Software quality attributes illustrate how the system is built and qualitative constraints on the functions of the system. These requirements deal with all the stakeholders of the software. Unlike functional requirements, quality requirements may vary from system to system, from stakeholder to stakeholder [98]. These quality attributes affect each other positively or negatively. A deep understanding of these quality attributes, their relation and finding the optimal balance among them is very essential to produce high quality software.

High quality software can be obtained from the high quality architecture. Design decisions involve with quality attributes differently. The conflicts among the quality attributes make the decision process tougher [99]. Hence capturing the right set of quality attributes as non functional requirements is very important. Software architect evaluates the design alternatives with respect to quality attributes and obtains the preferences of design alternatives. The optimal design alternative will be obtained based on the right balance of quality attributes for meeting the stakeholder’s requirements and to produce high quality software [99] [100] [101]. To evaluate the design decisions, architecture analysis methods and techniques are used by the software architect. Widely used architecture analysis techniques are presented in this chapter. The requirements elicitation, identification of design decisions and evaluating the design alternatives of these architecture analysis techniques are discussed here.
4.1 SOFTWARE ARCHITECTURE ANALYSIS METHOD (SAAM)

Software Architecture Analysis Method (SAAM) shows how the software architecture meets certain properties through a method that explains and analyzes software architecture. These properties are functionality, structure, and allocation. SAAM quickly assess many quality attributes such as modifiability, portability, extensibility and integrability. It can also assess quality aspects of software architectures such as performance or reliability.

The main objective of SAAM is architectural suitability and risk analysis [102]. The direct and indirect scenarios are identified as shown in fig 4.1. The impact analysis counts the affected components. The objects are analyzed in the logical views of Architectural documentation.

![Fig 4.1 SAAM steps](image-url)
The steps involved in the SAAM are:

1. Scenarios development
2. Architecture description
3. Scenarios classification and prioritization.
4. Indirect scenarios evaluation individually
5. Scenario Interaction assessment.
6. Overall evaluation of the interaction.

4.2 ARCHITECTURE TRADEOFF ANALYSIS METHOD (ATAM)

ATAM is a risk identification method that helps to understand the technical tradeoff and risks. The main aim of this method is to assess the consequences of architectural decision alternatives with respect to quality attributes. Scenarios are developed by the stakeholders to represent the quality attributes and their prioritization [103]. These scenarios are analyzed with respect to the various architectural approaches to list out the risks, sensitivity points and tradeoffs. The steps that are shown in fig 4.2 are:

1. Present ATAM
2. Present business drivers
3. Present architecture
4. Identify architectural styles
5. Generate quality attribute utility tree
6. Elicit and analyze architectural styles
7. Generate seed scenarios
8. Brainstorm and prioritize scenarios
9. Map scenarios onto styles
10. Present out-brief and/or write report
These methods don’t help to identify design decisions. Based on the stakeholders’ opinions and knowledge, the method evaluates the architecture’s support of quality attributes. And, based on the developer’s experiences and opinions, the design alternatives are compared. These methods don’t help the designers to realize the relationship between design decisions, even they provide tradeoff and sensitivity points. The method gives limited guidance on mapping the quality attributes to architectural elements and doesn’t give guidance on optimization of quality attributes.

4.3 COST BENEFIT ANALYSIS METHOD (CBAM)

The SEI Cost Benefit Analysis Method (CBAM) is a method for improvement of existing architectural design by performing economic analysis of the architecture. Based on the economic tradeoff and the return on investment, software architect decides the architecture alternatives. The analysis result of ATAM is given to the CBAM as shown in fig 4.3.
Then it relates priorities, cost and benefits with architecture decisions. The CBAM consists of following steps:

1. Scenarios and architecture strategies selection from ATAM output.
2. Refinement of scenarios
3. Prioritization of scenarios by voting.
4. Stakeholder should assign numbers to each quality attribute with the sum value of 100.

5. Quantifies the architecture strategies’ benefits by the Stakeholder’s rank on each architecture strategy in terms of its contribution to each quality attribute on a scale of -1 to +1. The benefit of each architecture strategy has been calculated [104].

6. Quantify the expected cost of implementing each architecture strategy that results in the expected benefit.

7. Calculate a desirability to compare the desired architecture strategies. The desirability is the unit benefit/cost.

8. Decision making based on these scores and their uncertainty.

9. Confirmation of results.

The method works on the quality attributes from ATAM outputs that are collected from stakeholder meetings and decides a set of desired improvements on them. The method chooses the intended quality attributes from the scenarios of ATAM. Assigning a number as a quality attribute score, proves the involvement of each stakeholder on quantifying the alternative’s benefit. The method doesn’t have the capability to model software architecture and quality attributes before the analysis [84]. Nowhere in the method is it provided how the identification of design decisions proceeds. The method quantifies priorities, costs and benefits, which are associated with architecture decisions as additional attributes to be considered during the software architecture maintenance phase. The method quantifies design alternatives’ benefits to each quality attribute, which in turn, provides tradeoff information on the relative ranking of how well each quality attribute is being supported by the design alternative. The CBAM is a decision framework. It provides no guidance on sensitivity analysis and mapping quality attributes to elements. It doesn’t consider the relationship among the decisions. It makes quality attribute optimization decisions.
4.4 WinCBAM

WinCBAM deals with the application of WinWin techniques with CBAM methods. This is a decision-support method that allows the negotiation of stakeholders on their conflict requirements. It helps to evaluate the software architecture alternatives systematically and iteratively until it reaches the agreement as shown in fig 4.4.

![Diagram of WinCBAM steps]

Fig 4.4 WinCBAM steps

The steps of CBAM are interleaved with the steps of the WinWin process. Initially, stakeholders enter their win conditions. Issue schema will be created when conflict among stakeholders’ win condition arises. The schema will have the conflict and win conditions that are involved with it. The conflict-resolution options are considered as architecture strategies by the stakeholders. As WinWin conditions have tradeoffs to be balanced, CBAM facilitates it. CBAM helps the WinWin process to evaluate and negotiate
architecture alternatives by extorting stakeholders’ benefits and costs [105].
Steps involving in WinCBAM are following:

1. Elicitation of Win Conditions
2. Identification of Conflict issues
3. Exploration of options and Architecture strategies
4. Calculation of benefits and costs using CBAM
5. Reaching agreements

The identification of quality attributes is based on the stakeholders’ decisions. It supports the unquantifiable quality attributes. The Win conditions represent the stakeholder’s measurement of quality attributes. The method starts from requirements phase, and during the requirement negotiation, the architecture development happens. That means it evaluates the architecture before it is developed. This method doesn’t have any formal format to obtain the requirements. WinCBAM gives no guidance on sensitivity analysis and on mapping quality attributes to architectural elements. It doesn’t consider the relationships among the design decisions. To quantify and compare the design alternatives, CBAM is used. The WinCBAM is not a decision-making tool. It is a decision-support tool that helps to focus the discussion of requirements by showing the stakeholders the implications of their requirements.

4.5 SAAM FOR EVOLUTION AND REUSABILITY (SAAMER)

SAAMER developed a modeling technique to verify whether the stakeholder objectives are addressed and traced. This makes use of SAAM. It consists of a framework and set of architectures views. The framework is used to model relevant information. Relevant information may be scenarios, quality information, stakeholder information and architecture information. They are arranged during modeling phase. Architectural views are used to analyze and compare software architectures. It analyses whether each stakeholder considered each quality attribute and the levels of abstraction of the
stakeholders’ objectives. During the analysis phase, SAAM is taken into account with architecture views. This framework consists of the following four phases.

1. Gathering four different sets of information
2. Information modeling
3. Analyze various artifacts generated in the last phase
4. Identification of common reference models after the evaluation.

As a number of scenarios are developed based on stakeholder and architecture objectives, the quality attributes are identified based on the stakeholders. For a deep understanding of quality attributes, domain experts are interviewed using a prepared elicitation question. It doesn’t help to reach a consensus when conflicts occur among different stakeholders’ opinions. The method assesses an existing architecture for reuse in a future project in the same problem domain or product line. Using architectural views and scenarios, architecture and quality attributes are modeled before the analysis. As SAAM is adopted in this method, the measurement of quality attributes is human based. It provides qualitative methods to compare design alternatives. It doesn’t have tradeoff analysis, sensitivity analysis and doesn’t consider the relationships among the design decisions. But, it highly supports the mapping of quality attributes to architectural elements [106].

4.6 SCENARIO-BASED SOFTWARE ARCHITECTURE REENGINEERING (SSAR)

SSAR aims for the assessment of quality attributes in software architectural level by the following approaches: Scenario-based evaluation, Simulation, Mathematical Modeling, and Experience-based reasoning. During the analysis, the best suitable approach for each quality attribute will be selected. Scenario-based evaluation is the widely used approach. The following steps are involved in this method.
1. Describe the candidate architecture
2. Scenarios development
3. Evaluation of each scenario
4. Reveal scenario interaction
5. Weight Scenarios and Scenario interactions

The quality attributes are identified based on the stakeholders’ requirements. It doesn’t help to reach the consensus in the midst of conflicts [107]. The aim of this method is to improve an architecture based on the quality attributes. Before the analysis, the architecture and quality attributes are modeled. It doesn’t lead to identify the design decisions. Scenario-based evaluation and experience-based reasoning use qualitative method, simulation and mathematical modeling use a quantitative method to compare design alternatives.

The deficiency is identified by mapping the unsatisfied quality attributes to architectural elements after the evaluation. It gives a very less guidance on mapping the quality attributes to architectural elements. It doesn’t detect the confliction among required quality attributes. It doesn’t support trade off and sensitivity analysis and doesn’t consider relationships among design decisions.

4.7 NON-FUNCTIONAL REQUIREMENT FRAMEWORK (NFR FRAMEWORK)

NFR Framework guides to select between architecture design alternatives. During the design process, Nonfunctional requirements are represented as goals, and their related knowledge is codified into methods and correlation rules [108]. The goals can be achieved and decomposed by the methods. The methods argument design decisions. To interpret the tradeoffs among design alternatives for the selection of alternatives, correlation rules are used. These rules help to find out goal conflicts. NFR’s may affect the design
alternative positively or negatively. This can be modeled in the goal graph. The consequence of each design decision moves from child to parent by the evaluation procedure, during the goal graph expansion process. NFR gives clarity in the tradeoff of design alternatives while considering multiple quality attributes simultaneously.

The quality attributes are collected, decomposed, and evaluated based on the stakeholders’ point of view. It supports the unquantifiable quality attributes. It doesn’t deal about reaching the consensus when conflicts arise, but, it is used to find and explore the conflicting goals. It should be used in the initial stage of the architecture design phase. This method doesn’t require any modeling. It doesn’t identify the design decisions. From the stakeholders’ opinions, the support for each quality attribute with architecture is measured. It uses qualitative methods to compare design alternatives. It somehow supports the tradeoff analysis, but doesn’t deal with the sensitivity analysis, relationship between design decisions and mapping quality attributes to architectural elements.

4.8 ANALYSIS OF ARCHITECTURE ANALYSIS METHODS

Stakeholders should come to the agreement on understanding the quality attributes to evaluate the design alternatives accurately. Selecting a particular design alternative impact more than one quality attributes positively or negatively. Stakeholders should also agree on the prioritization of quality attributes to resolve the conflicts by the designer. The methods discussed above are analyzed to know how they support the understanding and prioritization of quality attributes to resolve the conflicts and to reach the consensus. WinCBAM identifies issues/ conflicts among stakeholders by requirements negotiation, resolve them by listing design options and evaluating them to reach an agreement. During requirements determination phase, the designer should be available with the information about required
quality attributes to compare the alternatives. The actual meanings of the quality attributes vary from system to system and stakeholder to stakeholder because of their unquantifiable nature. Hence, the clear understanding of these quality attributes is necessary to compare the design alternatives meaningfully. It is very tough to compare the design alternatives amidst conflicting goals. Henceforth, the support to reach consensus should be provided by the analysis technique with the meaning of unquantifiable quality attributes and prioritization of goals.

The methods used, the quality attributes identified in Requirements Elicitation phase, to ensure whether all the stakeholders’ requirements are met in the architectural design [89] [90. The architecture design should be divided into a set of design decisions with respect to quality attributes by the designer. This will help stakeholders to conclude the best architectural solution that meets all the required quality attributes. Even though all methods listed above identify the quality attributes from stakeholders, it is unsure that whether they all have been considered in design decisions.

The design decisions are interrelated, because same quality attributes can be influenced by multiple design decisions in various ways. This may make changes in the decision also. The architecture decision-making technique should evaluate the architecture based on the inter-dependence relationships between design decisions and the mapping relationships between quality attributes and architectural elements.

SAAMER and SSAR support modeling mechanism to represent the architecture and quality attributes. SAAM provides modeling for architecture and ATAM provides modeling for quality attributes. As both the architecture and quality attributes play important role in decision making, the decision making technique should model both of them. To compare the unquantifiable quality attributes, the approaches may use different measurements. Only
SSAR supports qualitative and quantitative measurements of different quality attributes. Even though some level of tradeoff analysis has been done by all these approaches, they don’t provide all the required information. ATAM identifies tradeoff points but hands over the decision to requirements negotiation. Similarly it identifies sensitivity points but leaves the sensitivity analysis to designer.

Table 4.1 given below, listed out and compared various aspects of these analysis methods. This table clearly says that quality attributes identification and prioritization is heuristic based in all the methods discussed above.

Table 4.1 Comparison of Architecture Analysis methods

<table>
<thead>
<tr>
<th></th>
<th>SAAM</th>
<th>ATAM</th>
<th>CBAM</th>
<th>WINCB AM</th>
<th>SAAMER</th>
<th>SSAR</th>
<th>NFR Framework</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quality Attributes Identification</td>
<td>human based</td>
<td>human based</td>
<td>human based</td>
<td>human based</td>
<td>human based</td>
<td>human based</td>
<td>human based</td>
</tr>
<tr>
<td>Handling conflicts</td>
<td>No support</td>
<td>No support</td>
<td>Quantitative</td>
<td>No Support</td>
<td>No Support</td>
<td>No Support</td>
<td>No Support</td>
</tr>
<tr>
<td>Quality Attributes prioritization</td>
<td>human based</td>
<td>human based</td>
<td>human based</td>
<td>human based</td>
<td>human based</td>
<td>human based</td>
<td>human based</td>
</tr>
<tr>
<td>Architecture Evaluation</td>
<td>after development</td>
<td>after development</td>
<td>after development</td>
<td>before development</td>
<td>after development</td>
<td>after development</td>
<td>after development</td>
</tr>
<tr>
<td>Design decisions identification</td>
<td>No support</td>
<td>No support</td>
<td>No support</td>
<td>No support</td>
<td>No support</td>
<td>No support</td>
<td>No support</td>
</tr>
<tr>
<td>Guidance for Design decisions identification</td>
<td>Human based</td>
<td>Human based</td>
<td>Human based</td>
<td>Human based</td>
<td>Machine based</td>
<td>Human based</td>
<td>Human based</td>
</tr>
<tr>
<td>Comparisons of Design alternatives</td>
<td>Qualitative</td>
<td>Qualitative</td>
<td>Quantitative</td>
<td>Quantitative</td>
<td>Qualitative</td>
<td>Qualitative</td>
<td>Qualitative</td>
</tr>
<tr>
<td>Dynamic changes on quality attributes priority before implementation</td>
<td>No support</td>
<td>No support</td>
<td>No support</td>
<td>No support</td>
<td>No support</td>
<td>No support</td>
<td>No support</td>
</tr>
<tr>
<td>Support to reach consensus</td>
<td>Human based</td>
<td>Human based</td>
<td>Human based</td>
<td>Human based</td>
<td>Human based</td>
<td>Human based</td>
<td>Human based</td>
</tr>
<tr>
<td>support for tradeoff</td>
<td>No support</td>
<td>yes, but low</td>
<td>medium</td>
<td>medium</td>
<td>No support</td>
<td>No support</td>
<td>low</td>
</tr>
<tr>
<td>support for sensitivity analysis</td>
<td>yes</td>
<td>yes</td>
<td>No support</td>
<td>No support</td>
<td>No support</td>
<td>No support</td>
<td>No support</td>
</tr>
<tr>
<td>Quality attributes optimization</td>
<td>No support</td>
<td>No support</td>
<td>Support</td>
<td>Support</td>
<td>Support</td>
<td>No Support</td>
<td>No Support</td>
</tr>
</tbody>
</table>

Most of the approaches depend on stakeholders’ opinions to analyze the design alternatives and to find conflicts. However, human opinions are prone
to faults; and sometimes, the stakeholders themselves are unaware of the level of support given to the quality attributes and the existence of conflicts.

This chapter explained various architectural analysis methods during architectural design phase and lists out the comparison between them in a tabular format. Except CBAM and WINCBAM, all other methods do not have mechanism to identify the conflicts goals. Again the identification is heuristic based. Except WinCBAM, all the methods evaluate the architecture and make decisions after the architecture is fully developed, but before implementation. There is no systematic way of taking decisions on design alternatives except SSAR whereas others are human based. Once the architecture is developed, there is no mechanism to handle the change in the quality attributes priority dynamically, before the implementation starts. These issues are identified as gaps that lead to the problem focused in this research.