CHAPTER 4

ENHANCED ATAM (ATAM AND DESIGN PATTERNS)

4.1 PREAMBLE

The ATAM method, the steps involved in the ATAM process, its features and drawbacks are presented in detail. The design pattern approach is also dealt with in detail. The chapter explains how the design patterns could be combined with ATAM method to derive the EATAM approach. The algorithm for this approach is also described in detail (This chapter was written based on the publications done in the WSEAS Conference, U.K, 2008, pp. 169-173 and CSS Journals, Malaysia, 2008, Vol. 2, No. 2, pp. 27-34).

4.2 ARCHITECTURE TRADE OFF ANALYSIS METHOD (ATAM)

In software engineering, the Architecture Tradeoff Analysis Method (ATAM) is a risk-mitigation process used early in the software development life cycle. ATAM was developed by the Software Engineering Institute at the Carnegie Mellon University. Its purpose is to choose a suitable architecture for a software system by discovering trade-offs and sensitivity points (Colquitt and Leaney 2002; Arnon Rotem-Gal-Oz 2006).

- Purpose: To assess the consequences of architectural decisions in the light of quality attribute requirements.
- Primarily a risk identification mechanism.
ATAM - Vocabulary

- **Scenario** – A scenario is a short statement describing an interaction of one of the stakeholders with the system.

- **Stakeholder** – An individual, team, or organization (or classes thereof) with interests in, or concerns relative to a system

- **Architectural view** – A representation of a whole system from the perspective of a related set of concerns

- **Functional requirements** - specify what software has to do.

- **Non-functional requirements** - specify how well it should be done.

4.2.1 The Evaluation Team

The Evaluation team usually consists of three to five people. Each member of the team is assigned a number of specific roles to play during the evaluation. This evaluation team may be a standing unit in which architecture evaluations are regularly performed (Kazman et al 2000).

4.2.2 Project Decision Makers

These people are empowered to speak for the development project or have the authority to mandate changes to it. They usually include the project manager, and if there is an identifiable customer who is footing the bill of the development, he or she will be present as well. The architect is always included; a cardinal rule of architecture evaluation is that an architect must willingly participate.
4.2.3 ATAM Process

The ATAM process consists of gathering stakeholders together to analyze business drivers and from these drivers extract quality attributes that are used to create scenarios. These scenarios are then used in conjunction with architectural approaches and architectural decisions to create an analysis of trade-offs, sensitivity points, and risks (or non-risks). This analysis can be converted to risk themes and their impacts whereupon the process can be repeated. ATAM – Cost/Benefit. The ATAM process diagram is shown in Figure 4.1 (Len Bass et al 2003; Nord et al 2003).

Figure 4.1 ATAM
4.2.4 ATAM Benefits

The following are some of the benefits of the ATAM process:

- Promotes the gathering of precise quality requirements.
- Creates an early start at architecture documentation.
- Creates a documented basis for architectural decisions.
- Promotes identification or risks early in the life-cycle.
- Encourages increased communication among stakeholders.
- Clarified quality attribute requirements.
- Improved architecture documentation.
- Documented basis for architectural decisions.
- Identified risks early in the life-cycle.
- Increased communication among stakeholders.

4.2.4.1 Risks, non-risks, sensitivity points and trade-off points

- Risks are potentially problematic architectural decisions.
- Non-risks are good decisions that rely on assumptions that are frequently implicit in the architecture.
- A sensitivity point is the property of one or more components (and/or component relationships) that is critical for achieving a particular quality attribute response.
- A trade-off point is a property that affects more than one attribute and is a sensitivity point for more than one attribute (Koscho and Ries 2009).
4.2.5 **Steps of the ATAM Process**

1. Present ATAM - Present the concept of ATAM to the stakeholders, and answer any questions about the process.

2. Present Business Drivers - Everyone in the process presents and evaluates the business drivers for the system in question.

3. Present the Architecture - The architect presents the high level architecture to the team, with an 'appropriate level of detail'.

4. Identify Architectural Approaches - Different architectural approaches to the system are presented by the team, and discussed.

5. Generate Quality Attribute Utility Tree - Define the core business and technical requirements of the system, and map them to an appropriate architectural property. Present a scenario for this given requirement.

6. Analyze architectural approaches - Analyze each scenario, rating it by priority. The architecture is then evaluated against each scenario.

7. Brainstorm and prioritize scenarios - among the larger stakeholder group, present the current scenarios, and expand.

8. Analyze architectural approaches - Perform step 6 again with the added knowledge of the larger stakeholder community.

9. Present results - provide all documentation to the stakeholders shown in Table 4.1.
### Table 4.1 ATAM steps associated with stakeholder groups

<table>
<thead>
<tr>
<th>Step</th>
<th>Activity</th>
<th>Stakeholder Groups</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Present the ATAM</td>
<td>Evaluation team / Customer representatives / Architecture team</td>
</tr>
<tr>
<td>2</td>
<td>Present Business Drivers</td>
<td>Evaluation team / Customer representatives / Architecture team</td>
</tr>
<tr>
<td>3</td>
<td>Present Architecture</td>
<td>Evaluation team / Customer representatives / Architecture team</td>
</tr>
<tr>
<td>4</td>
<td>Identify Architectural Approaches</td>
<td>Evaluation team / Customer representatives / Architecture team</td>
</tr>
<tr>
<td>5</td>
<td>Generate Quality Attribute Utility tree</td>
<td>Evaluation team / Customer representatives / Architecture team</td>
</tr>
<tr>
<td>6</td>
<td>Analyze Architectural Approaches</td>
<td>Evaluation team / Customer representatives / Architecture team</td>
</tr>
<tr>
<td>7</td>
<td>Brainstorm and prioritize scenarios</td>
<td>All Stakeholders</td>
</tr>
<tr>
<td>8</td>
<td>Analyze Architectural Approaches</td>
<td>Evaluation team / Customer representatives / Architecture team</td>
</tr>
<tr>
<td>9</td>
<td>Present Results</td>
<td>All Stakeholders</td>
</tr>
</tbody>
</table>

#### 4.2.6 Producing ATAM’S Outputs

In Step 2 of the ATAM, the customer presents the business context for the problem being solved. The utility tree is used to translate the business
context first into quality attribute drivers and then into concrete scenarios that
represent each business driver. Each scenario is described in terms of a
stimulus and the desired response. These scenarios are prioritized in terms of
how important they are to the overall mission of the system and the perceived
risk in realizing them in the system. The highest priority scenarios will be
used in the analysis of the architecture.

In Step 3 the system architect presents the architecture to the
evaluation team. The architect is encouraged to present the architecture in
terms of the architectural approaches that are used to realize the most
important quality attribute goals. The evaluation team also searches for
architectural styles and approaches as the architecture is presented. The
approaches that address the highest priority scenarios will be the subject of
analysis during the remainder of the evaluation.

Attribute-specific models (such as queuing models, modifiability
models, and reliability models) as they apply to specific architectural styles,
are codified in ABASs. These models provide attribute-specific questions that
the evaluators employ to elicit the approaches used by the architect to achieve
the quality attribute requirements.

The components used to find risks, sensitivity points, and tradeoff
points, high-priority scenarios, attribute-specific questions (as guided and
framed by the attribute characterizations), and architectural approaches are
shown in Figure 4.2.
During an evaluation, the architect traces the scenarios through the subset of the architecture represented by the approaches. To do this the architect must identify the components and connectors involved in realizing the scenario.

As the architect identifies the relevant components, the evaluators ask attribute-specific questions. Some of the answers to these questions lead to the identification of sensitivity points, some of which will turn out to be risks and/or tradeoff points.

**Business track**

- Interview project leader
- Interview business representatives
- Prepare quality attribute tree
- Prepare scenario brainstorm

**Figure 4.2 Concept interactions**
Architecture track

- Interview architect
- Interview lead developers
- Prepare example architecture documentation
- Identify approaches

Business drivers and the software architecture are elicited from the project decision-makers. These are refined into scenarios and the architectural decisions made in support of each one. The analysis of scenarios and decisions results in the identification of risks, non-risks, sensitivity points, and tradeoff points in the architecture. Risks are synthesized into a set of risk themes, showing how each one threatens a business driver.

The most important results are improved architectures. The output of an ATAM is an out-brief presentation and a written report that includes the major findings of the evaluation (Shaw and Garlan 2008).

In short, ATAM outputs can be described as

- A concise presentation of the architecture
- Articulation of the business goals
- Quality requirements in terms of a collection of scenarios
- Mapping of architectural decisions to quality requirements
- A set of identified sensitivity and tradeoff points
- A set of risks and non-risks
- A set of risk themes
4.2.7 ATAM Drawbacks

Scenarios are used to elicit the quality goals and are subdivided into use cases that cover the expected uses of the system, growth scenarios that probe anticipated extensions to the systems, and exploratory scenarios that stress the limits of the system. As each scenario is traced through the architecture, elicitation questions are asked to draw out attribute specific information. Screening questions are also asked to guide or focus the elicitation process. A great deal of knowledge exists in the various quality attribute communities to help facilitate the elicitation process. These attribute taxonomies help ensure that all aspects of a system attribute are covered and offer a rationale for asking elicitation questions.

The outcome of elicitation should be the identification of sensitivity points and tradeoff points in the architecture. A sensitivity point is the property of one or more components that is critical for achieving a particular quality. A tradeoff point is a property that affects more than one attribute and is a sensitivity point for at least one attribute. Once these points have been identified, attribute-based architecture styles (ABASs) are used to reason about the architectural design choices in the system. ABASs are patterns of design and experiential evidence of how similar designs have been used in other systems. ABASs offer routine and predictable designs, a standard set of analysis questions, and a tight link between design and analysis. The analysis of each point using ABASs offers some insight as to the strength and weakness of the overall system architecture and provides the direction for refinements to the system, identifies architectural weaknesses early when they are the easiest to correct, and provides confidence in the architecture if no problems are uncovered.

ATAM is a very solid application of the common engineering practice of critical design reviews for large, complex, and unknown systems.
in software development. It identifies the critical system attributes that are most sensitive to design decisions and uses experiential evidence to evaluate the system architecture with respect to the critical attributes. The method is fairly easy and relatively inexpensive – 40 to 60 staff days for a medium sized project was the estimate given to perform ATAM. As mentioned, it detects critical design flaws early, when they are easiest to correct and provides confidence if the architecture is proven to be solid. Without seeing ATAM in action it is difficult to judge if it is actually effective; however, it seems very sound. Only a couple of drawbacks are immediately apparent. The first is that ATAM only addresses architecture, and does not assess whether the system is functionally correct. It does not claim to do this; in fact, the paper explicitly says that ATAM does not assess functionality, but if full confidence in the system is to be ensured, additional effort must be expended separately from ATAM to perform a proper code review. Also ATAM is largely dependent upon a good architectural representation to be effective. Finally, we conclude that the drawbacks of ATAM are summarized as,

- ATAM is easy to implement but in complex architecture it is not sure of the result.
- In large systems the achievement of quality attributes such as maintainability, reusability, extensibility, and scalability are not fulfilled.
- ATAM promoted stakeholder participation, yet did not fulfilled. Openly tackle how to address the impact of stakeholder diversity.
- Business view is not taken into account during the analysis in ATAM.

In the next section we will discuss about the patterns and its impact on software architecture (Tyree and Akerman 2005).
4.3 PATTERNS

Each pattern describes a problem which occurs over and over again in our environment and then describes how we can use this solution a million times over, without ever doing it the same way twice. An emerging idea in system development under the process can be improved significantly if the system can be analyzed, designed and built from prefabricated and predefined system components.

Originally, patterns addressed common idioms in the world of object-oriented design and implementation. Patterns can be used in other areas; in particular, analysis patterns are being used to describe common idioms at the business analysis level. The various components of patterns is explained in the next section.

4.3.1 Components of Patterns

- **Name**: It must have a meaningful name.

- **Problem**: A statement of the problem which describes its intent: the goals and objectives. It wants to reach within the given context and forces.

- **Context**: The preconditions under which the pattern is applicable.

- **Forces**: A description of the relevant forces and constraints and how they Interact / conflict with one another.

- **Solution**: Static relationships and dynamic rules describing how to realize the desired outcome.

- **Examples**: One or more sample applications of the pattern.
• **Resulting Context**: It describes the post conditions and Context side-effects of the pattern.

• **Rationale**: A justifying explanation of the steps or rules in the pattern.

• **Related Patterns**: The static and dynamic relationships between this pattern and others within the same pattern language or system.

• **Known Uses**: Describes the known occurrences of the pattern uses and its application.

In the subsequent section design patterns and its structure is discussed.

### 4.4 DESIGN PATTERNS

In software engineering, a design pattern is a general reusable solution to a commonly occurring problem in software design. A design pattern is not a finished design that can be transformed directly into a code. It is a description or template for how to solve a problem, that can be used in many different situations. Object-oriented design patterns typically show relationships and interactions between classes or objects, without specifying the final application of classes or objects that are involved.

At a higher level there are architectural patterns which are larger in scope, usually describing an overall pattern followed by an entire system.

Design patterns can speed up the development process by providing tested, proven development paradigms. Effective software design requires considering issues that may not become visible until later in the implementation. Reusing design patterns helps to prevent subtle issues that
can cause major problems, and it also improves code readability for coders and architects who are familiar with the patterns.

In order to achieve flexibility, design patterns usually introduce additional levels of indirection, which in some cases may complicate the resulting designs and hurt application performance.

In addition, patterns allow developers to converse by means of well-known, well understood names for software interactions. Common design patterns can be improved over time, making them further robust than unplanned designs.

4.4.1 Design Pattern Structure

Design patterns were originally grouped into the following categories: Creational patterns, Structural patterns, and Behavioral patterns; and they were described using the concepts of delegation, aggregation, and consultation. Another classification has also introduced the notion of architectural design pattern which may be applied at the architecture level of the software, such as the Model-View-Controller pattern.

To overcome the drawbacks of ATAM, the EATAM (Enhanced Architecture Trade-Off Analysis Method) is introduced in the next section along with design patterns.

4.5 ENHANCED ARCHITECTURE TRADE-OFF ANALYSIS METHOD (EATAM)

ATAM is a method for evaluating architecture-level designs; it identifies trade-off points between attributes, facilities communication between stakeholders (such as user, developer, customer, maintainer) from the perspective of each attribute, clarifies and refines requirements, and provides a framework for an ongoing, concurrent process of system and analysis.
ATAM is a risk identification mechanism of quality achievement. Normally, ATAM does not discuss all the possible quality attributes. The efficiency of ATAM depends on the expertise and potential of the stakeholders (SH) and quality attributes.

To overcome the drawbacks of ATAM a new model called EATAM (Enhanced Architecture Tradeoff Analysis Method) has been proposed. By combining ATAM process with the design patterns this new has been formed and it is shown in Figure 4.3.

![Figure 4.3 EATAM](image-url)
The evaluation of the software architecture, would result in better solutions when we use EATAM instead of ATAM and the procedure to use EATAM is discussed in the next section.

4.5.1 Procedure for Enhanced ATAM (EATAM)

The ATAM approach involves an iterative process consisting of the following steps. After each iteration the quality can be incrementally enhanced by applying design patterns.

1. Present ATAM - Present the concept of ATAM to the stakeholders, and answer any questions about the process.

2. Present Business Drivers - Everyone in the process presents and evaluates the business drivers for the system in question.

3. Present the Architecture - The architect presents the high level architecture to the team, with an appropriate level of detail.

4. Identify Architectural Approaches - Different architectural approaches to the system are presented by the team, and discussed.

5. Generate Quality Attribute Utility Tree - Define the core business and technical requirements of the system, and map them to an appropriate architectural property. Present a scenario for this given requirement.

6. Analyze architectural approaches - Analyze each scenario, rating it by priority. The architecture is then evaluated against each scenario.

7. Brainstorm and prioritize scenarios - among the larger stakeholder group, present the current scenarios, and expand.
4.5.1.1 Influence of design patterns on the ATAM process

The judicious and intelligent application of design patterns can result in a significant enhancement of software quality.

Creational Patterns can be used while evaluating the approaches for finalizing the architecture. This approach is essentially centered on three goals:

(a) Interface for creating families of objects
(b) Interface for creating an object
(c) Keep construction and representation of the object separate from each other.

Structural patterns can then be applied to further fine-tune the architecture. This can result in improved cohesion and overall flexibility.

After the analysis phase the design phase is taken up, where behavioral and concurrency design patterns can be applied effectively.

4.5.1.2 Results and Discussion

The EATAM method is applied to a case study (which is described in chapter 5) and the results are discussed.

The EATAM method is analyzed based on a quality attribute called performance. Performance is defined as the time required to respond to the number of events processed in some interval of time. For quantitative analysis, performance is interpreted in terms of the response time.
The various projects have been taken and the function points of the projects were assigned in the x axis and the response time in days on the y axis. The graph drawn and it shows the relationship between response time and function points for both ATAM and EATAM.

A comparison was done about the overall performance in the case of the two approaches. The time taken to develop the product using EATAM was slightly better than ATAM. The result is presented as shown in Figure 4.4.

![Graph showing the relationship between response time and function points for ATAM and EATAM](image)

**Figure 4.4 Performance analyses of ATAM and EATAM**

By applying the above mentioned design patterns iteratively, the efficiency of ATAM can be significantly improved, and this justifies the EATAM approach which is derived by integrating design patterns into the various ATAM phases. But the drawbacks identified in the ATAM method like maintainability, extensibility, scalability and reusability are not fulfilled to the entire satisfaction of stakeholders. To overcome these drawbacks innovative patterns are introduced in the next chapter.
4.6 SUMMARY AND CONCLUSION

The ATAM approach was presented in detail. Its merits and demerits were analyzed. The method of enhancing the ATAM approach by combining with design pattern was presented. The resulting EATAM approach was analyzed. Both ATAM and EATAM have drawbacks, the customized patterns called Innovative Patterns are introduced in the subsequent chapter.