IJCCN | INTERNATIONAL JOURNAL OF COMPUTING & COMMUNICATION NETWORKS

ISSN: 2664-9519 (Online); Vol. 1, Issue 2, November 2019

Software Architecture Evaluation Methods: A Comparative Study

Ali Raza[1], Shaista Zafar[2], Saeed Ur Rehman[3], and Umar Khattak[3]

[1] University of Sialkot, Department of Software Engineering, Sialkot, Punjab, Pakistan

[2] University of Lahore, Department of Software Engineering Gujrat Campus

[3] Department of Computer Science, Bahria University Islamabad

* Corresponding Author: Ali Raza; Aliraza.1354@gmail.com

Received 4 August 2019; Revised 20 October 2019; Accepted 10 November 2019; Published 20 November 2019

Abstract: Software Quality is considered as an important aspect throughout the entire software development process. Software quality makes sure the satisfaction of client needs and requirements. Software architecture plays a key role in an efficient software development process. Evaluation of software architecture involves different properties of the evaluation including strengths and shortcomings of the software architectural style or design pattern. Software architecture evaluation methods are utilized to analyze the design based on the suitability of the architecture. In this survey paper, we examined different existing scenario based techniques for the evaluation of software architecture. The comparison between these evaluation methods is also presented based on different architectural behavior.

Keywords- Software, Engineering, Architecture, Software testing, Scenario-Based Architectural Evaluation.

1. Introduction

Software architecture evaluation is a technique which evaluates the properties, shortcoming, and qualities of software architectural style or a framework plan [1]. Software architecture evaluation offers affirmation to the creators that the selected architecture will assemble both functional and non-functional quality requirements. Architectural evaluation should give a greater number of favorable circumstances rather than the rate of directing the evaluation itself. Such evaluation ensures extended understanding and documentation of the structure, acknowledgment of issues with existing architecture and enhances various hierarchical learning.

The objective of software architecture evaluation is to study the abilities of the chosen architecture to pass on a system fit for fulfilling required quality necessities and to recognize any potential risks. Moreover, it is concise and less exorbitant to recognize and settle outline mistake during the basic periods of the software development. That is the only reason an effective software architecture evaluation method to analyze potential architectures is of extraordinary business significance. A couple of strategies have been suggested to deal with the quality-related issues at the software architecture level [2, 3]. Software architecture has a critical influence in accomplishing framework wide quality attributes.

Quality prerequisites are ahead of schedule as could reasonably be expected. A classification of evaluation techniques on scenario-based methods is considered extremely developed approach [4, 5]. On the other hand, quantitative models and attribute models are also viewed as equally important. Existing techniques are vanishing as new ones are emerging and occupying the position of old ones. In this way, it is getting particularly difficult to find the likeness and contrasts among different techniques. There is little work on systematically evaluating or comparing existing methods and perceiving an arrangement of segments. The comparison of software architecture evaluation methods can enhance the understanding of the existing methodologies' which offer potential research directions to the researchers [6, 7].

This paper presents a comprehensive comparison of the following software architecture evaluation techniques, including: Software Architecture Analysis Method (SAAM), Architecture based Tradeoff Analysis Method (ATAM), Architecture-Level Modifiability Analysis (ALMA), SAAM for Complex Scenario (SAAMCS), Cost Benefit Analysis Method (CBAM), Family- Architecture Assessment Method (FAAM). We aim to review existing scenario-based architectural evaluation methods and present a comprehensive comparative analysis.

In scenario-based architecture evaluation methods a scenario profile is constructed which aims to evaluate a particular quality attribute that strengthens an extremely solid portrayal of the quality constraint. The results of the scenarios are documented after each scenario from the profile is wander through software. The scenario-based evaluation methods offer methodical plans to investigate software architecture using scenarios. These methods make sure that the software architecture can accomplish a scenario. In contrast, if the design of product fails to execute the scenario, these methods list the movements to the software configuration required to fortify the situation.

The rest of the paper is structured as follows: in section 2, related work is presented. Section 3 explains the detail description and comparison of the scenario-based software evaluation methods. Finally, section 4 concludes this research work.

2. Classification of Approaches

Complex software has complex architecture [8] and sometimes it is difficult to execute the plan choices encapsulated by the architecture. Therefore, software architecture ought to be assessed before submitting resources to it. Techniques to assess software architecture have been developing with various methodologies. Architecture evaluation strategies focus on various quality properties, for example, reusability, modifiability, accessibility, practicality, security, and testability [9, 10].

We have examined and compared the recent research work based on evaluation methods of software. To the best of our knowledge, this paper is the first attempt that presents a detailed comparison of scenarios-based software architecture evaluation methods. In an earlier study [11] compared three architecture evaluation methods including, SAAM, ATAM, and ARID. These three evaluation methods are proposed by the Software Engineering Institute. Their comparison lacks some imperative components for assessment technique. In another work by [12, 13] proposed a comparison framework where software architecture evaluation methods are compared with each other. In another survey paper [14] compared architecture assessment methods with respect to the setting of architectures in software product offerings. The survey [15] does not address a substantial number of architecture assessment techniques yet utilizes two assessment techniques as cases for representing how the technique satisfies various criteria the researcher contend are exceedingly required for an architecture assessment technique to be usable.

Following are the set of evaluation methods:

- *SAAM (Software Architecture Analysis Method)*
- *ATAM (Architecture based Tradeoff Analysis Method)*
- *ALMA (Architecture-Level Modifiability Analysis)*
- *SAAMCS (SAAM for Complex Scenario)*
- *CBAM (Cost Benefit Analysis Method)*
- *FAAM (Family- Architecture Assessment Method)*

2.1 SAAM (Software Architecture Analysis Method)

SAAM (Software Architecture Evaluation Method) [3] proposed in 1993. Initially, SAAM is pronounced as the Scenario-based Architecture Evaluation Method [6]. SAAM focused on surveying only one architecture or making a couple of architectures using estimations, such as coupling between architectural components [12]. It aims to review the structures' modifiability in its different names [16].

A. SAAM Purpose

SAAM is proven to be helpful for rapid analysis of numerous quality properties [15], for example, modifiability, portability, extensibility, integrability and functional coverage. This method can likewise be utilized to review quality aspects of software architecture such as reliability and performance. SAAM is originally proposed to help in comparing architectural solutions [17]. Generally, the objective of SAAM is to evaluate the software architecture in opposition to the needed quality [3]. SAAM can differentiate particular software architectures with respect to given properties. SAAM and its variations are connected to various areas as well as CASE tools and combat frameworks.

B. Steps in SAAM

There are six steps in SAAM, which are discussed as follows:

1. **Specify requirements and design constraints:** In the first step, SAAM accumulate both functional & non-functional requirements and summarizes design constraints. The evaluation panel gathers requirements from the stakeholders and indicates the prerequisites for a gathering meeting.
2. **Describe software architecture:** In the second step, SAAM displayed the candidate architectures [18]. The evaluation team and participants surely understood the architectural details. SAAM utilizes particular architectural perspectives to clarify the functional view [13]. Furthermore, the architect utilizes a module coordination view and dynamic perspective to understand the dynamic conduct of the system.
3. **Elicit scenarios:** Scenarios are evoked with the presence of suitable stake-holders. The scenarios describe the tasks recognized by the stakeholders, developers, and architects. The meeting is held in which stakeholders exchange their assumptions to generate new ideas and for making a delegate set of scenarios that deal with imperative quality attributes. SAAM analyze a lot or less endless supply of elicited scenario. Scenarios ought to get various things such as fundamental utilization of the framework and the qualities that the system must fulfill now and soon.
4. **Prioritize Scenarios:** Scenarios are prioritized by their significance. It permits the basics scenarios to be tended within the limited measure of time available for evaluation. The views of the stakeholders play a vital role in determining the importance of the scenario. The significance of the scenarios depends on the views of the stakeholders. The prioritization of the scenarios depends on the voting system. Since SAAM deals with the system's modifiability consideration; the voting results will be an arrangement of indirect scenarios that are more expected to happen.
5. **Evaluate architectures w.r.t scenarios:** In the 5th step, the outcome of scenarios on the software architecture is considered. SAAM classifies two types of scenarios. In the indirect scenario, the architect illustrates how the architecture would be changed according to a suitable scenario. The direct scenario does not cause changes to the architecture. In this scenario, the architect reveals how the scenario will be implemented by the architecture.
6. **Create an overall Evaluation:** SAAM create the evaluation results according to the defined evaluation goals, just before the end of architecture evaluation regarding the scenarios. A weight is allotted to every scenario regarding its relative significance to the accomplishment of the system. In light of this scenario, weighting can propose a general positioning if different architectures are looked at. An alternative for the most appropriate architecture can be proposed, wrapping the direct scenarios and require minimum changes in supporting the indirect scenario.

2.2 ATAM (Architecture based Tradeoff Analysis Method)

ATAM is a software architecture evaluation method that is used for evaluating the quality attributes, for example, modifiability, compactness, security, and extensibility [12]. Besides evaluation of attributes, ATAM investigates interaction and trade-off system's interdependencies [19]. It is a scenario-based method where a scenario depicts association with the system from the customer's perspective.

A. ATAM Purpose

ATAM investigations how well software architecture fulfills specific quality objectives. The purpose of ATAM is to give a guideline method for understanding the capacity concerning numerous contending quality attributes of software architecture. ATAM perceives the need of trade-off among different quality attributes when the software architecture of a system is determined and before the system is produced [12]. Software architecture description and business objectives specification are inputs for ATAM. Trade-off points, software architecture methodologies, sensitivity points, risks, and scenarios are outputs of ATAM as discuss:

- The entire IT Infrastructure deployed as a cyber-secure - the smart city should follow standards like ISO-27001, ISO-22301, ISO-37120, BSI-PAS 182, for Wi-Fi access PEAP (Protected Extensible Authentication Protocol, 3GPP (3rd Generation Partnership Project) and related.
- Generic APIs should be published and application should be based on standard protocols like JSON/XML/Html.
- At network security level the information and data flow must be authenticated and secure via valid encryption and confidentially to be maintained at all the communication end ports and endpoints.
- Plan for the Wireless broadband architecture should be Fiber Optical System based and should be interoperable and connective with other land and wireless communication devices.
- Authentication system to be present at the nodal endpoints of all echelons of processing and communication systems capable of heterogeneous data management. To minimize the latency issues, standard network protocols to be used at different communication layers for data flow. All deployed applications should be indigenously hosted and developed.
- Updating of all software and firmware's, all modules to be proficient in auditing and logging, elimination of backdoors and undocumented hard cored accounts to ensure compliance with vendor, peer to peer solution with full-service availability for which a service agreement should be materialized for a minimum period of 3 years since systems operations.
- Appropriate teams to be in place for monitoring and mitigation of cyber incidents and information of such to be shared with Emergency Response Team and Federal Critical Information Protection Infrastructure Centre for recovery at any eventuality.

ATAM is depicted in Figure 1.

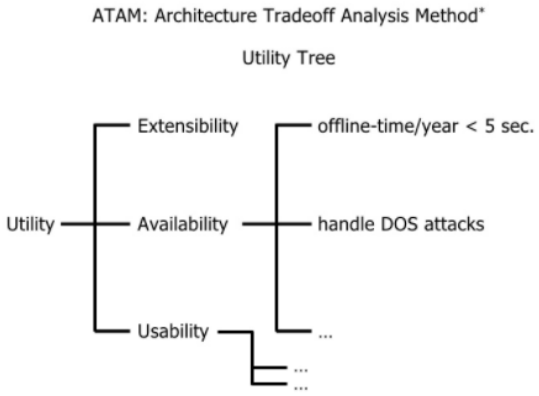

Figure 1: ATAM Utility Tree

B. Steps in ATAM

ATAM method comprises four stages: 1: Presentation, 2: Investigation and Analysis, 3: Testing and 4: Reporting. These four steps can be further divided into the subprocess. The four stages are present in the following discussion.

Presentation

1 **Present ATAM:** At first, the group leader tells the participants about the ATAM. All the queries of the participants are answered and the expected outcome of the product is discussed in detail.
2 **Present Business Drivers:** Project representative depicts what business objectives rousing development exertion. He shows a system outline from a business point of view [19].
3 **Present Architecture:** In this step, the architect presents system architecture at the best possible level of detail. The "best possible level" relies on upon a couple of elements [15]: the measure of the architecture has been arranged and recorded; how much time is available; and the method for the behavioral and quality essentials.

Investigation and Analysis

4 **Identify Architectural Approach:** In this step, the better approach to architecture is identifying software quality attributes. All approaches affect quality attributes differently.

5 **Generate Quality Attribute Utility Tree:** The evaluation team identifies quality attributes and then prioritizes.
6 **Analyze Architectural Approach:** The architectural approaches that are distinguished in previous step 4 are analyzed. In this step different possible risks and sensitive points are also distinguished.

Testing

7 **Brainstorm and prioritize Scenarios:** The evaluation team involves stakeholders in the brainstorming activity, give them many scenarios and select the best one by voting.
8 **Reanalyze Architectural Approach:** The scenarios are used that are priorities in the previous step as input for recycling of step 6. All other points are also identified and document.

Reporting

9 **Present Result:** Finally, the evaluation team presents all information and requirements to the stakeholders in summarized form for the ATAM evaluation and all the points are documented.

2.3 SAAMCS (SAAM for Complex Scenarios)

SAAMCS is an augmentation of SAAM, which acquires exercises of SAAM [8]. The fundamental objective of SAAMCS is to evaluating hazards amid framework change. The strategy handles particular issues; consequently, it is intended to execute modifiability or adaptability quality characteristics. SAAMCS is connected to the last archive of the architecture outline. The stakeholder inclusion in the strategy is the same as SAAM. SAAMCS assumes a real part of scenario initiators to execute a specific scenario. It characterizes an estimation instrument to distinguish complex scenarios in the framework. A two-dimensional system graph is utilized to find complex scenarios. This technique has been approved in the business data framework utilizing contextual analyses. There is no any device accessible for technique computerization.

SAAMCS describe a class of scenario that is conceivably perplexing to figure it out. Scenarios describe three components that control the complex nature of scenarios. These components are as follows: impact of the scenario on SA, coordination among different administrators and proximity of clashes. SAAMCS examine the software architecture by measuring at what level the software architecture manipulates these components to address the boggling scenarios.

2.4 Cost-Benefit Analysis Method (CBAM)

The CBAM simplifies architecture-based financial analyses of software-intensive systems. This technique aids the systems investors to select among architectural choices for enhancing the design and different maintenance phases of the system.

This method is a scenario-based analysis method suitable for evaluating the cost, benefits and schedule implications of architecture decisions.

A. CBAM Purpose

CBAM connects two areas (architecting process and economics) in software development of the organization. CBAM is calculating costs as quality elements, which essential to be careful when planned a software system. SAAM and ATAM mainly measured design outcomes with features like modifiability, performance, availability, usability. CBAM claiming that expenses, profits, and possibilities are essential quality attributes. SAAMCS describe a class of scenario that is conceivably perplexing to figure it out. Scenarios describe three components that control the complex nature of scenarios. These components are as follows: impact of scenario on SA, coordination among different administrators and proximity of clashes. SAAMCS examine the software architecture by measuring at what level the software architecture manipulates these components to address the boggling scenarios.

B. Steps of the CBAM

1. **Collate scenarios:** Examine the situations inspired throughout the ATAM practice and provide the possibility to stakeholders to contribute new ones. Prioritize these situations based on fulfilling the business objectives of the system and select the best for further study. We started eventually eliciting a situated from claiming situations from assembled stakeholders. A subset of raw situations places forward by stakeholders.
2. **Refine scenarios:** Improve the scenarios, centering around their stimulus/reaction measures. Inspire that worst, existing, desired, also best-case quality-attribute-response level to every situation. In light of this representational of the situation, the stakeholder group continued to vote. This group selected to converse each situation and come to a purpose. This might have been a sensible resolve, and that additional accuracy in votes might have been not required and might not be advocated. It facilitated the team rapidly influence consensus on the comparative significance of the situation.

3. **Assign intra-scenario utility:** In this step, the utility to every scenario has been resolved eventually by the stakeholder, once more by a consensus process. A utility score of 0 speaks to no utility; a score from claiming 100 speaks to the vast majority utility workable. This step demonstrated to be valued because it helped the team rapidly reach a consensus on how they assumed the system should progress.
4. **Develop architectural plans and control their projected quality attribute-reaction levels:** Create (or catch as of now developed) structural methodologies. That location those picked situations and determine the needed quality-element-response. Stages will result in starting to implement these structural methodologies. Provided that a structural method might influence numerous scenarios, this computation must make performed for every influenced situation.
5. **Calculate Desirability:** According to those inspired qualities those assessments group the desirability level to every structural technique. In light of the proportion "benefit divided by cost". Moreover, a greater amount there may be computed those vulnerabilities connected with these principles, which serves as the last step of settling on choices.
6. **Make Decisions:** Values brought in previous step and degree from claiming practicality those values there need aid decided those best cost-benefit actual structural methodologies which might satisfy best elicited descriptive situations.

2.5 ALMA (Architecture-Level Modifiability Analysis)

This method is defined in 1996. It is based on change scenarios. A changing scenario is an explanation of particular occurrence that might be taking place in the life cycle of the system and requires the system to be improved. Change cases and change scenarios are used interchangeably and hence capture the same meaning, and potentially mean the changes. Change cases would attach to methodologies supporting use cases; progress situations don't require such a situation.

A. ALMA Purpose

ALMA is a scenario-based investigation technique appropriate for software design modifiability calculation by employing a set of indicators: maintenance budget prediction and hazard evaluation. In the situation of evaluating and relating different system, those modifiability examinations performed with ALMA helps software architecture choices as well. To this end goal, ALMA practices change-scenarios gave by system stakeholders.

B. Steps of ALMA

1. **Goal Setting:** Architecture evaluation must occur inside the context of stakeholders, necessities or requirements. The initial step takes in the analysis to set the objectives. These objectives determine the analysis results. Also, the objects impact those techniques choices to be used in the following steps.
2. **Description of Software Architecture:** After the setting of analysis goals, the next step is to make the description of software architecture. Architecture design utilized inside the development group is a significant source of data. The system architect might be asked for additional data like an estimation of component size and other architectural components. The description of software architecture serves two needs inside the analysis. As a matter of first importance, it must result in an explanation that is in detail to empower architecture level influence analysis for a set of progress scenarios, and secondly, evaluate their impact given the software architecture.
3. **Scenario elicitation:** The most vital steps are elicitation of an arrangement of change scenarios in modifiability analysis. This arrangement of change scenarios catches the actions that stakeholder hopes to happen in the future in the system. The principle method to evoke this set to meet stakeholders, since they are in the best situation to anticipate what may occur later in the system. Moreover, they can evaluate the probability of change scenarios attained.
4. **Scenario evaluation:** In the wake of evoking an arrangement of progress situations, we regulate their impact on the framework. To do as such, we execute the architecture level effect examination for each situation independently. This implies to decide the segments of the framework and parts of different frameworks that must be adjusted to actualize the change situation. Additionally, parts to be included or erased are recognized. This undertaking is normally performed as a team with individuals from the improvement group.
5. **Interpretation:** After determining the impact of change situations, one can interpret these outcomes to decide on the framework under investigation. The way outcomes are explained is subject to the objective of the examination. The on-off chance that the objective of the investigation is hazard appraisal, the outcomes of the situation assessment are explored to figure out which change situations posture dangers, i.e. in which the result of situation likelihood and expenses is too high. The proprietor of the framework ought to settle on the criteria for figuring out which qualities are still satisfactory.

2.6 FAAM (Family-Architecture Assessment Method)

FAAM method is used for the evaluation of information system family architecture, it concentrating on quality attributes: interoperability and extensibility.

A. FAAM Purpose

The purpose of FAAM is to focus on the quality attributes and architecture evaluation. In the creation process of a product, FAAM involves the product stakeholders.

B. Steps in FAAM

The FAAM method consists of six steps which are presented as follows:

1. **Define the Assessment Goal:** This is an essential practice proposed to decide the objective of the assessment. Firstly, define the content and also the scope of the system with stakeholder. Define the plans for the future changes in the system [20]. Provide rules to stakeholders to create necessities.
2. **Prepare System-Quality Requirements:** The stakeholders' cooperation in recognizing and organizing system-quality necessities is asked. The challenge here is to give the way to stakeholders to speak to the necessities in an organized, appraisal prepared way.
3. **Prepare Architecture:** In this step, getting the representation of architecture accessible for presentation and evaluation against the stakeholders' prerequisites and also provide detail guidelines to the architects for presentation [21].
4. **Review Artifacts:** The challenge here is to clear up the business or intelligent imperatives that may impact the evaluation continuation.
5. **Assess Architecture Conformance:** The design depiction is confirmed against the predetermined necessities with concentrate on the capacity and easy to incorporate or to fulfill the change-cases indicated in Step 2.
6. **Report Results and Proposals:** In this stage, the appraisal results are recorded and conveyed back to the stakeholders [11]. In view of the outcomes of Step 5, the facilitator together with the architecting group draws the lessons learned from the evaluation work out.

3. Comparison OF SCENARIO-BASED EVALUATION METHODS

A few software architecture evaluation methods have been talked about in scenarios in the earlier segment. SAAM is the premise of all evaluation techniques. In this section, a comparison of different evaluation methods has been done with the help of architecture evaluation factors i.e. method's goal, evaluation approaches, QA 's, appropriate project stage, method's activities, stakeholder involvement, applications of method and tool support. Table 1 shows the comparison evaluation of discussed methods.

Table 1: Comparison of Evaluation Methods

Comparison Elements	SAAM	ATAM	ALMA	CBAM	SAAMCS	FAAM
Method's Goal	Risk identification and architectural suitability	Sensitivity and Trade-off Analysis	Maintenance cost, risk prediction, calculation, architectures comparison	Give business measures to specific system changes Make express the vulnerability connected with the appraisals	Risk assessment, developing a complex scenario for flexibility quality attribute	Helps stakeholders in identify the future changes
Evaluation Approaches	Scenario-based functionality and change Analysis	Integrates Questioning and measuring techniques	Scenario elicitation centered on goals of estimation.	Evaluate the benefits of the distinctive architectural system, assess quality & estimate the interest w.r.t cost and time aspect	Scenario base (for complex scenarios)	Information -system architecture assessment

Quality Attributes	Modifiability	Multiple Quality Attributes	Modifiability	Costs, Benefits, and Schedule Implications	Flexibility	Interoperability and extensibility
Appropriate project stage	Once functions assigned to modules	After software architecture/ detailed design or iterative improvement process	During design phase	Calculate Quality profits, cost and schedule effects of architectural plans	In the final version of software architecture	
Activities of Methods	Six activities as well as some activities work parallel	Six activities in four phases	Five activities are used.	Six principle steps, measure the Quality advantages, cost and plan effects	Three activiti, two in parallel executed	Six activities are used
Stakeholder's Involvement	All major stakeholders	All relevant stakeholders	Relevant stakeholder.	All major stakeholder.	All major stakeholders	Major stakeholder
Scenario Classification and impact Analysis	Direct and Indirect Scenarios.	Use-case, growth & exploratory scenarios.	Alternate situations. Estimates change the size of each element.	Direct, indirect and examining scenarios. Calculate Time and cost used by scenarios.	Scenarios based for complex scenarios	For architectural views

4. CONCLUSION

In this paper, we have overviewed the six software architectural evaluation methods using an extended version of the comparison framework. Architecture evaluation criteria help us to choose the ideal architecture evaluation method for the development of software. The discussed scenario-based architectural evaluation methods are SAAM, ALMA, CBAM, FAAM, SAAMCS, ATAM. These evaluation methods have been evaluated on few parameters, for example, Method's goals, Evaluation approaches, QA's, Appropriate project stages, Activities of Methods, Stakeholder involvement and scenario classification and impact analysis. Just a single technique, ATAM, gives far-reaching process support.

References

[1] L. Etxeberria and G. Sagardui, "Product-line architecture: New issues for evaluation," in *International Conference on Software Product Lines*, 2005, pp. 174-185: Springer.

[2] A. Patidar and U. Suman, "A survey on software architecture evaluation methods," in *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, 2015, pp. 967-972: IEEE.

[3] R. Bahsoon and W. Emmerich, "Evaluating software architectures: Development stability and evolution," in *Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications, Tunis, Tunisia*, 2003, pp. 47-56: IEEE Computer Society Press.

[4] S. Mahmood, A. Ullah, and A. K. Kayani, "Fog Computing Trust based Architecture for Internet of Things Devices," *International Journal of Computing Communication Networks*, vol. 1, no. 1, pp. 18-25, 2019.

[5] K. N. Qureshi, A. H. Abdullah, F. Ullah, and Informatics, "Beaconless Packet Forwarding Approach for Vehicular Urban Environment," *Bulletin of Electrical Engineering Informatics*, vol. 5, no. 2, pp. 253-262, 2016.

[6] H. Nafea, K. Kifayat, Q. Shi, K. N. Qureshi, and B. Askwith, "Efficient Non-Linear Covert Channel Detection in TCP Data Streams," *IEEE Access*, 2019.

[7] S. Iqbal, A. H. Abdullah, and K. N. Qureshi, "An adaptive interference-aware and traffic-aware channel assignment strategy for backhaul networks," *Concurrency Computation: Practice Experience*, p. e5650, 2019.

[8] E. Anjos and M. Zenha-Rela, "A framework for classifying and comparing software architecture tools for quality evaluation," in *International Conference on Computational Science and Its Applications*, 2011, pp. 270-282: Springer.

[9] M. S. Aliero, I. Ghani, K. N. Qureshi, and M. F. a. Rohani, "An algorithm for detecting SQL injection vulnerability using black-box testing," *Journal of Ambient Intelligence Humanized Computing*, pp. 1-18, 2019.

[10] K. N. Qureshi, F. Bashir, and A. H. Abdullah, "Provision of Security in Vehicular Ad hoc Networks through An Intelligent Secure Routing Scheme," in *2017 International Conference on Frontiers of Information Technology (FIT)*, 2017, pp. 200-205: IEEE.

[11] M. T. Ionita, D. K. Hammer, and H. Obbink, "Scenario-based software architecture evaluation methods: An overview," in *Workshop on methods and techniques for software architecture review and assessment at the international conference on software engineering*, 2002, pp. 19-24.

[12] L. Zhu, A. Aurum, I. Gorton, and R. J. S. Q. J. Jeffery, "Tradeoff and sensitivity analysis in software architecture evaluation using analytic hierarchy process," vol. 13, no. 4, pp. 357-375, 2005.

[13] M. Mattsson, H. Grahn, and F. Mårtensson, "Software architecture evaluation methods for performance, maintainability, testability, and portability," in *Second International Conference on the Quality of Software Architectures*, 2006: Citeseer.

[14] L. Dobrica and E. Niemela, "A survey on software architecture analysis methods," *IEEE Transactions on software Engineering*, vol. 28, no. 7, pp. 638-653, 2002.

[15] P. Shanmugapriya and R. Suresh, "Software architecture evaluation methods-a survey," *International Journal of Computer Applications*, vol. 49, no. 16, 2012.

[16] M. A. Babar and I. Gorton, "Comparison of scenario-based software architecture evaluation methods," in *11th Asia-Pacific Software Engineering Conference*, 2004, pp. 600-607: IEEE.

[17] M. A. Babar, L. Zhu, and R. Jeffery, "A framework for classifying and comparing software architecture evaluation methods," in *2004 Australian Software Engineering Conference. Proceedings.*, 2004, pp. 309-318: IEEE.

[18] P. Shanmugapriya and R. J. I. J. o. C. A. Suresh, "Software architecture evaluation methods-a survey," vol. 49, no. 16, 2012.

[19] B. Roy and T. N. Graham, "Methods for evaluating software architecture: A survey," *School of Computing TR*, vol. 545, p. 82, 2008.

[20] P. Kruchten, H. Obbink, and J. Stafford, "The past, present, and future for software architecture," *IEEE software*, vol. 23, no. 2, pp. 22-30, 2006.

[21] P. Bengtsson, N. Lassing, J. Bosch, and H. van Vliet, "Architecture-level modifiability analysis (ALMA)," *Journal of Systems Software*, vol. 69, no. 1-2, pp. 129-147, 2004.