Volume 5, Issue 5, May 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Evaluation of Software Architecture: Methods and Techniques Comparison (ATAM)

Prathamesh R. Deshpande, Snehal P. Jagtap
MCA Department, Mumbai University,
Maharashtra, India

Abstract: ***The software architecture has been keyed as an important part of a software system. The software architecture impacts the quality attributes of a system, e.g., performance and maintainability. So, methods for evaluating the quality attributes of software architectures are important. The objective of the evaluation is to assess whether or not the architecture will lead to the desired quality attributes. Advancing evaluation could reduced quality risk when software is designed. Recently, there have been a number of evaluation methods proposed. In this paper, we present a survey of some of the software architecture evaluation methods. We concentrate on methods for evaluating one or several of the quality attributes performance, maintainability, testability, and portability. We study some evaluation method in this paper. We mainly studied the Architectural Tradeoff Analysis Method (ATAM) method for the software architecture evaluation. We believe that ATAM capability of evaluating software security will provide potential benefits in secure software development.***

Keywords- Mobility Management, WiMAX, handover, Wi-Fi, Homogeneous and Heterogeneous networks.

I. INTRODUCTION

In this present net-centric age, The software engineering discipline is becoming more wide-spread in industry and organizations due to the increased presence of software and software-related products and services in all areas. Simultaneously, this demands for new concepts and innovations in the development of the software.

software architecture is a key asset for any organization that builds complex software- intensive systems. A software architecture is created early in the development and gives the developers a means to create a high level design for the system, making sure that all requirements that has to be fulfilled will be possible to implement in the system. Software Architecture plays a significant role in achieving system wide quality attributes, it is very important to evaluate a system's architecture with regard to desired quality requirements as early as possible. The definition for software architecture is given as "The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them."[1]. This means that the architecture describes which high level components a software system consists of as well as which responsibilities that these components have towards other components in the system. It also describes how these components are organized, both on a conceptual level as well as a decomposed detailed level since there can be an architectural structure inside components as well. Finally the architecture defines which interfaces the components present to other components and which interfaces and components that they use.

The principle objective of SA evaluation is to assess the potential of the chosen architecture to deliver a system capable of fulfilling required quality requirements and to identify any potential risks . Additionally, it is quicker and less expensive to detect and fix design errors during the initial stages of the software development. That is why an effective SA assessment method to analyze prospective architectures is of great business value. In order to help software developers make sure that software architecture will be able to fulfill the quality requirements; several methods for evaluating software architectures have been proposed. We study the some of method in this paper given as, Architecture Analysis Method (SAAM), Architecture Tradeoff Analysis Method (ATAM), Active Reviews for Intermediate Design (ARID), SAAM for Evolution and Reusability (SAAMER), Architecture-Level Modifiability Analysis (ALMA) , Architecture-Level Prediction of Software Maintenance (ALPSM), Scenario-Based Architecture Reengineering (SBAR), SAAM for Complex Scenarios (SAAMCS), and integrating SAAM in domain-Centric and Reuse-based development (ISAAMCR).

II. EVALUTION OF SOFTWARE ARCHITECTURE

Architecture evaluations can be performed in one or more stages of the software development process. They can be used to compare and identify strengths and weaknesses in different architecture alternatives during the early design stages. They can also be used for evaluation of existing systems before future maintenance or enhancement of the system as well as for identifying architectural drift and erosion.

A. Evaluation Categories

Software architecture evaluation methods can be divided into four main categories, i.e., experience-based, simulation-based, mathematical modeling based. Methods in the categories can be used independently but also be combined to evaluate different aspects of software architecture, if needed.

Experience-based evaluations are based on the previous experience and domain knowledge of developers or consultants . People who have encountered the requirements and domain of the software system before can based on the previous experience say if a software architecture will be good enough.

Simulation-based evaluations rely on a high level implementation of some or all of the components in the software architecture. The simulation can then be used to evaluate quality requirements such as performance and correctness of the architecture. Simulation can also be combined with prototyping, thus prototypes of an architecture can be executed in the intended context of the completed system.

Mathematical modelling uses mathematical proofs and methods for evaluating mainly operational quality requirements such as performance and reliability of the components in the architecture. Mathematical modeling can be combined with simulation to more accurately estimate performance of components in a system.

Scenario-based architecture evaluation tries to evaluate a particular quality attribute by creating a scenario profile that forces a very concrete description of the quality requirement. The scenarios from the profile are then used to step through the software architecture and the consequences of the scenario are documented. Several scenario based evaluation methods have been developed, e.g., Software Architecture Analysis Method (SAAM), Architecture Trade-off Analysis Method (ATAM), and Architecture Level Modifiability Analysis (ALMA).

B. Quality Attributes

A quality attribute can be defined as a property of a software system. A quality requirement is a requirement that is placed on a software system by a stakeholder; a quality attribute is what the system actually presents once it has been implemented. During the development of the architecture it is therefore important to validate that the architecture has the required quality attributes, this is usually done using one or more architecture evaluations. We address some quality attribute given as bellows:

Maintainability is a multifaceted quality requirement. It incorporates aspects such as readability and understandability of the source code. Maintainability is also concerned with testability to some extent as the system has to be re-validated during the maintenance.

Testability "The degree to which a system or component facilitates the establishment of test criteria and the performance of tests to determine whether those criteria have been met." We interpret this as the effort needed to validate the system against the requirements. A system with high testability can be validated quickly.

Performance "The degree to which a system or component accomplishes its designated functions within given constraints, such as speed, accuracy, or memory usage." There are many aspects of performance, e.g., latency, throughput, and capacity.

Portability "The ease with which a system or component can be transferred from one hardware or software environment to another." We interpret this as portability not only between different hardware platforms and operating systems, but also between different virtual machines and versions of frameworks.

These four quality attributes are selected, not only for their importance for software developing organizations in general, but also for their relevance for organizations developing software in the real-time system domain in a cost effective way, e.g., by using a product-line approach. Performance is important since a system must fulfill the performance requirements, if not, the system will be of limited use, or not used. The long-term focus forces the system to be maintainable and testable, it also makes portability important since the technical development on computer hardware technology moves quickly and it is not always the case that the initial hardware is available after a number of years.

III. A RCHITECTURE EVALUTION METHODS

In this paper each of the software architecture evaluation methods will be described according to a pre-defined template. The template structures the description of the architecture according to the following elements: Name and abbreviation, Category of method, Reference where the method are described in detail, Short description of the method, Evaluation goal of the method, How many quality attributes the method addresses, What specific quality attributes the method address and finally, the usage of the method.

SAAM — SOFTWARE ARCHITECTURE ANALYSIS METHOD

Software Architecture Analysis Method (SAAM) [2] is a scenario-based software architecture evaluation method, targeted for evaluating a single architecture or making several architectures comparable using metrics such as coupling between architecture components. SAAM was originally focused on comparing modifiability of different software architectures in an organization's domain. It has since then evolved to a structured method for scenario-based software architecture evaluation. Several quality attributes can be addressed, depending on the type of scenarios that are created during the evaluation process.

The method consists of five steps. It starts with the documentation of the architecture in a way that all participants of the evaluation can understand. Scenarios are then developed

that describe the intended use of the system. The scenarios should represent all stakeholders that will use the system. The scenarios are then evaluated and a set of scenarios that represents the aspect that we want to evaluate is selected. Interacting scenarios are then identified as a measure of the modularity of the architecture. The scenarios are then ordered according to priority, and their expected impact on the architecture.

ATAM — ARCHITECTURE TRADE-OFF ANALYSIS METHOD

Architecture Trade-off Analysis Method (ATAM) [3] is a scenario-based software architecture evaluation method. The goals of the method are to evaluate an architecture- level design that considers multiple quality attributes and to gain insight as to whether the implementation of the architecture will meet its requirements. ATAM builds on SAAM and extends it to handle trade-offs between several quality attributes. The architecture evaluation is performed in six steps. The first one is to collect scenarios that operationalize the requirements for the system (both functional and quality requirements). The second step is to gather information regarding the constraints and environment of the system. This information is used to validate that the scenarios are relevant for the system. The third step is to describe the architecture using views that are relevant for the quality attributes that were identified in step one. Step four is to analyze the architecture with respect to the quality attributes. The quality attributes are evaluated one at a time. Step five is to identify sensitive points in the architecture, i.e., identifying those points that are affected by variations of the quality attributes. The sixth and final step is to identify and evaluate trade-off points, i.e., variation points that are common to two or more quality attributes.

ALMA — ARCHITECTURE-LEVEL MODIFIABILITY ANALYSIS

Architecture-Level Modifiability Analysis (ALMA) is a scenario-based software architecture evaluation method with the following characteristics: focus on modifiability, distinguish multiple analysis goals, make important assumptions explicit, and provide repeatable techniques for performing the steps. The goal of ALMA is to provide a structured approach for evaluating three aspects of the maintainability of software architectures, i.e., maintenance prediction, risk assessment, and software architecture comparison. ALMA is an evaluation method that follows SAAM in its organization. The method specifies five steps: 1. Determine the goal of the evaluation, 2. Describe the software architecture, 3. Elicit a relevant set of scenarios, 4. Evaluate the scenarios, and 5. Interpretation of the results and draw conclusions from them. The method provides more detailed descriptions of the steps involved in the process than SAAM does, and tries to make it easier to repeat evaluations and compare different architectures. It makes use of structural metrics and base the evaluation of the scenarios on quantification of the architecture.

CBAM – COST BENEFIT ANALYSIS METHOD

CBAM begins where ATAM leaves off; being an architecture-centric method for analyzing the costs, benefits and schedule implications of architectural decisions. CBAM also assess the level of uncertainty associated with these judgments, so as to provide a basis for an informed decision process with regards to architecture. Different form the former methods CBAM is bridging two domains in software development the architecting process and the economics of the organization. CBAM is adding the costs (and implicit budgets or money) as quality attributes, which need to be considered among the tradeoffs when a software system is going to be planned. SAAM and ATAM primarily considered the design decisions with respect to architectural quality attributes like modifiability, performance, availability, usability, and so on. CBAM is claiming that costs, benefits and risks are as important as the other quality attributes and they are relevant to be considered when the architectural decisions are being made.

EBAE — EMPIRICALLY-BASED ARCHITECTURE EVALUATION

The main goal was to evaluate the maintainability of the new system as compared to the previous version of the system. The paper outlines a process for empirically based software architecture evaluation. The paper defines and uses a number of architectural metrics that are used to evaluate and compare the architectures. The basic steps in the process are: select a perspective for the evaluation, define/select metrics, collect metrics, and evaluate/compare the architectures. In this study the evaluation perspective was to evaluate the maintainability, and the metrics were structure, size, and coupling. The evaluations were done in a late development stage, i.e., when the systems already were implemented. The software architecture was reverse engineered using source code metrics

ABAS — ATTRIBUTE-BASED ARCHITECTURAL STYLES

Attribute-Based Architectural Styles (ABASs) build on the concept of architectural styles and extend it by associating a reasoning framework with an architectural style. The method can be used to evaluate various quality attributes, e.g., performance or maintainability, and is thus not targeted at a specific set of quality attribute. The reasoning framework for an architectural style can be qualitative or quantitative, and are based on models for specific quality attributes. Thus, ABASs enable analysis of different quality aspects of software architectures based on ABASs. The method is general and several quality attributes can be analyzed concurrently, given that quality models are provided for the relevant quality attributes. One strength of ABASs is that they can be used also for architectural design.

IV. ATAM

In section III we studied some evaluation method. In this section we focus on Architecture-Level Modifiability Analysis (ATAM) method.

ATAM analyses how well software architecture satisfies particular quality goals. It also provides insight into quality attribute interdependencies – meaning how they trade-off against each other. ATAM is based on Software Architecture Analysis Method (SAAM).

A. Key Factors in ATAM Development

The system's external stakeholders were considering all software architectural changes as equally possible. Business and technical perspectives were not discussed at the same time during the architecting process. Stakeholders were not aware of architectural risks that may jeopardize long-term business goals. Lack of evaluation methods which consider the
impact of architectural decisions on the architectural quality requirements like availability, performance, security, modifiability, usability, time-to-market, etc.

B. Prerequisites and Inputs for ATAM

For successfully conducting an ATAM evaluation session, the practitioners of this method and the stakeholders involved must consider a set of initial prerequisites:

− The evaluators must understand the system architecture, recognize the architectural parameters, define their Implications with respect to the system quality attributes, and compare these implications against the requirements.

− Problem areas were so called "sensitivity points", "tradeoff points" and risks. These must be carefully identified. A sensitivity point is a collection of components in the architecture that are critical for achievement of a particular quality attribute. A trade-off point is a sensitivity point that is critical for the achievement of multiple quality attributes. Risks are a subset of sensitivity points that may inhibit the system from achieving its quality goals.

− ATAM is a context-based evaluation method in which quality attributes of the system must be understood. This can be achieved employing descriptive scenarios for evaluating the quality attributes. An ATAM evaluation session uses as input the initial requirements of the system and the software architecture description of the system. Operational wise, ATAM can use templates, written rules, and other supporting materials for structuring the presentations of the system architecture and scenario generation.

C. Steps in an ATAM Evaluation Session

ATAM method consists of four phases: presentation, investigation and analysis, testing, and reporting. Each phase is a collection of steps. The presentation phase involves exchanging information through presentations. The investigation and analysis phase concerns the assessment of the key quality attribute requirements versus the architectural approaches. The testing phase compares the results of the previous phase to the needs of the relevant stakeholders.
Finally, the reporting phase summarizes the ATAM results.

The following subsections present ATAM steps in detail.

ATAM Presentation Phase

ATAM Step 1 – Present ATAM
Initially, the evaluation group leader describes ATAM to the participants. He tries to set their expectations and answer questions they may have.

ATAM Step 2 – Present Business Drivers A project spokesperson describes what business goals are motivating the development effort and hence the primary architectural quality drivers.

ATAM Step 3 – Present Architecture In this step, the architect describes the system's software architecture, focusing on how it addresses the business drivers set in the previous step.

Investigation & Analysis Phase

ATAM Step 4 – Identify Architectural Approaches In the step four the architect identifies architectural approaches, but they are not analyzed yet.

ATAM Step 5 – Generate Quality Attribute Utility Tree The quality attributes that comprise the system "utility" are elicited, specified down to the level of scenarios, annotated with stimuli and response, and prioritized.

ATAM Step 6 – Analyze Architectural Approaches Based on high-priority scenarios identified in the previous step, the architectural approaches that address those scenarios are elicited and analyzed. During this step, potential risks, possible non-risks, sensitivity points and trade-off points are identified.

Testing Phase

ATAM Step 7 – Brainstorm and prioritize scenarios. During a brainstorm session stakeholders provide a large group of scenarios. ATAM team together with the stakeholders prioritizes these scenarios by voting.

ATAM Step 8 – Reanalyze Architectural Approaches. The prioritized scenarios from the previous step are used as input for reiterations of step six. This set of scenarios is the most important one. The aim is to identify and document any other architectural approaches, risks, non-risks, sensitivity points, and trade- off points.

Reporting Phase

ATAM Step 9 - Present Results In the last step, based on the information collected during the first three phases of the ATAM session, the evaluation team summarizes and presents back the findings to the stakeholders. First the steps performed are reiterated together with the information collected in each step. What is finally relevant are the findings: the documented architectural styles, the final set of scenarios and their prioritization, the qualities' utility tree and the risks, non-risks, sensitivity points and tradeoff points identified. However is out of the ATAM scope to offer ways to solve the above-mentioned findings. Based on the evaluators' experience there can be identified mitigation strategies as well for the architectural risks, but this is not mandatory. .

D. ATAM Roles

The participation roles may be categorized as follows: external stakeholders, internal stakeholders, and the ATAM evaluation team.

a. External stakeholders are not directly involved in the software architecture development process. During the ATAM session their role is to present the business context of the project, based on the initial requirements to provide scenarios. Also they have decided which tradeoffs are appropriate at the end of the evaluation session together with the evaluation results presentation. Examples of external stakeholders are customers, project management, end users, system administrators, sponsors, etc.

b. Internal stakeholders are directly involved in the software architecture development process. Their role is to analyze, define and implement the architecture. During the ATAM evaluation session they are also responsible for describing, presenting and assessing (together with the ATAM team) the software architecture. Examples of the internal stakeholders are architects, design team leaders, testers, and integrators.

c. The ATAM team should be external to the development team for neutrality reasons. The ATAM team has no direct stake in the system software architecture; it is invited to conduct the evaluation session. ATAM team has also the leading role in proceeding with the evaluation, recording the intermediate assessment artifacts, and presenting the final results. Prior to that, if necessary, ATAM team must be able to support the stakeholders and the architects in generating the scenarios and presenting the software architecture, respectively. The ATAM evaluation team usually consists of a team leader or spokesperson, architecture analyst(s), and secretary.

E. ATAM Outcomes and Strengths

The general strengths of an ATAM session are:

− Stakeholders understand more clearly the architecture.

− Improved software architecture documentation. In some cases the architecture documentation must be recreated.

− Enhanced communication among the stakeholders. In terms of practical outcome ATAM delivers:

− Quality scenarios produced by stakeholders based on the quality attributes requirements.

− Architecture elicitation results based on quality scenarios and use cases.

− Quality attributes taxonomies, which provide evaluators with a catalogue of architectural parameters and appropriate stimuli for tracing different quality attributes and their interdependencies.

F. DISCUSSIONS AND CONCLUSIONS

In this paper have been presented five existing techniques forassessing the quality attributes of software architectures. Given the redundancy in the provided information, in the table below there will be summarized the main features of each method. For each of the presented techniques there are common points, weaknesses or strengths. In the Appendix, Table A, we summarize the relevant aspects of all the different methods. In the same time, we identified a set of general remarks applicable for all of the scenario-based evaluation techniques.

These issues are presented below:

- The methods' outcomes are highly culture dependent
- The methods lack the means to decide upon the accuracy of the modifiability estimates and the completeness of the risk assessment.

During the evaluation process the existing/proposed architecture(s) will be always surprised in their incapacity of accommodating the new generated scenarios since they haven't been considered in the design phase. Thus is highly predictable that there will be a lot of changes and open points that an architect must further reconsider.

− The evaluation team relies on stakeholders' ability to write true, feasible and useful scenarios, which can turn to give lots of trouble, being not the best possible option.

− In general the scenario prioritization process is highly influenced by the majority of a certain group of stakeholders as long as the voting procedure is used. Thus the prioritization process is very culture dependent as well.

- Scenario-based evaluation methods help stakeholders in identifying and expressing future changes-cases of the system. They actually contributed to the improvement of the interaction between stakeholders and architects during the system's development process.

- Most of the times the scenario-based evaluation techniques, make explicit the architecture-rationales for the systems stakeholders by proposing dialogue between different parties involved in the creation process.
- The methods were developed for enabling software architects in reasoning about the systems' quality aspects earlier in the development process.

The classical approach in evaluating software quality is "conformance to requirements"; the scenario-based evaluation methods are shifting the focus of the analysis towards estimating risks and uncertainty associated with the systems' requirements, architectural decisions and strategies. Scenario-based assessment techniques may coexist with the classical approaches. The innovation with these new approaches resigns in explicitly associate quality requirements with scenarios and validating them. For the time being scenario-based assessment methods are easy to comprehend and to apply. The effort in applying the methods is relatively low. Besides the outcomes of the different methods, some other benefits are: improved communications between stakeholders and meaningful architectural discoveries and improvements if the assessment is performed early in the development phase.

REFERENCES

[1] Muhammad Ali Babar, Liming Zhu, Ross Jeffery, *A Framework for Classifying and Comparing Software Architecture Evaluation Methods,* ISBN 1530-0803/04, 2004 IEEE.

[2] SHEN Qun-Li, LIU Jie, Two software Architecture Evaluation Method based on Scenario,978-1-4244-1734-6/08, 2008 IEEE

