A LITERATURE REVIEW AND RECOMMENDATIONS ON
SOFTWARE ARCHITECTURE EVALUATION

WAQAR UL HUSNAIN, SHAHBAZ AHMED
1Research Scholar, Department of Computer Science and Software Engineering, International Islamic
University Islamabad, PAKISTAN
2Asst. Professor, Department of Computer Science and Software Engineering, International Islamic
University Islamabad, PAKISTAN

E-mail: waqarulhk@gmail.com, shahbaz.ahmed@iiu.edu.pk

ABSTRACT

University Laboratory case studies are the crucial part for the success of the software engineers in real life
development of software in small and medium enterprises. Most of the case studies performed for the
evaluation of software processes and methods are for large scale software development. At University level
most students do not have access to such environments. Therefore, this paper first presents sorted list of
software architecture processes and then recommends those software architecture evaluation processes that
can better be implemented in university laboratory, so that individual software engineer who want to find
the impact of architecture evaluation on Risk, Cost and Schedule can integrate in his process and can
perform single case holistic design case study.

Keywords: Software Architecture, Architecture Evaluation, Risk, Cost, Schedule, Architecture Proposal

1- INTRODUCTION

Software system architecture design and
evaluation are the major phases in software
development lifecycle of mature and competent
organizations. The focus of such design and
evaluation of software architecture is mostly based
on evaluation of system quality attribute, business
goals/qualities, and architecture qualities. The
research literature provides evidence of number of
methods, techniques, models, and processes for
design and evaluation of software architecture. This
paper is only concerned with those that address only
architecture evaluation. This paper used the
guidelines for reviewing literature described in [19].

2- RELATED WORK

One of the available surveys on the software
architecture evaluation is given in [15]. The survey
presents the comprehensive coverage of research
literature from 1972 to 2002. There is another
framework for classifying and comparing software
architecture evaluation methods given in [16]. The
framework presents the comparison of various
evaluation methods from 1976 to 2004. The
extended form of this framework is given in [17].
The comparison of these two frameworks can be
found in [18], and this paper presents four
fundamental criteria for analyzing and evaluating
software architecture analysis methods. This paper
used the criteria and framework elements described
in these papers to select appropriate evaluation
method. Since large number of evaluation methods
have been proposed after the publication of surveys
such as [15] and [16] which requires the evaluation
and comparison of new methods. Also this review
will help in selecting appropriate tailorable [14]
architecture evaluation processes for individual
engineer to perform single-case holistic design case
studies.

3- RESEARCH STRATEGY

The research strategy consists of formation of
research question; selection of appropriate
keywords; formation of search string; selection of
appropriate search sources; and formation of
inclusion and exclusion criteria. All of these are
discussed one by one.

Research Question:
Which software architecture evaluation method
provides better support for Risk, Cost, Schedule,
and quality attribute Tradeoffs?

Keywords:
The keywords selected from research question are:
“Software”; “Architecture”; “Evaluation Methods”.

Search String:
The search string was formulated based on
keyword. Following search string were used.
1- Software Architecture Evaluation.
“Software” OR “Architecture” OR “Evaluation Method”.

Search Sources:
The search sources include digital libraries, publishers, search engines, and references/bibliography. These sources include: IEEExplore Digital Library; ACM Digital Library; Springer Verlag; Springer Science + Business Media; Kluwer Academic Publishers; Elsevier; Wiley InterScience; IEEE Computer Society; British Computer Society; Web of Science; ScienceDirect; CiteSeer; and Google.

Phase-1:
The reviewed article includes research papers from Journals, Conferences, Workshops, and Symposia. Copyright Technical Reports, Doctoral Dissertation with ISBN, ISSN or Copyright. However, White Papers, Technical Reports, Website Articles without copyright were not considered. Total selected publications for primary study using search string = 155

Phase-2:
Total selected research papers for primary study for detailed study = 155

Phase-3:
Total selected research papers before applying inclusion and exclusion criteria = 76

Phase-4:
Inclusion Criteria:
Object Oriented Methods; Generic Methods/Techniques; Architecture Evaluation; Multiple Quality Attributes; Scenario Based; Availability Evaluation; Modifiability Evaluation; Performance Evaluation; Security Evaluation; Usability Evaluation; Reusability Evaluation.

Exclusion Criteria:
Agent Oriented Methods; Aspect Oriented Methods; Domain Specific Methods; Architecture Design Methods; Architecture Evaluation (Software Product Line Engineering; Simulation Based Evaluation; Prototype Based Evaluation; Metrics Based Evaluation; Questionnaire Based Evaluation; Checklist Based Evaluation; Experiment Based Evaluation; Experience Based Evaluation; Value Based Evaluation) and Architecture Based Maintenance Evaluation. The total selected research papers after applying inclusion and exclusion criteria = 13

4- EVALUATION METHODS IN BRIEF

4.1- Software Architecture Analysis Method (SAAM):
Software Architecture Analysis Method (SAAM) [1] is a five-step method for analyzing software architectures. The steps or activities are listed below:

Activities of SAAM:
1- Characterization of canonical functional partition for a given domain.
2- Mapping of this partition over the structural decomposition of architecture.
3- Selection of Quality Attribute for architecture assessment.
4- Identify set of concrete tasks which test the selected quality attribute.
5- Evaluation of architecture for these tasks.

This method accompanied with an architectural description language which was used to analyze three competing architecture with respect to modifiability. This language describes the structural perspective of the competing architectures. It is a practical and proven method and applied for examining architectures of user interface portion of interactive system. It uses three perspectives for understanding and describing architectures which are Functionality, Structure, and Allocation. SAAM used these three perspectives of architecture description to evaluate system with respect to modifiability. Modifiability was the key quality attribute used in SAAM case study. However, SAAM can be used to evaluate other quality attributes as well.

4.2- The 4 + 1 View Model: 4+1VM
The Model [2] was developed to remedy the problem of software architecture representation using five concurrent views. These views include Logical View, Physical View, Development View and Scenarios. The model can be used to evaluate system quality attributes such as availability, reliability, scalability and portability.

The 4 + 1 View Model Process:
The process is evolutionary iterative development which involves prototyping, testing, measuring, and analysis. The analysis involves risk assessment and mitigation at the architecture level. The risk identification is performed using scenarios which are the instances of use case.

Step-1: In first step scenarios are selected based on risk and criticality, these risk may be technical risks. These critical but small in number scenarios are synthesize by abstracting user requirements. The next activity is to script the scenarios followed by decomposing them into sequences of object and operation pairs.

Step-2: In this step architectural elements are organized in to the four views followed by implementation, testing and measuring of the
architecture. The architecture analysis is performed at the end of these activities to reassess the risks and discovering any flaws.

Step-3: The analysis helps in identifying the additional architectural elements or changes. These changes are reflected using additional scenarios.

Step-4: The additional scenarios are implemented, tested, and measured for incremental architecture prototyping. Now by using five views reuse is considered.

Step-5: After analyzing reuse of existing architectural prototype for evolutionary development the final system architecture is produced.

4.3- Scenario-based Analysis of Software Architecture: SAAM-v2

Scenario-based Analysis of Software Architecture [3] shortly (SAAM-v2) is a structured scenario-based architectural analysis. It is the refined form of Software Architecture Analysis Method (SAAM). This version of SAAM has five activities explained below:

1- Describe The Candidate Architecture:
   This activity emphasizes the architecture analysis with well defined architectural description language that all stakeholders can understand.

2- Develop Scenarios:
   In this activity the scenarios of tasks for all software system stakeholders are developed.

3- Evaluate Each Scenario:
   In this activity Direct and Indirect Scenarios are identified with the associated cost and effort.

4- Reveal Scenario Interaction:
   In this activity scenarios with high interaction are separated from low interaction.

5- Weight Scenarios and Scenario Interactions:
   In this activity scenarios are weighted and ranked subjectively based on the relative importance of these scenarios with respect to all stakeholders.

4.4- Software Architecture Analysis Method for Evolution and Reusability: SAAEMER

Software Architecture Analysis for Evolution and Reusability [4] (shortly as SAAEMER) is a framework and a set of architectural views designed to evaluate software architecture for evolution and reuse. It is based on SAAM which is a scenario-based approach for software evaluation. This framework consists of four phases described below.

Gathering
   In this phase four different set of information is gathered which includes Stakeholder Information, Architecture Information, Quality Information and Scenarios.

Modeling
   Modeling is second phase of the framework in which information is aligned across information categories and mapped information into usable artifacts.

Analyzing
   Analyzing is the third phase of the framework in which SAAM is used for further analysis of various artifacts generated in the last phase.

Evaluating
   Evaluating is the fourth phase in which recommendations are made, risk and their mitigation strategies are suggested, and common reference models are identified. The framework presented in this paper can be used for better estimation of cost, schedule, and risk at early stage of software development for evolution and reusability.

4.5- Extended SAAM Method: E-SAAM

The extended SAAM method proposed in [5] was developed to overcome the limitations of SAAM for reuse of architecture knowledge in domain centric development processes. It considers reuse of architecture assets and reuse activities by integrating SAAM in domain centric process. The analysis template provides reusability of architecture knowledge in the domain-oriented development. These reuses of architecture assets help in reduce cost of development. Analysis template also facilitates the identification and mitigation of risk of neglecting crucial issues. It also helps in comparing different architectures in a domain. The analysis template together with architecture helps in determining the risky decisions for developing application that will be based on this architecture. The architecture specific analysis templates and domain specific analysis templates provides similar advantages such as reduction in cost and time of development.

4.6- Active Review for Intermediate Design: ARID

Active Review for Intermediate Design (ARID) [6] is a technical review which is a blend of four approaches: 1-stakeholder-centric; 2-scenario-based; 3-Active Design Review ; and 4-ATAM

This technical review is conducted after the architecture phase and before the start of design specification document. It is conducted for a portion of software architecture. It is one of the reason that such technical review conducted in prerelease stages are effective for discovering errors, inadequacies and inconsistencies in design.
There are several limitations of Active Design Reviews (ADR) and ATAM, e.g. ADR cannot be conducted if detailed documentation is not available, ATAM cannot be used to evaluate a portion of architecture, or for single quality attribute. To overcome these limitations ARID is developed. ARID adapts strong qualities of these techniques such as active participation from ADR and stakeholder-generated scenarios from ATAM.

4.7- The Architecture Tradeoff Analysis Method: ATAM
The ATAM [7] is a spiral model of architecture design, it is iterative in its nature and its each iteration is used to reduce risk that could result from competing quality attributes that a software system inherits. This method is designed to characterizing the interactions of these competing quality attributes and identifies the tradeoff points between such quality attributes. It is designed for general quality attributes such as performance, security, availability, reliability and so forth. This method has four phases in iterations. These phases are explained below:

Scenario & Requirements Gathering:
In this phase the scenarios are collected and requirements are gathered.

Architectural Views & Scenario Realization:
The various architectural views such as module view, process view, dataflow view and class view are used in this phase.

Model Building and Analysis:
In the phase the attribute specific analyses is analyzed in isolation and performed concurrent by individual quality attribute experts. In this step attribute models for quantitative and qualitative analysis is built.

Tradeoffs:
In this phase data regarding sensitivity points is collected along with location of tradeoff points.

4.8- Cost Benefit Analysis Method: CBAM
CBAM [8] is primarily designed to address the economics needs of an organization. This method is build upon ATAM [7]. This is why CBAM has many inputs from ATAM process and inherit many features and properties of this process, such as it design to address the risk and cost associated with the software system and the resulting benefits.

CBAM however, differ from ATAM in that it can be applied in maintenance and upgrade phases of the software development lifecycle. It presents an economic model of software that considers cost, risk and resulting benefits to the organization. This method focuses on three types of uncertainties of understanding and emphasize on elicitation and recording of these uncertainties. These uncertainties comes form:
1- Mapping architectural decisions onto quality attributes responses.
2- Mapping architectural decisions onto cost, and
3- Mapping quality attributes responses onto benefits.

4.9- 4+1 View Model Extension: 4+1VME
The method presented in [9] is an extended form of 4+1 View Model consisting of 9 major activities divided into three main phases.

Preparation:
In first phase of architecture evaluation three major steps are involved. First to define the template for requirements, second is to generate evaluation contract, and third is to define templates for software architecture.

Execution:
In the second phase of software architecture evaluation execution is performed. The execution is performed in four steps from step-4 through step-7. In the forth step architectural sub-designs are identified from functional requirements. In fifth step architectural design decisions are determined. In the sixth step rationale for these design decisions is determined. And in the final step relationship among these architectural decisions is finalized. The output of this step is the tradeoffs among quality attributes and dependencies among the decisions.

Finalize:
In the step-8 finalized decisions are grouped based on quality attributes. The step-9 is the last step in which prediction data is generated. The structure of prediction data is organized in four layers Qualities, Rationale, Architectural design decisions, and Sub-designs. This help in identification of risks and fitness of architecture with respect to quality attributes.

4.10- Software Architecture Review and Assessment: SARA
International Working Group on Software Architecture Review and Assessment (SARA W.G.) has developed a process for software architecture and system architecture review and assessment [10]. The process provides guidelines on steps to follow for review, question to ask from stakeholders, information to elicit, collect and document.

SARA provides a roadmap for finding and structuring Architecturally Significant Requirement (ASR) and finding Architecturally Significant Decision (ASD) for possible impact of these ASRs on ASDs. These reviews are conducted on concrete
architecture artifacts such as architectural description documents, business case, stakeholder's concerns, standards, and requirements. The objective of review may be for analyzing the conformance to specific standard, quality assessment of the architecture, architecture improvement and for effective communication between stakeholders.

**Review and Assessment Inputs:**
The inputs to SARA for review and assessment includes: Review Objectives; Review Scope; Architectural Artifacts; Architecturally Significant Requirements; Product Strategy and Product Planning; Requirements; Standards and Constraints; Quality Assurance Policies; Risk Assessment Artifacts;

**Review and Assessment Outputs:**
The SARA review outputs are: Assessment Report that includes: Objectives; Scope; Methodology; Evaluation Criteria for Architecture; Architectural Foundation and Approaches; Architecture Analysis, Findings and Recommendations; Executive Summary; Lesson Learned.

### 4.11- Software Architecture Comparison Analysis Method: SACAM

The Software Architecture Comparison Analysis Method (SACAM) [11] was developed to compare and analyze the software architectures. It is architecture centric, qualitative, goal-oriented analysis approach. This method is developed using scenario generation concept of Architecture Tradeoff Analysis Method (ATAM) and Quality Attribute Workshop (QAW). SACAM based on the assumption that the software architecture addresses best level of abstraction for organization’s business goals. The scenarios are used to compare the candidate architectures for any commonality or variability analysis of software product. However, it emphasizes on use of existing architecture documentation or architecture reconstruction techniques for analysis. The stakeholders score each scenario that leads to the selection of architecture. The technical reuse context in which SACAM was developed helps in reuse of architectural design. The goal of SACAM is achieved by two objectives i.e. the extraction of comparable architectural views, and criteria collation and analysis. The outputs of the process are summary report with recommendations and a presentation.

### 4.12- Architecture Centric Concern Analysis Method: ACCA

ACCA [12] method is analogous to Defect Causal Analysis (DCA) method for software defects and it uses root causes of concern in software architecture. The architectural concerns are triggered by architectural evaluation process. These architectural concerns are characterized by tradeoff points, sensitivity points, and risks at architecture stage. Hence, by knowing root causes of associated concern early in the requirement phase will help in mitigation of these risks, prevention and reduction of architectural problems. ACCA method uses a meta-model which is the extended form of Ramesh & Jarke’s Requirement Traceability Reference Model. Attribute Driven Design (ADD) is used in its Architectural Design stage and ATAM method is used in Architecture Evaluation stage. ACCA method is based on iterative process which helps in detection and prevention of architectural problems early in the requirement engineering phase. There by resulting in a high quality architecture, stakeholder satisfaction and low utilization of resource. This method is composed of eight processes.

### 4.13- Architecture Potential Analysis: APA

The purpose of this process [13] is to provide a concise architecture evaluation method. This method use quality attribute directed acyclic graph (QADAG) to uncover the dependencies of the quality attributes such as performance and modifiability for expected cost. QADAG basically is a structure of quality attributes. Any change in the architecture is calculated using uncovered dependencies. The method provides a systematic way of documentation of architecture knowledge and helps in traceability of architectural decisions. QADAG which is the hierarchical structuring of quality attributes is based on composite pattern. These quality attribute can be decomposed into sub-attributes or may consist of evaluation techniques. Quality Attribute Directed Acyclic Graph based evaluation structure is used to set dependencies into relation and for determining optimal potential. The method facilitates the architecture evaluation process to be carried out either early in the design phases or for later to ensure quality. The dependencies help in determining relationship between cost, performance and modifiability based on target market and available technology. By considering these dependencies one can uncover the optimization potential and development tendencies.

### 5- OPEN RESEARCH PROBLEMS

Most of the evaluation method discussed and review are performed for large scale software projects with large software engineering team. However, only few methods and their case studies
exist at this time that provides support of architecture evaluation for individual engineer for very small and small scale projects. So there is a strong need for such evaluation method along with their case studies and experimental results.

6- MISSING IN LITERATURE

As discussed in above the most case studies performed to evaluate the architectural processes are done for large scale software development. Not a single case study was found that was performed for individual engineer at university level demonstrating the use of such processes and integrated personal software process. Therefore, there is strong need for single-case holistic design case studies and their results at university level so that student can use historical data and prebuilt process.

7- CONCLUSION

Since the review was conducted to select appropriate method for software architecture evaluation. The comparative analysis given in Table-1 reveals that ATAM/CBAM provides better support for evaluation of risk, cost, schedule and tradeoff. The scope of SARA is broad and provides integrated capabilities of SAAM, ATAM, RMA and 4+1 View Model. Organization may use any method/techniques within SARA. Another conclusion drawn for review is that ARID is best for analysis of portion of the architecture. Finally ATAM/CBAM can be implemented in university laboratory and can be tailored [14] for individual software engineer. Therefore, these processes can be integrated in personal software process for individual engineer.

8- FUTURE WORK

In future the focus of review will remain the same but to cover those areas which fall in exclusion criteria. It would also be tried to report results of single-case holistic design case studies for integrated process for individual engineer. This integration of evaluation methods will help in enhancing his knowledge, skills, and competencies. The integrated process will also help to better understand the impact of architecture design and evaluation on cost, schedule and risk for engineers that use personal software process at individual level.

REFERENCES


<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SAAM</td>
<td>[1]</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>P</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>4+1VM</td>
<td>[2]</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>SAAM-v2</td>
<td>[3]</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>P</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>SAAMER</td>
<td>[4]</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>P</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>E-SAAM</td>
<td>[5]</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>ARID</td>
<td>[6]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATAM</td>
<td>[7]</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>P</td>
<td>P</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>CBAM</td>
<td>[8]</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>4+1VME</td>
<td>[9]</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>SARA</td>
<td>[10]</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>SACAM</td>
<td>[11]</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>P</td>
<td>P</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>ACCA</td>
<td>[12]</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>P</td>
<td>P</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>APA</td>
<td>[13]</td>
<td>P</td>
<td>Y</td>
<td>Y</td>
<td>P</td>
<td>P</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

Yes = Y; No = N; Partially Address = P