each column of the normalized decision matrix *R* is multiplied by the associated criteria weight corresponding to that column. Hence, an element v_{ij} of weighted normalized matrix *V* is represented as follows:

$$v_{ij} = W_j . r_{ij} \tag{3}$$

Step 4

This step produces the positive ideal solution (A^+) and negative ideal solution ($A-$) in the following manner:

$$A^+ = \left\{ \left(\max v_{ij} \,/\, j \in \tau \right), \left(\min v_{ij} \,/\, j \in \tau' \right) \, for i = 1,2,3,..,M \right\} \\ = \left\{ V_1^+, \dots, V_N^+ \right\} \tag{4}$$

$$A^- = \left\{ \left(\min v_{ij} \,/\, j \in \tau \right), \left(\max v_{ij} \,/\, j \in \tau' \right) \, for i = 1,2,3,..,M \right\} \\ = \left\{ V_1^-, \dots, V_N^- \right\} \tag{5}$$

Where $\tau = \{ j = 1, 2, ..., N \}$ j is associated with benefit or positive criteria, and $\tau' = \{ j = 1, 2, ..., N \}$ j is associated with cost or negative criteria.

The most preferred one have the maximum value among the alternatives. Therefore, A^+ indicates the positive ideal solution. Similarly, *A*− shows the negative ideal solution.

Step 5

The *N* dimensional Euclidean distance method is applied, as shown in Equation (6) and Equation (7), to compute the separation distances of each alternative from the positive and negative ideal solution:

$$S_i^+ = \left\{ \sum_{j=1}^{N} \left(V_{ij} - V_j^+ \right) \right\}^{0.5}, I = 1, 2, ..., m \tag{6}$$

and

$$S_i^- = \left\{ \sum_{j=1}^{N} \left(V_{ij} - V_j^- \right) \right\}^{0.5}, I = 1, 2, ..., m \tag{7}$$

Where, S_i^+ and S_i^- are the separation distances of alternative *i* from the positive ideal solution and negative ideal solution, respectively.

Step 6
In this step the relative closeness (C_i^*) value of each alternative with respect to the ideal solution is determined using Equation (8). The value of C_i^* lies within the range from 0 to 1:

$$C_i^* = \frac{S_i^-}{\left(S_i^+ + S_i^- \right)} \tag{8}$$

Step 7
At the final step, the preference order is ranked. All the alternatives are now arranged in a descending order according to the value of C_i^*. In *TOPSIS* method, the chosen alternative has the maximum value of C_i^* with the intention to minimize the distance from the ideal solution and to maximize the distance from the negative ideal solution. Finally, the best SA alternative is recommended.

4. Illustrative Case Study

The SDMF proposed in section 3 is utilized to compare software architectures based on three different architectural patterns: publisher/subscriber with push model [27], repository [28] and broadcast pattern [28]. Notice that publisher/subscriber alternatively termed as subject/observer [29]. The Stock Exchange Monitoring System is developed using the architectures as mentioned earlier. Section 4.1 briefly presents the system requirements [30].

4.1. Requirements for a Stock Exchange Monitoring System (SEMS)

Capturing, analyzing and broadcasting events (data) in real-time is the primary goal of a real-time monitoring system. SEMS falls into a category of a soft real-time system. Thus, the whole system's behavior is not affected even if some of the events may miss their deadline. The SEMS facilitates for brokers and independent investors for monitoring in real-time small and medium size stock exchange. The components of a system categorize into – feed server, data server, client (subscriber/customer details) server and client interface (browser). An antenna that is external to the system is also termed as feed server supplies the data (feed) to the data server. Each relevant information of a stock exchange transaction is treated as a feed that is supposed to be reliable and available. To avail the data feeds the clients (brokers and customers) have to be subscribed with the data server. To avail the information, clients need not present at any particular geographical location rather the feed can be read on the move. Whenever a change is occurred in the event, for example change in feed or updation of a feed, then the subscribed clients get to know the change or updation of the feed as per the severe time delay. The delay in time is largely dependent on the network infrastructure utilized to disseminate the information. This affect the type of service offered to the clients. The browsers are used act as an interface to the data and other services.

4.2. Architectures proposed for the monitoring system

The proposed architectures based on three different architectural patterns, publisher/subscriber (push model), repository and broadcast, are shown in Figures 2, 3 and 4 respectively.

4.2.1. Publisher/Subscriber Pattern: In this candidate architecture, the publisher is responsible to publish the change of events (data) to the subscriber(s). The clients need to subscribe to access the data. The subscriber maintains the repository/database of the subscriber(s) for the data. A change in an event for example stock price change triggers the publisher to notify these changes to the subscribed. Publisher/Subscriber pattern is shown in Fig 2.

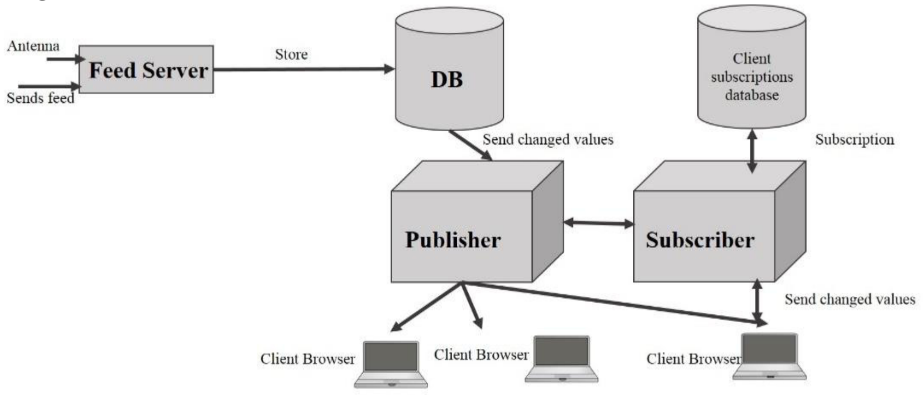

Fig. 2. Publisher/Subscriber Pattern (adapted from [31-32])

4.2.2. Repository Pattern: In this candidate architecture, the clients request the data as and when needed from the data server. This request is not pre-decided thus may or may not be done periodically. The queuing mechanism is used to handle the request conflicts. The repository is shown in Fig.3.

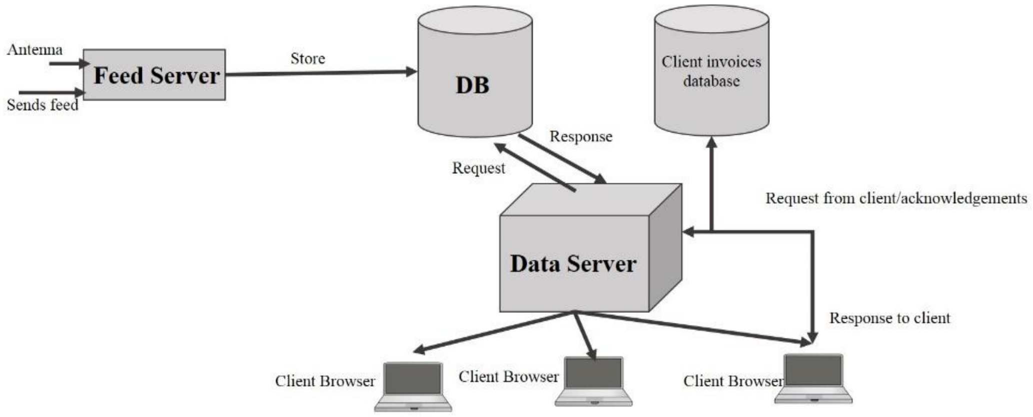

Fig. 3. Repository Pattern (adapted from [31-32])

4.2.3. Broadcast Pattern: In this candidate architecture, the change of an event is broadcasted to the clients. It is to be noted that the communication between server and client is uni-directional. The Broadcast pattern is shown in Fig.4.

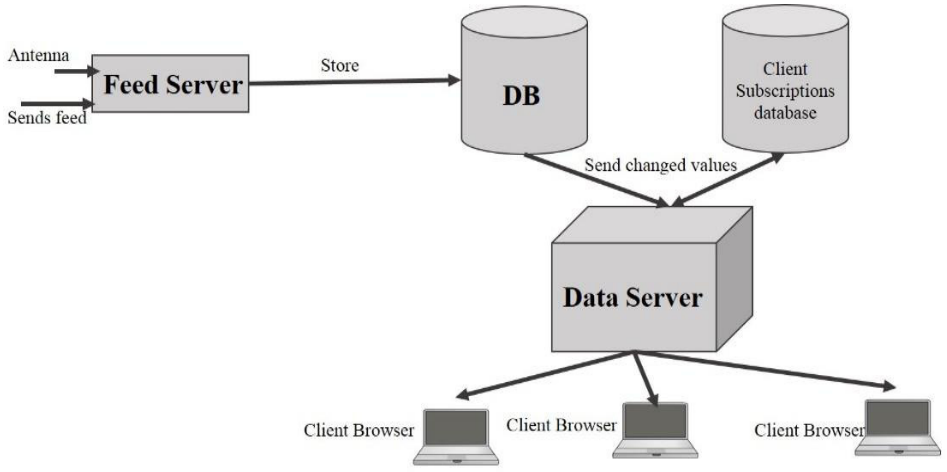

Fig. 4. Broadcast Pattern (adapted from [32])

4.3. Quality Attributes

The SDMF is utilized for the SA evaluation. In Phase I of the framework the eight quality attributes namely response time, learnability, maintainability, recoverability, reusability, cost, development time and skilled team size are identified. The SA evaluation is performed for these quality attributes in Phase II. The values quality attributes about candidate architectures are listed in Table 1. Follows are the brief description of the quality attributes.

4.3.1. Response Time

It is defined as the time required for performing a complete transaction. It is the function of time required for processing data and request, queuing request and transferring data. The unit of the attribute is milliseconds (ms). It is to be noted that since repository pattern utilizes queuing mechanism thus the response time is high.

4.3.2. Learnability

It is defined as the time required to understand the component or software and work with it. The unit of the attribute is hours (hrs).

4.3.3. Maintainability

It is defined as the time required to perform successful changes in the software. The unit of the attribute is hours (hrs). It is a function of the number of components and their interactions required for achieving the functionalities of the system.

4.3.4. Recoverability

It is the time required for recovery of a system or a component from failure state to working state. The unit of the attribute is seconds (secs). The repository SA pattern has the quick recovery for example in the case when clients fail then it can reestablish the current status by requesting the server. However, the other two SA pattern have a constraint of periodic information cycle. Thus, these systems have to wait until the next cycle begins.

4.3.5. Reusability

It is defined as the availability of the number of components and connectors that can be reused to achieve the functionality. The unit of the attribute is number (nos).

4.3.6. Cost

It is defined as the cost associated with developing the software product. The unit of the attribute is the currency - rupees (Rs). It can be seen that cost for building SEMS through repository is minimum as compared to other SA patterns as repository can be built by built by using the existing components.

4.3.7. Development Time

It is defined as the time required to develop the software. The unit of the attribute is weeks (wks). The development time is less for the repository SA pattern as it reuses existing components.

4.3.8. Skilled Team Size

It is defined as the number of skilled persons required to develop the software. The unit of the attribute is numbers (nos).

4.4. Analysis and Discussion

To validate the proposed framework for the evaluation of SA, experiments have been conducted using three architectures - publisher/subscriber (PS), Repository (R) and Broadcast (B) and eight quality attributes, (Q_{i1}, $Q_{i2}\ldots Q_{i8}$). The data for the attributes have been taken from the data specified in [32] to maintain the consistency in the usage of the measure values for the quality attributes for the SA evaluation. Preferences as weights (wt.) associated to respective quality attributes were captured from key stakeholder, see Table 2.

Phase II of the SDMF is utilized for the evaluation of the SA based on the measured values of the quality attributes of the candidate architecture. Learnability(L), Reusability(Reu) and Maintainability(M) are the benefit quality attributes. Response time(RT), Recoverability (Rec), Cost(C), Skilled Team size(TS) and Development Time(DT) are the cost quality attributes. The SDMF applicability on the evaluation of the three software architectures result into the ranking as - PS < B < R.

Table 1. Evaluation - Selection and Ranking using SDMF

	Response time (ms)	Learnability (hrs)	Reusability (nos)	Maintainability (nos/time in hrs)	Recoverability (secs)	Cost (rs in lacs)	Skilled team size (nos)	Development time (week)
PS	10	5	1	200	20	8	20	60
R	20	8	5	25	10	4	10	30
B	12	3	1	200	5	6	5	20

Thus, publisher/subscriber software architecture pattern is benchmarked as best among all the three alternatives. The results produced by the SDMF is consistent with the results mentioned in [32]. It has been verified that the SDMF proposed in the paper is effective and consistent for the evaluation - selection and ranking of Software Architecture. The framework provides benefits to the different stakeholders in SA evaluation e.g. architect, decision makers, modelers, analysts, designers, developers, quality analysts, domain experts, testers, product manager and consultants. The framework is flexible as it allows customization or tailoring and meeting various projects' requirements. The feedback mechanism is available at each phase which allows stakeholders to review the current status and takes appropriate decisions. The system analyst, decision makers and designer can generate alternative design solutions and select the optimum one by using the framework along with morphological chart/tree. The framework is capable enough to consider multiple and extended attributes of interest for the stakeholders to evaluate the SA.

Table 2. Evaluation - Selection and Ranking using SDMF

	Q_{i1}	Q_{i2}	Q_{i3}	Q_{i4}	Q_{i5}	Q_{i6}	Q_{i7}	Q_{i8}
PS	10	5	1	200	20	8	20	60
R	20	8	5	25	10	4	10	30

B	12	3	1	200	5	6	5	20
Wt.	0.35	0.25	0.25	0.15	0.15	0.15	0.15	0.15

	S^+	**S^-**	**C_i^***	**Ranking**
PS	0.269	0.173	0.391	3
R	0.173	0.262	0.605	1
B	0.233	0.219	0.484	2

5. Conclusion

This research work present systematic decision-making framework for evaluating – selecting and ranking the ‘right' software architecture. The framework provides benefits to architect, R&D experts, designers, evaluators, decision makers and other key stakeholders to accept the design at the conceptual stage by considering all the critical quality, functional and non-functional factors/criteria. The framework proposes a new decision-based model, which considers all the key factors/criteria in an integrated approach without losing any useful information. The validation of the proposed framework has been accomplished using a suitable case study. The framework opens new dimensions in the field of effective design and building of software systems. Finally, the underlying concept of this method is rational and comprehensive.

Acknowledgements

The author would like to thank the Goa Institute of Management for supporting the research study.

References

[1] Kuwahara Y and Y. Takeda. A managerial approach to research and development cost effectiveness evaluation, *IEEE Trans. Eng. Manage.*, 1990; **37(2)**: 134–138.
[2] Svahnberg M, C. Wohlin, L. Lundberg and M. Mattsson. 2002. “A Method for understanding Quality Attributes in Software Architecture Structures,” Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering, p. 819-826.
[3] Zayaraz G and Thambidurai P. Quantitative Model for the Evaluation of Software Architectures, *Journal of Software Quality Professional*, **9 (3)**,2007; 28-40.
[4] Zayaraz G and Thambidurai P. 2005. “Software Architecture Selection Framework Based on Quality Attributes,” Proceedings of the IEEE Conference INDICON, p. 67-170.
[5] Triantaphyllou E. The impact of aggregating benefit and cost criteria in four MCDA methods. *IEEE Transactions on Engineering Management* 2005; **52(2)**.
[6] Shanmugapriya P and Suresh R. M. Software Architecture Evaluation Methods – A survey. *International Journal of Computer Applications* 2012; **49(16)**
[7] Bengtsson P, Lassing N, Bosch J, and Vliet H. V. Architecture-Level Modifiability Analysis, *Journal of Systems and Software*, 2004; **69**.
[8] Kazman R, Klein M, Barbacci M, Longstaff T, Lipson H, and Carriere J. 1998. “The Architecture Tradeoff Analysis Method,” Proceedings of IEEE, ICECCS.
[9] Kazman R, Bass L, Abowd G and Webb M. 1994. “SAAM: A Method for Analyzing the Properties of Software Architectures,” Proceedings of the 16th International Conference on Software Engineering.
[10] Clements P. *Active Reviews for Intermediate Designs*, SEI, Carnegie Mellon University CMU/SEI-2000-TN-009, 2000.
[11] Clements P, Kazman Rand Klein M. *Evaluating Software Architectures: Methods and Case Studies*, Addison Wesley, 2002.
[12] Folmer E, Gurp J and Bosch J. 2004. “Software Architecture Analysis of Usability,” Proceedings on 9th IFIP Working Conference on Engineering Human Computer Interaction and Interactive Systems, 2004, p. 321-339.
[13] Lassing N, Rijsenbrij D and Vliet H. V. 1999. “On Software Architecture Analysis of Flexibility, Complexity of Changes: Size isn't Everything,” Proceedings of 2nd Nordic Software Architecture Workshop, 1999.
[14] Molter G. 1999. “Integrating SAAM in Domain-Centric and Reuse-based Development Processes,” Proceedings of the 2nd Nordic Workshop on Software Architecture, 1999.
[15] Tekinerdogan B. 2004. “ASAAM: aspectual software architecture analysis method,” Proceedings of the Working IEEE/IFIP Conference on Software Architecture (WICSA'04), p. 5-14.
[16] Bengtsson P and Bosch J. 1999. “Architectural Level Prediction of Software Maintenance,” Proceedings of 3rd European Conference on Software Engineering Maintenance and Reengineering, 1999.
[17] Atasheneha M, Harounabadi A and Mirabedinib S J. Performance evaluation of enterprise architecture using fuzzy sequence diagram. *Decision Science Letters*, 2014; **3**: 103–108

[18] Haghania S K, Abbasnejada Y and Harounabadib A. An evaluation of the software architecture efficiency using the Clichés and behavioral diagrams pertaining to the unified modeling language. *Decision Science Letters*, 2014; **3**: 411–430
[19] Atasheneha M, Harounabadi A and Mirabedinib S J. Service oriented architecture assessment based on software components. *Decision Science Letters*, 2016; **5**: 109–118
[20] Bengtsson P and Bosch J. 1998. "Scenario-based Architecture Reengineering," Proceedings of the 5th International Conference on Software Reuse, 1998.
[21] Jee D H, Kang K J. A method for optimal material selection aided with decision-making theory. *Materials and Design* 2000; **21**: 199–206.
[22] Prabhakaran R T D, Babu B J C and Agrawal V P. Optimum selection of a composite product system using MADM approach. *Materials and Manufacturing Process*, 2006; **21**: 883–891.
[23] Satapathy B K, Bijwe J. Wear data analysis of friction materials to investigate the simultaneous influence of operating parameters and compositions. *Wear* 2004; **256**: 797–804.
[24] Tong KW, Kwong CK, Ip KW. Optimization of process conditions for the transfer molding of electronic packages. *Journal of Materials Processing Technology* 2003; **138**: 361–365.
[25] Wang T Y, Shaw C F, Chen Y L. Machine selection in flexible manufacturing cell: a fuzzy multiple attribute decision-making approach. *International Journal of Production Research* 2000; **38**: 2079–2097.
[26] Upadhyay N, Deshpande B and Agrawal V P. Integrated decision approach for COTS selection. *International Journal of Decision Sciences, Risk and Management* 2010; **2(3-4)**: 165-177
[27] Buschman F., Meunier R., Rohnert H., Sommerlad P., Stal, M. *Pattern-Oriented Software Architecture. A System of Patterns*, John Wiley & Sons Inc., New York, 1996.
[28] Shaw M., Garlan D. *Software Architecture – Perspective of an Emerging Discipline*, Prentice Hall, Upper Saddle River, New Jersey, 1996.
[29] Gamma E., Helm R., Johnson R., Vlissides J. *Design Patterns – Element of Reusable Object-Oriented Software*. Addison Wesley, New York, 1995.
[30] Ordaz Jr. O. Aplicaciones Tempo Real en Internet: Arquitecturas, Lenguajes y un Caso de Estudio, License Thesis, Universidad Central de Venezuela, Caracas, 2000.
[31] Losavio . F, L. Chirinos, N. Levy and Ramdane A. Quality Characteristics of Software Architecture, *Journal of Object Technology* 2003; **2(2)**: 133- 150.
[32] Vijayalakshmi S., Zayaraz G and Vijayalakshmi V. Multicriteria Decision Analysis Method for Evaluation of Software Architectures. *International Journal of Computer Applications* 2010; **1 (25)**.