3.1 PREAMBLE

A detailed analysis based on comparison of the popular architecture evaluation methods is presented in this chapter. The relative merits and demerits including complexity analysis are dealt with in detail (This chapter was written based on the publications done in the Proceedings of National Conference at KCET, Virudhunagar, 2007, pp. 99-101).

3.2 SOFTWARE ARCHITECTURE ANALYSIS METHOD (SAAM)

The SAAM review process is a technique that aims to aid the evaluation and understanding of software architectures, and to address quality concerns such as maintainability, portability, modularity, reusability, and so on (Kazman et al 1994). The main features of the SAAM review process are the use of a facilitator and a special review process that is driven by scenarios. The participants in a SAAM review identify and select, scenarios to drive discussions about the software architecture (Dobrica et al 2002).

The main activities involved in the SAAM are enumerated below:

- Map the functional partitioning onto the architecture’s structural decomposition.
• Choose a set of quality attributes with which to assess the architecture.

• Choose a set of concrete tasks which test the desired quality attributes.

• Evaluate the degree to which each architecture provides support for each task (Kazman et al 1994).

SAAM has the following characteristics.

• **The Evaluation Technique:** Scenarios

• **Quality Attributes:** Modifiability

• **Stakeholders involvement:** All

• **The Phases of SA design:** In the final version of the SA.

• **When to stop generating scenarios:** If the addition of a new scenario no longer perturbs to design.

• **Scenario impact evaluation:** Relationships

• **Reusability of the existing knowledge:** Not considered.

In the next section we will discuss about SAAMCS.

### 3.3 **SAAM FOUNDED ON COMPLEX SCENARIOS (SAAMCS)**

SAAMCS considers that the complexity of scenarios is the most important factor for risk assessment. SAAMCS contributions for extending SAAM are, on the one hand, directed to the way of looking for the scenarios and, on the other, to where their impact is evaluated (Dobrica et al 2002; Maranzano et al 2005).
SAAMCS has the following characteristics:

- **The Evaluation Technique**: Scenarios
- **Quality Attributes**: Flexibility
- **Stakeholders involvement**: All
- **The Phases of SA design**: In the final version of the SA.
- **When to stop generating scenarios**: Defines a framework to discover all the complicated scenarios.
- **Scenario impact evaluation**: Relationship, owners, and version
- **Reusability of the existing knowledge**: Not considered.

In the next section we will discuss about ESAAMI.

### 3.4 EXTENDED SAAM BY INTEGRATION IN THE DOMAIN (ESAAMI)

ESAAMI is a combination of analytical and reuse concepts, and is achieved by integrating SAAM in the domain-specific and reuse-based development process. The degree of reuse is improved by concentrating on the domain. ESAAMI is similar to SAAM with regard to the evaluation technique, the quality attributes, the stakeholder's involvement, and SA description. However, an improvement is seen in the reuse of domain knowledge defined by SAs and analysis templates (Dobrica et al 2002).

ESAAMI has the following characteristics:

- **The Evaluation Technique**: Scenarios
- **Quality Attributes**: Similar to SAAM
• **Stakeholders involvement**: All

• **The Phases of SA design**: In the final version of the SA.

• **When to stop generating scenarios**: Similar to SAAM, but it considers proto scenarios, too

• **Scenario impact evaluation**: Similar to SAAM

• **Reusability of the existing knowledge**: Analysis template and reusable SAs in the domain.

In the next section we will discuss about SAAMER.

### 3.5 SAAM FOR EVOLUTION AND REUSABILITY (SAAMER)

SAAM is extended in SAAMER. SAAMER better suggests how a system could support each of the quality objectives or the risk levels for evolution or how to reuse it. Evolution and reusability are considered. Evolution integrates new quality objectives (maintainability and modifiability) obtained from domain experts.

SAAMER considers the following architectural views as critical: static, map, dynamic, and resource. The static view integrates and extends SAAM to address the classification and generalization of a system’s components and functions and the connections between components. These extensions facilitate the estimation of the cost or effort required for changes to be made. The dynamic view is appropriate for the evaluation of the behavior aspect, to validate the control and communication to be handled in an expected manner. The mapping between components and functions could reveal the cohesion and coupling aspects of a system.
SAAMER has the following characteristics:

- **The Evaluation Technique**: Scenarios
- **Quality Attributes**: Evaluation and reusability
- **Stakeholders involvement**: All
- **The Phases of SA design**: In the final version of the SA.
- **When to stop generating scenarios**: uses a practical two step-procedure
- **Scenario impact evaluation**: Estimate the cost required for a change.
- **Reusability of the existing knowledge**: Not considered.

In the next section we will discuss about ATAM.

### 3.6 ARCHITECTURE TRADE-OFF ANALYSIS METHOD (ATAM)

ATAM gets its name because it not only reveals how well an architecture satisfies particular quality goals (such as performance or modifiability), but it also provides an insight into how those quality goals interact with each other—how they trade-off against each other. Such design decisions are critical; they have the most far-reaching consequences and are the most difficult to change after a system has been implemented.

When evaluating an architecture using ATAM, the goal is to understand the consequences of architectural decisions with respect to the quality attribute requirements of the system. This method has also been used to analyze legacy systems. This frequently occurs when the legacy system needs to support major modifications, integration with other systems, porting,
or other significant upgrades. Assuming that an accurate architecture of the legacy system is available (which frequently must be acquired and verified using architecture extraction and conformance testing methods (Kazman 1999), applying the ATAM, results in increased understanding of the quality attributes of the system.

The ATAM draws its inspiration and techniques from three areas: the notion of architectural styles; the quality attribute analysis communities; and the Software Architecture Analysis Method (SAAM) (Kazman 1994), which was the predecessor of the ATAM. The ATAM is intended for the analysis of an architecture with respect to its quality attributes. Although this is the ATAM’s focus, there is a problem in operationalizing this focus. The software engineering community in general does not understand quality attributes well: what it means to be “open” or “interoperable” or “secure” or “high performance” changes from system to system, from stakeholder to stakeholder, and from community to community.

The ATAM is meant to be a risk identification method, a means of detecting areas of potential risk within the architecture of a complex software intensive system. This has several implications:

- The ATAM can be done early in the software development life cycle.
- It can be done relatively inexpensively and quickly (because it assesses architectural design artifacts).
- The ATAM will produce analyses commensurate with the level of detail of the architectural specification. Furthermore, it need not produce detailed analyses of any measurable quality attribute of a system (such as latency or mean time to
failure) to be successful. Instead, success is achieved by identifying trends.

The ATAM is an analysis method organized around the idea that architectural styles are the main determiners of architectural quality attributes. The method focuses on the identification of business goals which lead to quality attribute goals (Rick Kazman et al 2000).

ATAM has the following characteristics:

- **The Evaluation Technique**: Integrates existent questioning and measuring techniques
- **Quality Attributes**: Multiple quality attributes
- **Stakeholders involvement**: All or designer only
- **The Phases of SA design**: In the final version of the combined with the SA design into an iterative improvement process.
- **When to stop generating scenarios**: Uses a standard set of quality attribute-specific question.
- **Scenario impact evaluation**: Similar to SAAM when applied.
- **Reusability of the existing knowledge**: A set of pre-packages analyses and questions with known users.

In the next section we will discuss about SBAR.
3.7 SCENARIO-BASED ARCHITECTURE REENGINEERING (SBAR)

SBAR focuses on multiple software qualities. A number of quality attribute research communities have proposed their own methods for developing real-time, high performance, and reusable systems. All these methods focus on a single quality attribute and treat all others as having secondary importance, if at all any. SBAR considers these approaches unsatisfactory because a balance of various quality attributes is needed in the design of any realistic system (Dobrica et al 2002).

The contribution of this method is not only in the architecture design but also in the scenario-based evaluation of the software qualities of the detailed architecture of a system. SBAR estimates the potential of the designed architecture to reach the software quality requirements. Four different techniques for assessing quality attributes are identified: scenarios, simulation, mathematical modeling, and experience-based reasoning.

SBAR has the following characteristics:

- **The Evaluation Technique**: Depends on the attribute: scenarios, Mathematical modeling, simulators, objective reasoning.
- **Quality Attributes**: Multiple quality attributes
- **Stakeholders involvement**: Designer
- **The Phases of SA design**: Combined with the SA design into an iterative important process and re-engineering.
- **When to stop generating scenarios**: Defines a complete set or a representative set of scenarios.
- **Scenario impact evaluation**: Optimized or fulfilled.
- **Reusability of the existing knowledge**: Not considered.

In the next section we will discuss about ALPSM.

### 3.8 ARCHITECTURE LEVEL PREDICTION OF SOFTWARE MAINTENANCE (ALPSM)

ALPSM defines a maintenance profile, like a set of change scenarios representing perfective and adaptive maintenance tasks. A scenario describes an action or a sequence of actions that might occur as related to the system. Hence, a change scenario describes a certain maintenance task.

The method has a number of inputs: the requirements statement, the description of the architecture, expertise from software engineers and possibly historical maintenance data. ALPSM defines a maintenance profile, like a set of change scenarios representing perfective and adaptive maintenance tasks (Dobrica et al. 2002).

ALPSM consists of the following six steps:

- The identification of categories of maintenance tasks,
- Synthesis scenarios,
- Assignation of a weight to each scenario,
- Estimation of the size of all elements,
- Scripting the scenarios, and
- Calculation of the predicted maintenance effort.

The first step formulates classes of expected changes based on the application or program description; then, for each of the maintenance tasks, a
A representative set of scenarios is defined. The scenarios are assigned a weight based on the probability of their occurring during a particular time interval. To be able to assess the size of changes, the size of all components of the system is determined. The total maintenance effort is predicted by summing up the size of the impact of the scenarios multiplied by their probability. The size of the impact of each scenario realization is calculated by determining the components that are affected and to what extent they will be changed.

ALPSM has the following characteristics:

- **The Evaluation Technique**: Scenarios.
- **Quality Attributes**: Maintainability
- **Stakeholders involvement**: Designer
- **The Phases of SA design**: During design to predict adaptive perfective software maintenance.
- **When to stop generating scenarios**: Defines a set of scenarios for each expected maintenance task.
- **Scenario impact evaluation**: Estimate the size of the components and the extent to which they are affected.
- **Reusability of the existing knowledge**: Not considered.

In the next section we will discuss about SAEM.

### 3.9 A SOFTWARE ARCHITECTURE EVALUATION MODEL (SAEM)

A quality model based on standard software quality assessment process is chosen and a conceptual framework that relates quality requirements, metrics, and internal attributes of the SA and the final system is
proposed. The elements required for the quality evaluation of a software system, based on standard specification, are a quality model, a method for evaluation, metrics, and the supporting tools (Briand et al 1993).

SAEM has the following characteristics:

- **The Evaluation Technique**: Different metrics based on GQM technique.
- **Quality Attributes**: A quality model
- **Stakeholders involvement**: Not applied
- **The Phases of SA design**: In the final version of the SA
- **When to stop generating scenarios**: Not applied
- **Scenario impact evaluation**: Not applied
- **Reusability of the existing knowledge**: Not applied

The comparison analysis of all the above methods is discussed in the next section.

### 3.10 COMPARISON OF VARIOUS ANALYSIS METHODS

**Classification**

- Purely scenario-based: **SAAM**.
- Scenario and attribute model-based: **ATAM**.
- Techniques depending on attribute: **SBAR**.
- Techniques related to Metrics: **SAEM**.
**Stakeholders’ involvements**

- SAAM and ATAM need cooperation from stakeholders.
- SBAR: without stakeholder, but a detailed design needed.

**When to stop generating**

- **SAAM**: complete scenario = addition no longer disturbs the design.
- **SBAR**: representative set, but lastly based on engineer’s creativity and subjectivity.
- **SAAMER**: QFD (Quality function deployment, 1993, R. Day) technique used.

**Evaluation**

- **SAAM**: modification cost is associated with indirect scenarios.
- **ALPSM**: estimating component size needed historical maintenance data.
- **ATAM**: focuses on finding trade-off.

A comparison of the various architecture analysis methods is shown in the following Tables 3.1.
Table 3.1 Comparison of the analysis methods

<table>
<thead>
<tr>
<th>PARAMETERS</th>
<th>SAAM</th>
<th>SAAMCS</th>
<th>ESAAMI</th>
<th>SAAMER</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Evaluation Method</td>
<td>Based on Scenarios.</td>
<td>Based on Scenarios</td>
<td>Based on Scenarios</td>
<td>Based on Scenarios</td>
</tr>
<tr>
<td>Quality Attribute</td>
<td>Modifiability.</td>
<td>Flexibility.</td>
<td>Modifiability</td>
<td>Continuous Evaluation and object reusability</td>
</tr>
<tr>
<td>Design Phase</td>
<td>Final Version</td>
<td>Final Version</td>
<td>Final Version</td>
<td>Final Version</td>
</tr>
<tr>
<td>Termination Condition for generating Scenarios</td>
<td>The addition of the new scenario does not disrupt the steady state achieved</td>
<td>An Efficient scheme is required to elucidate all complex scenarios</td>
<td>Scenario for steady state along with proto scenario</td>
<td>It uses a practical two step protocol.</td>
</tr>
<tr>
<td>Evaluation of the impact of the scenario</td>
<td>Scenario will influence relationships</td>
<td>Relationships, owners and versions.</td>
<td>Relationships</td>
<td>Cost Estimation for implementing changes to be considered.</td>
</tr>
<tr>
<td>Reusability</td>
<td>Not Applicable</td>
<td>Not Applicable</td>
<td>Analysis Template</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>Business focus</td>
<td>Not considered</td>
<td>Not considered</td>
<td>Not considered</td>
<td>Not considered</td>
</tr>
<tr>
<td>Software Complexity</td>
<td>Does not address complex systems</td>
<td>Addresses complex systems</td>
<td>Does not address complex systems</td>
<td>Does not address complex systems</td>
</tr>
<tr>
<td>Evaluation Approach</td>
<td>Subjective</td>
<td>Subjective</td>
<td>Subjective</td>
<td>Subjective</td>
</tr>
<tr>
<td>Stakeholders’ satisfaction</td>
<td>Below 50%</td>
<td>Below 50%</td>
<td>Below 50%</td>
<td>Below 50%</td>
</tr>
<tr>
<td>Criteria</td>
<td>ATAM</td>
<td>SBAR</td>
<td>ALPSM</td>
<td>SAEM</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-------------------------------------------</td>
<td>-------------------------------------------</td>
<td>--------------------------------------------</td>
<td>-------------------------------------------</td>
</tr>
<tr>
<td>The Evaluation Method</td>
<td>Integrates currently existing questions and measuring methods.</td>
<td>Attribute scenarios collected by mathematical modeling, simulation and objective reasoning.</td>
<td>Scenarios</td>
<td>Metrics based on GQM Techniques.</td>
</tr>
<tr>
<td>Quality Attribute</td>
<td>A collection of related quality attributes</td>
<td>A collection of related quality attributes</td>
<td>Maintainability</td>
<td>Considers a quality model</td>
</tr>
<tr>
<td>Design Phase</td>
<td>An iterative process.</td>
<td>An Iterative process along with re engineering</td>
<td>Adaptive and preventive Maintenances</td>
<td>Final Version</td>
</tr>
<tr>
<td>Termination Condition for generating Scenarios</td>
<td>Based on a set of quality attributes specific issues</td>
<td>Defines an exhaustive set of representative scenarios</td>
<td>Defines a set of scenarios based on maintenances issues.</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>Evaluation of the impact of the scenario</td>
<td>Scenario will influence relationships</td>
<td>An optimized method based on constraints and conditions</td>
<td>Depends on the size and depth of applications of components</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>Reusability</td>
<td>Analysis method and known solutions</td>
<td>Not Applicable</td>
<td>Not Applicable</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>Business focus</td>
<td>Not considered</td>
<td>Not considered</td>
<td>Not considered</td>
<td>Not considered</td>
</tr>
<tr>
<td>Software complexity</td>
<td>Does not address complex systems</td>
<td>Does not address complex systems</td>
<td>Does not address complex systems</td>
<td>Does not address complex systems</td>
</tr>
<tr>
<td>Evaluation approach</td>
<td>Objective</td>
<td>Subjective</td>
<td>Subjective</td>
<td>Subjective</td>
</tr>
<tr>
<td>Stakeholders’ satisfaction</td>
<td>Around 80%</td>
<td>Below 50%</td>
<td>Below 50%</td>
<td>Below 50%</td>
</tr>
</tbody>
</table>
The selection of a suitable method depends on how well each comparison element fits into the problem context. It is not the purpose of this survey to suggest a ranking list of analysis methods to the practitioners, but to give an understanding of how the methods differ.

3.11 SUMMARY AND CONCLUSION

A detailed comparative analysis of popular architecture evaluation methods was done in this chapter. Attention was given to the relative merits and demerits of each of these methods. Among all these methods ATAM is the best when compared to all other methods in satisfying the stakeholders expectation. In the subsequent chapter we can discuss the features and limitations of ATAM.