SCENARIO-BASED SOFTWARE ARCHITECTURE EVALUATION METHODS

An overview

Dieter K. Hammer & Mugurel Ionita

Department of Computing Science
Eindhoven University of Technology (EUT)
Eindhoven, The Netherlands

E-Mail: d.k.hammer@tue.nl

LITERATURE

P. Clements, R. Kazman, M. Klein, Evaluating Software Architectures: Methods & Case Studies, Addison-Wesley, 2001

Len Bass, Paul Clements and Rick Kazman, Software Architecture in Practice, Addison-Wesley, 1998

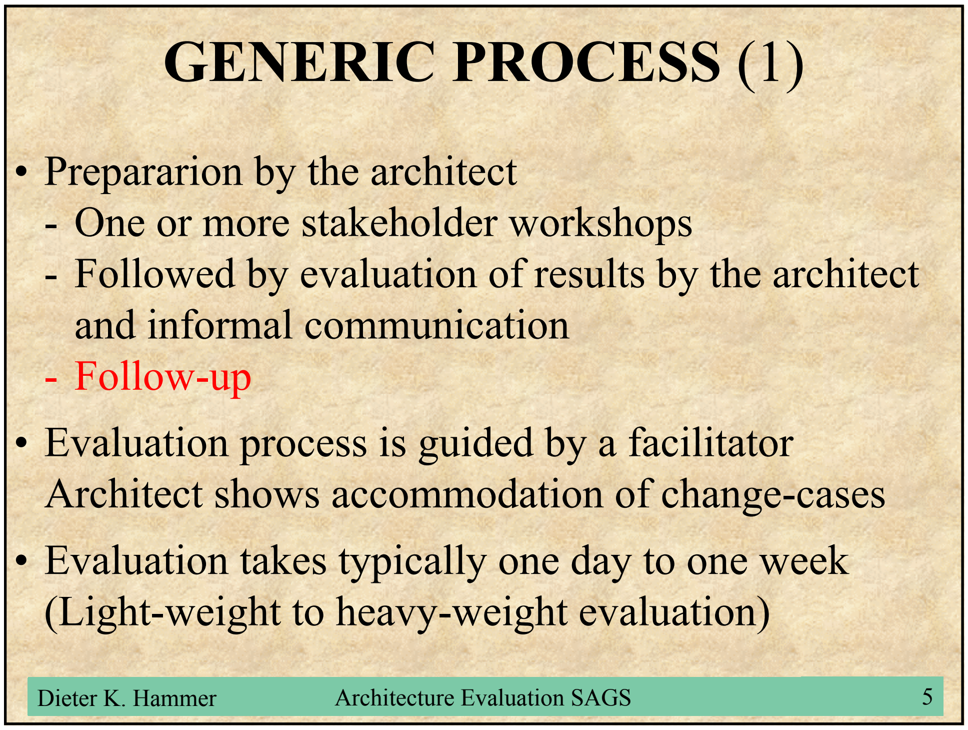
Also see the SEI's web site: http://www.sei.cmu.edu/ata/ata_init.html

USE/CHANGE-CASE BASED SW ARCH. EVALUATION METHODS (1)

- **SAAM**: Software Architecture Analysis Method
  Assesses *modifiability* and areas of potential high complexity (change-case interaction); suited for comparison of architectures
  Assesses *complete architecture* in a *qualitative* way
- **ATAM**: Architecture Trade-off Analysis Method
  Assesses *modifiability* and other *qualities*
  Supports the tradeoff between architectural alternatives
  Assesses *individual design decisions quantitatively*
- **CBAM**: Cost Benefit Analysis Method
  Evaluates the *costs, benefits and schedule implications* of architectural decisions and the level of uncertainty associated with these judgments

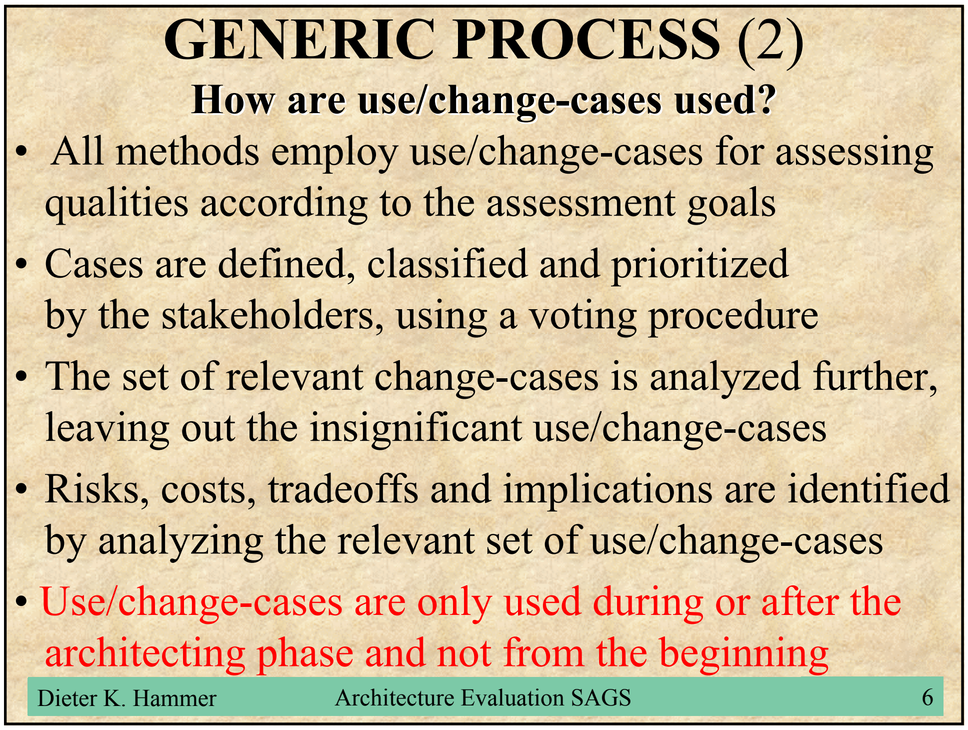
USE/CHANGE-CASE BASED SW ARCH. EVALUATION METHODS (2)

- **ALMA**: Architecture Level Analysis Method
  Assesses software architecture *modifiability*, including *maintenance costs prediction and risk assessment* [Lassing et al. 2001].
- **FAAM**: Family-Architecture Analysis Method
  Assesses the systems familiy *interoperability and extensibility* [Dolan et al. 2001].



GENERIC PROCESS (1)

- Prepararion by the architect
  - One or more stakeholder workshops
  - Followed by evaluation of results by the architect and informal communication
  - Follow-up
- Evaluation process is guided by a facilitator
  Architect shows accommodation of change-cases
- Evaluation takes typically one day to one week
  (Light-weight to heavy-weight evaluation)

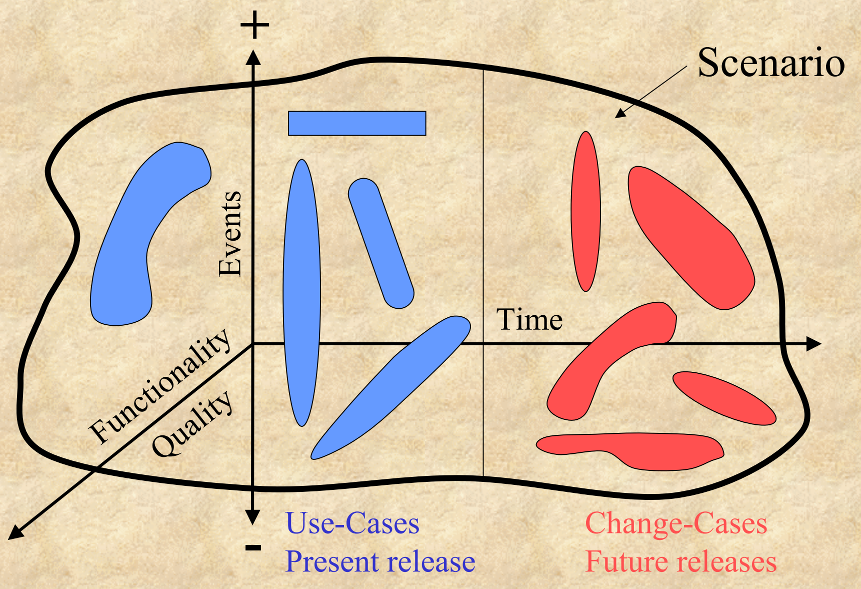


GENERIC PROCESS (2)

How are use/change-cases used?

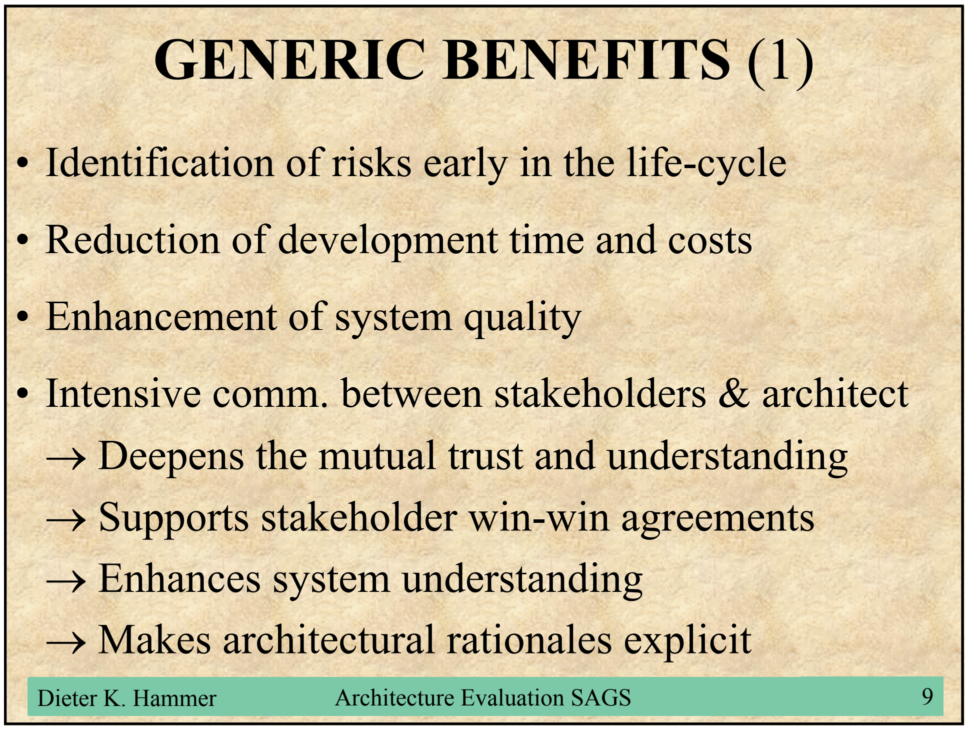
- All methods employ use/change-cases for assessing qualities according to the assessment goals
- Cases are defined, classified and prioritized
  by the stakeholders, using a voting procedure
- The set of relevant change-cases is analyzed further, leaving out the insignificant use/change-cases
- Risks, costs, tradeoffs and implications are identified by analyzing the relevant set of use/change-cases
- Use/change-cases are only used during or after the architecting phase and not from the beginning

SCENARIOS vs USE/CHANGE-CASES



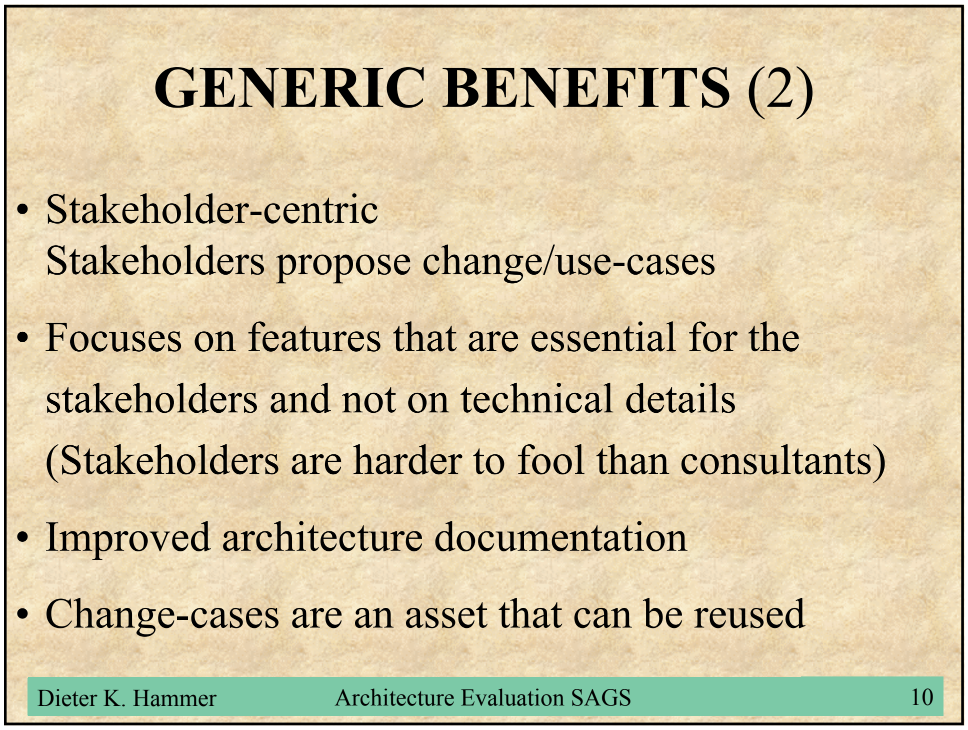
DEFINITIONS

- **Vision**
  Basic idea about a plausible future
- **Scenario**
  (Multimedia) story about a plausible future
  There are different types of scenarios like strategic-, quality-, learning-, failure scenarios and roadmaps
- **Use-case**
  Detailed description of system behaviour in response of a request from one ore more stakeholders (actors)
- **Change-case**
  Use-case about a plausible future situation



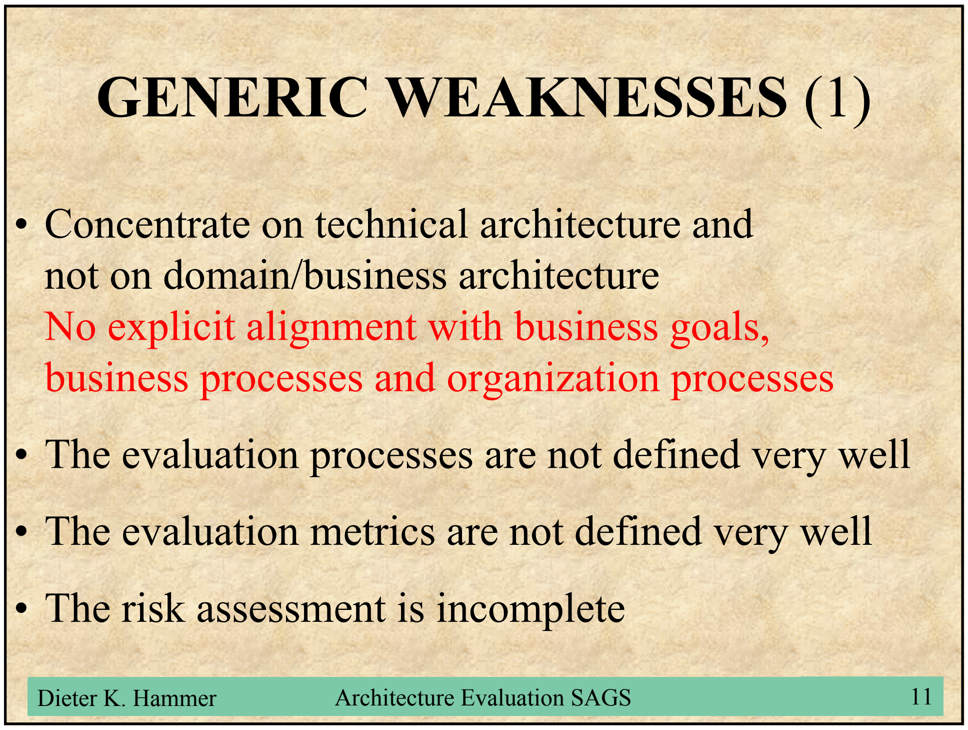
GENERIC BENEFITS (1)

- Identification of risks early in the life-cycle
- Reduction of development time and costs
- Enhancement of system quality
- Intensive comm. between stakeholders & architect
  → Deepens the mutual trust and understanding
  → Supports stakeholder win-win agreements
  → Enhances system understanding
  → Makes architectural rationales explicit



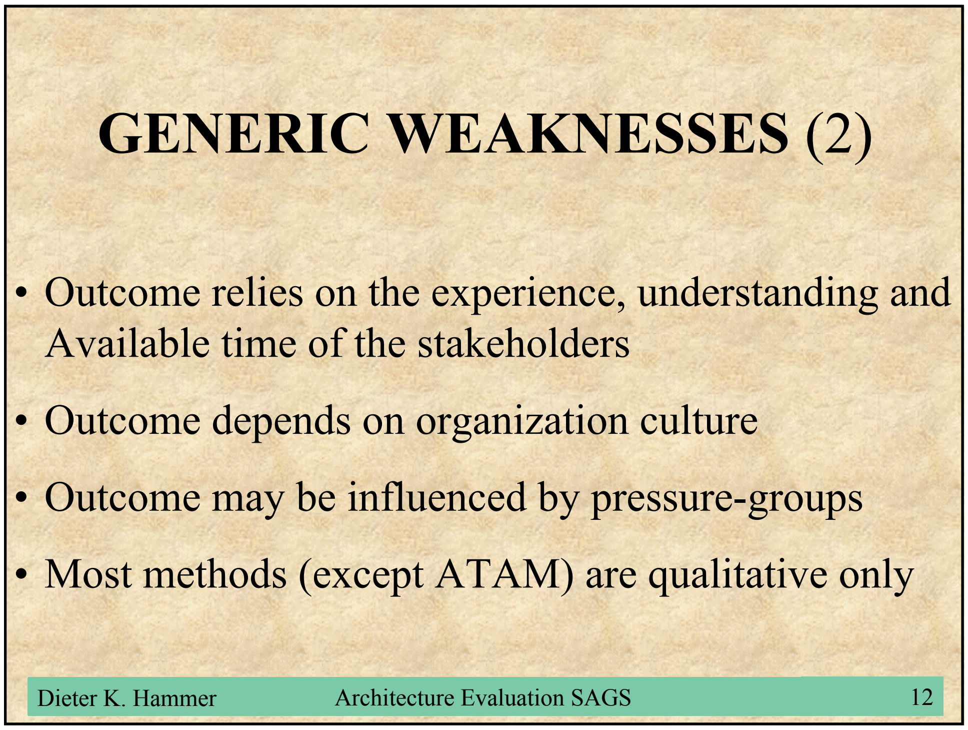
GENERIC BENEFITS (2)

- Stakeholder-centric
  Stakeholders propose change/use-cases
- Focuses on features that are essential for the stakeholders and not on technical details (Stakeholders are harder to fool than consultants)
- Improved architecture documentation
- Change-cases are an asset that can be reused



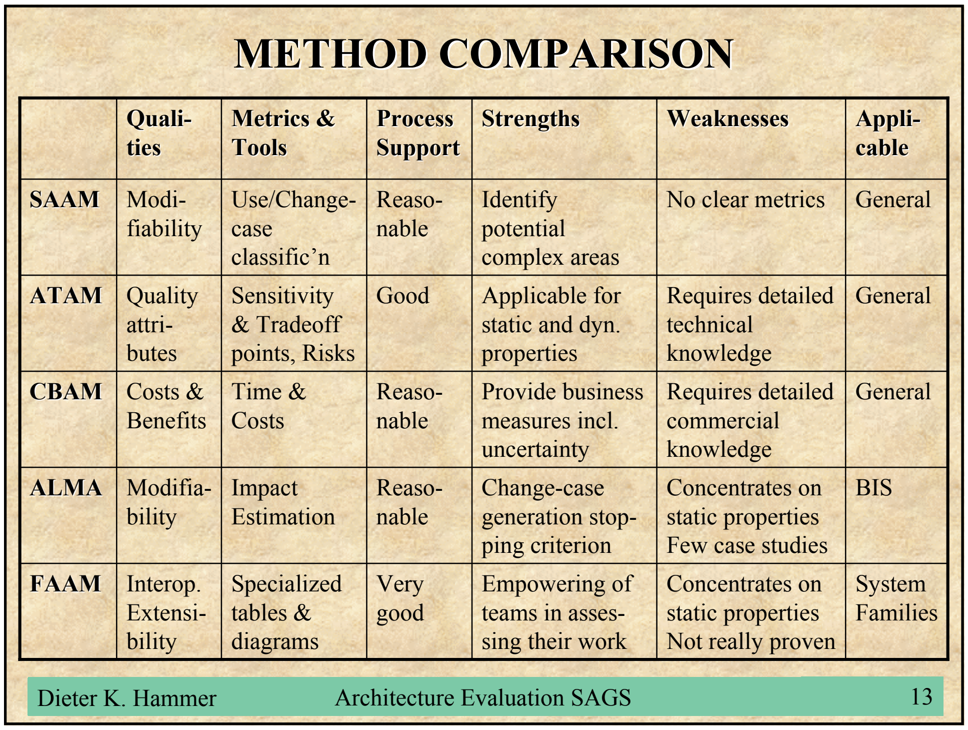
GENERIC WEAKNESSES (1)

- Concentrate on technical architecture and not on domain/business architecture
  No explicit alignment with business goals, business processes and organization processes
- The evaluation processes are not defined very well
- The evaluation metrics are not defined very well
- The risk assessment is incomplete



GENERIC WEAKNESSES (2)

- Outcome relies on the experience, understanding and Available time of the stakeholders
- Outcome depends on organization culture
- Outcome may be influenced by pressure-groups
- Most methods (except ATAM) are qualitative only



METHOD COMPARISON

	Quali-ties	Metrics & Tools	Process Support	Strengths	Weaknesses	Appli-cable
SAAM	Modi-fiability	Use/Change-case classific'n	Reaso-nable	Identify potential complex areas	No clear metrics	General
ATAM	Quality attri-butes	Sensitivity & Tradeoff points, Risks	Good	Applicable for static and dyn. properties	Requires detailed technical knowledge	General
CBAM	Costs & Benefits	Time & Costs	Reaso-nable	Provide business measures incl. uncertainty	Requires detailed commercial knowledge	General
ALMA	Modifia-bility	Impact Estimation	Reaso-nable	Change-case generation stop-ping criterion	Concentrates on static properties Few case studies	BIS
FAAM	Interop. Extensi-bility	Specialized tables & diagrams	Very good	Empowering of teams in asses-sing their work	Concentrates on static properties Not really proven	System Families

Scenario-Based Software Architecture Evaluation Methods: An Overview

Mugurel T. Ionita[1], Dieter K. Hammer[1], Henk Obbink[2]

[2]Department Software Architectures, Philips Research,
Prof. Holstlaan 4, 5656 AA Eindhoven, The Netherlands,
henk.obbink@philips.com

[1]Department of Mathematics and Computing Science, Technical University Eindhoven,
P.O. Box 513, Den Dolech 2, 5600 MB, Eindhoven, The Netherlands,
m.t.ionita@tue.nl, d.k.hammer@tue.nl

Abstract

Software analysis and evaluation becomes a well-established practice inside the architecting community of the software systems. The development effort, the time and costs of complex systems are considerably high. In order to assess system's quality against the requirements of its customers, the architects and the developers need methods and tools to support them during the evaluation process. Different research groups have taken such initiatives and are proposing various methods for software architecture quality evaluation.

1. INTRODUCTION

Recently, a number of new scenario-based software architecture evaluation methods have been developed by different academic groups and published in form of books or doctoral dissertation theses. Many of these methods are refinements of SAAM or ATAM, an initiative of Carnegie Mellon Institute. They usually restrict themselves to a particular class of systems and to a limited set of "ilities". For example, the ALMA method described in this paper focuses on the modifiability of Business Information Systems. Another newly developed approach, the FAAM, assesses the interoperability and extensibility of information-system families.

Even though the latest methods were always using the previous defined ones, there was not been yet much effort done towards an evaluation of their relative merits. The international working group on Software Architecture Review and Assessment (SARA) has taken the initiative of publishing a review with all existing evaluation methods.

This paper is intended as a contribution to this review. The analysis is performed in accordance with the requirements specified in the SARA report [8].

Mugurel T. Ionita – Ph.D. Student of Technical University Eindhoven

Below there is given the set of methods currently available and supporting the analysis of software architecture quality attributes:

1. SAAM, Software Architecture Analysis Method, [1],[3]
2. ATAM, Architecture Trade-off Analysis Method, [1],[2]
3. CBAM, Cost Benefit Analysis Method, [1],[4]
4. ALMA, Architecture Level Modifiability Analysis [5], [6]
5. FAAM, Family – Architecture Analysis Method [7]

The content of the analysis is organized in the following manner: firstly, the description of each method is given; secondly, an overview including all different methods together with a comparison between them is given.

2. OVERVIEW

2.1. Software Architecture Analysis Method (SAAM)

2.1.1. SAAM Context

SAAM is the first widely promulgated scenario-based software architecture analysis method. It was created [3] to assess the architectures' *modifiability* in its various names.

2.1.2. SAAM Purpose

SAAM creators looked for a method able to express the different quality claims of software architectures (such as modifiability, flexibility, maintainability, etc.) by means of scenarios and to evaluate them against the actuals. In practice SAAM has proven useful for quickly assessing many quality attributes such as *modifiability, portability, extensibility, integrability*, as well as *functional coverage.*
The method can also be used to assess quality aspects of software architectures such as performance or reliability. However, ATAM is treating these aspects in more detail (see page 3), being an improved version of SAAM.
If a single architecture is analyzed, SAAM indicates the weak or strong points, together with the points of where the architecture fails to meet its modifiability requirements.

If two or more different candidate architectures, providing the same functionality, are compared with respect to their modifiability SAAM can produce a relative ranking between them.

2.1.3. Key Factors in SAAM Development

The development of SAAM was motivated by a variety of opinions about software architectures and the lack of methods and common basis to address them. The designers and the architects of software systems were not able to reason about the quality of their developed software, or still under development. Consequently, common quality attributes like modifiability, flexibility or maintainability, were not associated with direct software artifacts that can be analyzed and measured.

2.1.4. Prerequisites and Inputs of SAAM

System quality attributes that are going to be evaluated in a SAAM session must be addressed in a certain context. This imposed the adoption of scenarios as the descriptive means in specifying and evaluating qualities. Besides scenarios, there must be available for all participants the system architecture description, the reference artifact, which the quality scenarios are mapped onto.
A number of scenarios, describing the interaction of a user with the system, are the primary inputs to a SAAM evaluation session.

2.1.5. Steps in a SAAM Evaluation Session

The method consists of six main steps, which typically are preceded by a short overview of the general business context and required functionality of the system.

SAAM Step 1– Develop Scenarios
The first step in a SAAM session is a brainstorm exercise with the scope of identifying the type of activities that the system must support. These activities together with possible modifications that the stakeholders can anticipate are grouped in so called *system scenarios*. In developing scenarios the challenge is to capture all major uses and users of the system, all quality attributes and their associated level that the system must reach and most important all foreseeable future changes to the system.
This exercise is usually performed two times. The more iterations and architectural information is shared the more scenarios are surfaced by the participants. Thus the architecture description and scenario development influence each other. The recommendation is to perform in parallel these activities.

SAAM Step 2 – Describe Architecture(s)
In the second step of the SAAM session are presented the candidate architecture(s). The architectural notations used should be well understood by the participants and must indicate the static representation of the system (components, their interconnections and the relation with the environment) as well as the dynamic behavior of the system. This can take the form of a natural-language specification of the overall behavior or some other more formal specification.

SAAM Step 3 – Classify and Prioritize Scenarios
At this point of the analysis the scenarios are classified in *direct* scenarios and *indirect* scenarios (their equivalents in UML notation are use-cases, respectively change-cases). A direct scenario is supported by the candidate architecture because it is based on requirements, which the system has been evolved from. The direct scenarios are perfectly candidates as a metric for the architecture's performance or reliability. An indirect scenario is that sequence of events for which realization or accomplishment the architecture must suffer minor or major changes. The prioritization of the scenarios is based on a voting procedure. Since SAAM is addressing the system's modifiability assessment, the voting results will be a set of indirect scenarios that are considered most likely to occur.

SAAM Step 4 – Individually Evaluate Indirect Scenarios
In case of a direct scenario the architect demonstrates how the scenario would be executed by the architecture. In case of an indirect scenario the architect describes how the architecture would need to be changed to accommodate the scenario. For each indirect scenario there must be identified the architectural modifications needed to facilitate that scenario together with the impacted and/or new system's components and the estimated cost and effort to implement the modification.

SAAM Step 5 – Assess Scenario Interaction
When two or more scenarios are requesting changes over the same component(s) of the architecture, they are said to interact. In this case, the affected components need to be modified or divided into sub-components in order to avoid of the interaction of the different scenarios.

SAAM Step 6 – Create an Overall Evaluation
Finally a weight is assigned to each scenario in terms of its relative importance to the success of the system. The weighting ties back to the business goals supported by a scenario or other criteria like costs, risks, time to market, and so on. Based on this scenario weighting can be proposed an overall ranking if multiple architecture are compared. Alternatives for the most suitable architecture can be proposed, covering the direct scenarios and requiring least changes in supporting the indirect scenarios.

2.1.6. SAAM Roles

There can be identified three classes of roles
a. External stakeholders are having no direct involvement in the software architecture development process. They are the system's stakeholders and their role is to present the project business goals, provide the system quality attributes and their expected level of achievement in a measurable way, and provide the direct and indirect scenarios together with their prioritization and classification. Examples of external stakeholders are customers, end users, marketing specialists, system administrators, maintainers, etc.
b. Internal stakeholders are having a direct involvement in proposing software architectural strategies that can meet the quality requirements. They have the role of analyzing, defining and presenting the architectural concepts estimating the costs and schedule associated with these strategies.

Examples of internal stakeholders are the software architects, system analysts or the architecture team.

c. The SAAM team has no direct stake in the system's software architecture but conducts the SAAM evaluation session. They have the role of supporting the system's stakeholders presenting the business goals as such as after the presentation the system's significant quality attributes and their associated scenarios can be easily elicited and formulated. SAAM evaluation team consists of an evaluator (team leader or spokesperson), application domain experts, external architecture experts (optional, for a more formal evaluation), and a secretary.

2.1.7. Effort Estimate in Applying SAAM

SAAM evaluation team in accordance with the project scale and goals can appreciate the effort in applying the method. A SAAM evaluation session's agenda is scheduling all the six steps to be performed in one day. This excludes the preparation time and effort of the architect invested in architecture description preparation and the scenarios' evaluation. Depending on the size of the project and the number of stakeholders involved, the duration of the session varies as well. A SAAM study report shows that in 10 evaluations performed for projects ranging in size from 5-100 KLOC the effort is estimated at 14 days [9]. Most participants also noted that there are increased start-up costs for an organization beginning an architecture review practice due to a lack of architectural maturity in the company. Rational Software Corporation has performed around 30 evaluations and charges an average cost of $50K for projects of at least 500 KLOC in size [9].

2.1.8. SAAM Tool Support

Up to now no tools support SAAM evaluation sessions. The voting procedure invoked for the scenario prioritization and the modifiability estimates with respect to costs and effort to adapt the architecture are the only techniques used.

2.1.9. Alternatives for SAAM

Starting form SAAM and building on the competence and opportunity created in the area, there have been developed several methods able to assess the architectures' modifiability. One of these methods is ATAM [2], developed by the same group, which initiated SAAM. Another method is ALMA [5], [6], which is also a scenario-based analysis method suitable for software architecture modifiability assessment.

2.1.10. SAAM Outcomes and Strengths

The strengths of the SAAM method are

- Stakeholders' in-depth understanding about the architecture being analyzed.
- In some cases, after a SAAM evaluation session the software architecture documentation is improved.
- Enhanced communication among the stakeholders.

With respect to modifiability the SAAM the strengths and outcomes are:

– A mapping of the brainstormed scenarios onto the architecture regarding future changes of the system. As a result, the areas of high potential complexity are identified. Also costs and effort for performing the necessary changes are estimated. Based on ranking of the future scenarios a side effect is the opportunity of creating roadmaps for future development.

2.1.11. Remarks about SAAM

There have been identified a set of open questions in applying SAAM and thus architecture's evaluation success. These may address SAAM future improvements.

– The scenario generation process is based on stakeholders' vision. It takes a very little effort for a stakeholder to imagine any of the "indirect scenarios".

– SAAM does not provide a clear quality metric for the architectural attributes being analyzed.

– The architecture description is a fuzzy notion being adopted no standardized notation or architectures' description methods. All SAAM is specifying is that must be a candidate architecture(s), which "should be described in an architectural notation that is well understood by the parties".

– The evaluation team is relying only on the architects experience in proposing different architectures (if any). Else there are no abilities to reason about possible architectural options since the evaluation team is not familiar with the complete set of requirements and the technical background in the business area.

– SAAM is a stepwise method for performing the software architecture analysis. However, it provides few techniques for performing the different steps, mainly relying on the analyst/evaluator experience.

– SAAM does not empower the architecting team for a pre-preparation in order to facilitate a possible SAAM session, thus will take a lot of effort for the evaluation team to be accepted by the system architects or designers.

2.2. Architecture Trade-Off Analysis Method (ATAM)

2.2.1. ATAM Context

ATAM is a scenario-based architecture method for assessing quality attributes such as: *modifiability, portability, extensibility, and integrability*.

Besides the assessment of quality attributes, ATAM explores the quality attributes interaction and their interdependencies highlighting trade-off mechanisms and opportunities between different qualities.

2.2.2. ATAM Purpose

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).

2.2.3. Key Factors in ATAM Development

ATAM development has been motivated and influenced by:

– 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.

2.2.4. 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", "trade-off 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 (1) the initial requirements of the system and (2) 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.

2.2.5. 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. .

2.2.6. 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.

2.2.7. Effort Estimate in Applying ATAM

Since the method is very project dependent, there are no actual effort figures presented in ATAM. The only reference is given by the method phases, which are spread over three different days. After each phase the ATAM evaluation team needs a few days for structuring and organizing the information and preparing the next ATAM phase. The evaluation team based on the number of participants at the session, the number of the quality attributes to be assessed as well as the size of the project and the architecture complexity can give effort estimation figures.

2.2.8. ATAM Tool Support

Newly, there has been developed a tool, which integrally supports an ATAM evaluation session. The tool has been developed in by Stephan Kurpjuwiet, a Ph.D. Student of Fraunhofer Institute in collaboration with the Software Engineering Institute (SEI) [10]. The tool provide a set of facilities: evaluation documents management and version control; process guidance; data model constraints checking; architectural transformations support; model data items relation maintenance; user defined constraints, templates support for quickly capture relevant pieces of information; collaborative data model maintenance; log an report features for model inconsistencies; collaborative editing (for evaluators and scribes). A close look to all these features reveals that they are very close to what Rational tools are providing. For example for capturing and communicating the architecture design description there is existing Rational Rose which provides the designers with an integrated framework for addressing use cases design and implementation. For documentation management and version control Rational Clear Case can be used. None of these tools are mentioned in ATAM description as such. There are suggested as possible helpful tools for improving the communication and the management of the ATAM artifacts during evaluation. Since we have no experience with the ATA Tool recently developed [10], we can't reason about the real benefits of the tool, thus it is only briefly introduced.

2.2.9. Alternatives for ATAM

Architecture Level Assessment Method (ALMA) can do the risk assessment possible to be performed by ATAM. Nevertheless, the tradeoff analysis of different quality attributes is best described in ATAM.

2.2.10. ATAM Outcomes and Strengths

According to Kazman et al. 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.

2.3. Cost-Benefit Analysis Method (CBAM)

2.3.1. CBAM Context

CBAM begins where ATAM leaves off; being an architecture-centric method for analyzing the *costs*, *benefits* and *schedule implications* of architectural decisions [4]. 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.

2.3.2. CBAM Purpose

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.

2.3.3. Key Factors in CBAM Development

The impulse of the CBAM development came form a set of questions, each of which contributed in shaping the method. These questions were addressed as

- How can the architectural decisions be measured and compared in terms of their different implications, costs and benefits?
-
- How can quality attributes be analyzed and trade-off with respects to their costs and benefits involved.
- How can be characterized the uncertainty level associated with these cost and benefits estimates.

2.3.4. Prerequisites and Inputs for CBAM

Since CBAM is building on the ATAM this implies that there will be necessary some prerequisites like:

– Architecture accommodation and presentation necessary for all participants,
– Familiarity with concepts like sensitivity points, trade-off points, descriptive scenarios and requirements elicitation where necessary.

Inputs in a CBAM evaluation session are:

– The business goals presentation.
– The architectural decisions and possible tradeoffs resulted in a former ATAM session.
– The quality attributes expectation level and economical constraints.

Templates and guidelines for supporting the descriptive scenarios' generation process can be provided. The architecture evaluation pre-session (ATAM) is also considered input for CBAM.

2.3.5. Steps in a CBAM Evaluation Session

CBAM consists of two phases. First phase is called *triage* followed by a second phase called *detailed examination*. The first phase is sometimes necessary in case there are many architectural strategies to be discussed and just a few must be chosen for further detailed examination. Else the evaluation process starts right form the second phase. For both phases in CBAM are prescribed six main steps:

CBAM Step 1– Choose Scenarios of Concern and their associated Architectural Strategies
In the first step are chosen the scenarios that concern most the system's stakeholders. For each of these scenarios there are proposed different architectural strategies that address the specific scenarios.

CBAM Step 2 – Assess Quality-Attribute Benefits
In the second step are elicited the quality-attributes benefits form participating managers who, presumably, best understand the business implications of how the system operates and performs.

CBAM Step 3 – Quantify the Benefits of the different Architectural Strategies
In the third step are elicited the architectural strategies from the participating architects who, presumably, understand how a certain architectural strategy can achieve the desired level of quality.

CBAM Step 4 – Quantify the Architectural Strategies' Costs and Schedule Implications
In the fourth step are elicited the cost and schedule information form the stakeholders (both business managers and architects). The evaluation team assumes that within the organization already exists enough experience in estimating time schedules and associated costs.
CBAM Step 5 – Calculate Desirability
Based on the elicited values resulted in the previous step, the evaluation team the desirability level for each architectural strategy based on the ratio "benefit divided by cost". Further more there is calculated the uncertainty associated with these values, which helps in the final step of making decisions.

CBAM Step 6 – Make Decisions
Based on the values resulted in step five and the degree of realism of these values there are chosen the best cost-benefit effective architectural strategies which can fulfil best the elicited descriptive scenarios.

2.3.6. CABM Roles

There can be identified three classes of roles
a. External stakeholders are having no direct involvement in the software architecture development process. They are the system's stakeholders and their role is to present the project business goals, provide the system quality attributes and their expected level of achievement in a measurable way, and assess the CBAM evaluation results. Examples of external stakeholders are business management team, project management, etc.
b. Internal stakeholders are having a direct involvement in proposing software architectural strategies that can meet the quality requirements. They have the role of analyzing, defining and presenting the architectural concepts estimating the costs and schedule and uncertainty associated with these strategies. Examples of internal stakeholders are the software architects, system analysts or the architecture team.
c. The CBAM team has no direct stake in the system's software architectural strategies but conducts the CBAM session. They the role of supporting the system's stakeholders presenting the business goals as such as after the presentation the system's significant quality attributes and their associated scenarios can be easily elicited and formulated. CBAM team also supports the architecting team in addressing the architectural strategies able to satisfy the quality scenarios and estimate the costs, benefits and time scheduling associated with these strategies. CBAM evaluation team consists of an evaluator (team leader or spokesperson), application domain experts, external architecture experts (optional, for a more formal evaluation), and a secretary if necessary.

2.3.7. Effort Estimate in Applying CBAM

The evaluation team in accordance with the project scale and goals must appreciate the effort. Looking at the organizational aspects and CBAM steps one can say that most of the effort is concentrated in architectural strategies elicitation and cost-benefit-schedule prediction part. A CBAM session takes one or two days. In addition an ATAM

session can be performed as well, thus the total time allocated is increasing to at least four working days.
In terms of man-hours estimation and procedural costs, the CBAM team can provide certain effort values.

2.3.8. CBAM Tool Support

No software applications are specified in the method description that would directly support the assessment session. The only indicator is given in the step four of the CBAM where the architectural strategies' costs and benefits are quantified using a desirability metric formulas and uncertainty expression estimation.

2.3.9. Alternatives for CBAM

So far, there is no method that incorporates the economical perspective in the software quality attributes evaluation and tradeoff analysis.

2.3.10. CBAM Outcomes and Strengths

CBAM general strengths and outputs are:

– The method provides *values* as a basis for a rational decision making process in applying certain architectural strategies
– The method provides a business *measure* that can determine the level of return on investment of a particular change to the system.
– The method will help organizations in analyzing and pre-evaluating the resource investment in different directions by adopting those architectural strategies that are maximizing the gains and minimize the risks.

Since CBAM is built on the general architecture assessment methods like SAAM and ATAM, the method is inheriting their benefits with respect to efficiency

2.4. Architecture-Level Modifiability Analysis (ALMA)

2.4.1. ALMA Context

Initially ALMA has been developed and tested for Business Information Systems (BIS) only, as a scenario-based evaluation method for software architecture quality attributes, focusing on *modifiability*. ALMA should also be applicable for Embedded Systems (ES), but this assumption has not been proven yet.

Modifiability analysis usually has one of three goals:

– Prediction of future modification costs
– Identification of system inflexibility
– Comparison of two or more alternative architectures

2.4.2. ALMA Purpose

ALMA is a scenario-based analysis method suitable for software architecture modifiability assessment by employing a set of indicators: maintenance cost prediction, risk assessment. In case of assessing and comparing different system, the modifiability analysis performed with ALMA supports software architecture selection as well. For this purpose ALMA uses change-scenarios, provided by the system stakeholders.
The modifiability analysis starts with defining a set of scenarios that might occur during the evolution of the system. Scenarios are used to verify how well the current architecture supports or may accommodate future changes.

2.4.3. Key Factors in ALMA Development

Major problems with the existing detailed methods for assessment and evaluation include the following issues:

– They focus on a single quality attribute and ignore the others, equally important attributes.
– They tend to be very detailed and elaborated in the analysis, requiring, sometimes, excessive amounts of time to perform a complete analysis.
– The techniques are often intended for the later design phases and often require detailed information not yet available during architecture design, e.g. Source metrics like average depth of inheritance trees.

Other factors that contributed to ALMA are:

– Actuals, which were showing that 50% to 70% of the total lifecycle cost for a software system is spent in evolving the system. Thus, systems' modifiability to (un)expected changes must be considered.
– Increasing pressure to improve quality, minimize costs and lead-time in order to stay in business is strongly influenced by the systems' modifiability.
– Lack of architecture assessment techniques that quantify quality attributes of the software architecture.
– Lack of methods that focus on modifiability, with possibility of addressing multiple analysis goals, making the assumption explicit and providing well-documented techniques for performing its steps.

2.4.4. Prerequisites and Inputs for ALMA

ALMA builds on top of SAAM. Thus the method partly inherits the same kind of prerequisites:

– Stakeholders are asked to provide possible change scenarios that may occur in the future system's lifecycle. ALMA uses these *change scenarios* to analyze the modifiability of the architectures.
– The evaluator/analyst(s) must be able to assess the impact and costs of these change scenarios.

The inputs for an ALMA evaluation session are:

– The "4+1model" proposed by Kruchten for specifying the architecture.
– The UML notation for software architecture description.

Operational wise, ALMA team may use templates, written rules and others supporting materials for scenario generation process or for the architecture description.

2.4.5. Steps in an ALMA Evaluation Session

ALMA method consists of five steps. The steps are not always performed sequentially. Re-iterations over the various steps are also possible.

ALMA Step 1 – Set the Analysis Goal
First activity is concerned with defining the analysis goal. ALMA can pursue different goals: risk assessment,

maintenance and costs prediction, or software architecture selection. Maintenance and cost prediction estimate the effort that is required to modify the system architecture for accommodating future changes. Assessing the risk gives an overview about the changes that are difficult to accommodate using a certain architecture model. Comparing proposed models and selecting the best one does the selections of software architectures.

ALMA Step 2 – Describe the software architecture(s)
The architecture description uses a number of architectural views, which describe the decomposition of the system into components, the relationship between components, and the relationships between the system and its environment.

ALMA Step 3 – Elicit change-scenarios
Change-scenario elicitation is the process of finding and selecting the scenarios that may play a role in architecture modifiability. These types of scenarios are used during the ALMA evaluation session. Eliciting change scenarios involves such activities as identifying the relevant stakeholders, interviewing them, properly documenting the resulted change-scenarios, and assessing the feasibility of the results together with the stakeholders.

ALMA Step 4 – Evaluate the change-scenarios
During this task, the ALMA analyst(s) cooperates with the architects and system developers to determine the impact of the change-scenarios and to express the result in a suitable and measurable way for the goal of the analysis.

ALMA Step 5 – Interpret results
After evaluating the change-scenarios, the results are interpreted in accordance with the goals of the analysis and verified against system requirements. The results are used to predict the maintenance effort. The value of the maintenance estimate is limited and not always trustable since no benchmarks or other estimates are available for this attribute.

2.4.6. ALMA Roles

Given the large participation of different stakeholders to an ALMA session the different roles may be grouped in: external stakeholders, internal stakeholders, and ALMA team.

a. External stakeholders have no direct involvement in the software architecture development process. Their role is to present the business context of the project, to provide the change scenarios and the system initial requirements. At the end this group must decide about the continuation of the development based on the evaluation outcome. Examples of external stakeholders are users, customers, maintainers, sponsors, product owner, product managers, etc.

b. Internal stakeholders have a direct involvement in the software architecture development process. They analyze, define and present the different architectural concepts and views. For the ALMA evaluation session they are responsible (with different efforts in participation) for describing and presenting the software architecture of the system. Together with the evaluation team the internal stakeholders estimate the impact of the change scenarios on the architecture with respect to the analysis goals; assess the reliability of the ALMA results in expressing the modifiability effort; re-design (if is applicable) the agreed upon solutions. Examples of internal stakeholders are software architects, analysts, designers, developers, etc.

c. The ALMA team has no direct stake in the system software architecture but it is invited to conduct the evaluation process. The ALMA team has the leading role in presenting and proceeding with the evaluation, recording the intermediate assessment artifacts, assessing and presenting the final results. The ALMA team has also the role of supporting the stakeholders in change-scenario generation and the architects in the software architecture presentation (if necessary). Together with the architects ALMA team has the role of identifying the change scenarios impact on different architectures and predicting the modifiability effort in each case. ALMA evaluation team consists of a team leader or spokesperson, architecture analysts and a secretary.

2.4.7. Effort Estimate in Applying ALMA

By knowing the stakeholders and the architectural candidates, as well as the number of change-scenario and the architectural complexity, an estimate can be given. In the method description there are no further actuals given.

2.4.8. ALMA Tool Support

Tools do not yet support ALMA. The change-scenarios are generated on a person-to-person interview basis, each of which may use templates, rules or guidelines. Handlers like white boards, flip charts, estimation tables, meeting notes or recordings on any media types may be used basic. For capturing and communicating the software architecture description UML diagrams are used.

2.4.9. Alternatives for ALMA

The Architecture Tradeoff Analysis Method (ATAM) is a possible substitute of ALMA with respect to modifiability. Both methods are quite similar, since they use scenarios for assessing the quality attributes and provide estimates with respect to the analysis goals.

2.4.10. ALMA Outcomes and Strengths

The general strengths and outputs of an ALMA session are claimed to be:

a. ALMA focuses on architectural abstractions, which represent the domain functionality and the driving quality attributes.
b. Scenario evaluation is based on impact analysis. This consists of identifying the affected components and determining the effect on those components, together with the ripple effects.
c. Stakeholders have two options in generating the change scenarios. A top-down approach a set of general change scenarios categories where are identified, followed by a refinement in terms of their particular instances. And the bottom-up approach, where the change scenarios are just collected as resulting form the interviews with stakeholders and later categorized in scenario classes.

d. The possibility of assessing modifiability from different perspectives: maintenance and cost prediction, risk assessment, and/or software architecture selection.

e. Make important assumptions explicit.

f. Provide repeatable techniques for performing the steps.

The outcomes of the method can be summarized in:

g. The results of the impact estimates for each scenario are expressed as (1) the size of the modification to existing components, or (2) the estimated size of the components needs to be introduced.

h. A *modifiability prediction model* based on the estimated change volume and productivity ratios. The model assumes that the change volume is the main cost driver, thus gives a productivity figure for the cost of adding new code and modifying old code

i. A *scenario generation stopping criterion* (1) if all categories from the classification scheme have been explicitly considered, or (2) generation of new change scenarios which do not affect the classification structure.

2.4.11. Remarks about ALMA

The method, as a general remark, lacks the means to decide upon the accuracy of the results of the analysis. ALMA cannot reason about the accuracy of the maintenance prediction numbers. Also, one cannot reason about the completeness of the risk assessment.

2.5. Family-Architecture Assessment Method (FAAM)

2.5.1. FAAM Context

FAAM is a method for architecture assessment of information-system families, focusing on two related quality aspects: *interoperability* and *extensibility.*

2.5.2. FAAM Purpose

The purpose of FAAM is to establish a process (supported by guidelines, metrics, recommendations and process) for assessing information-system family architectures. Different from other methods, FAAM contributes in:

– Actively involving the product-family stakeholders in the product creation process,
– Focusing on *interoperability* and *extensibility* quality attributes of the information-system families,
– Emphasizing the practical *know-how* mechanisms and techniques to enable the development teams within the organizations to implement the method.

2.5.3. Key Factors in FAAM Development

This section presents some of the concerns that contributed to FAAM development:

- The need of a method, which helps stakeholders in identifying and expressing future changes-cases of the system.
- The need of techniques which contribute to the stakeholders-architects interaction during the family-architecture development
- The need of techniques, which can make explicit the architecture-rationales for the family-systems stakeholders.
- The need of ability to reason about family-systems interoperability and extensibility in the early phases of architecture definition.

2.5.4. Prerequisites and Inputs for FAAM

FAAM is based on the same general evaluation principles like SAAM and ATAM with special attention to some prerequisites like:

– The invited participants must be made familiar with family-related techniques of establishing initial requirements and goals of the assessment.
– The architecture specification must exist or be prepared before the FAAM session.
– The facilitator (sponsor, architect or stakeholders) must be ready to express what is expected from the assessment.

The inputs for a FAAM evaluation session are:

– Change-case templates for specifying possible changes with respect to system-family interoperability and extensibility.
– Templates or techniques for generation of the family-feature-maps, migration-maps, family-context-diagrams, and criteria for requirements ranking.

There can be used also guidelines or rules for supporting the requirements and change-case generation process. The architecture description is based on "4+1 model" views, but typically, the emphasis is on the logical, process and views.

2.5.5. Steps in a FAAM Evaluation Session

The chronological steps of a FAAM session are described below. Important to notice is that the FAAM steps must be adapted in response to general architecture assessment experience of the organization.

FAAM Step 1 – Define the Assessment Goal
This is a basic exercise intended to determine the goal of the assessment. In order to answer this question, firstly some challenges must be dealt with:

- Establish the scope and content of the system-family with the stakeholders;
- Establish the future plans for the family (interoperability and extensibility changes);
- Provide guidelines to stakeholders to help generate requirements;
- Provide guidelines on prioritizing competing requirements for assessment;

FAAM Step 2 – Prepare System-Quality Requirements
In this step the stakeholders' participation in identifying and prioritizing system-quality requirements is requested. The challenge here is to provide the means for stakeholders to represent the requirements in a structured, assessment-ready manner.

FAAM Step 3 – Prepare Architecture
This step of the method is concerned with getting the architecture representation available for presentation and

assessment against the stakeholders' requirements. The challenge here is to provide guidelines for the architects in representing the architectural views.

FAAM Step 4 – Review / Refine Artifacts
Here the goal is to come to an agreement on the set of requirements and the architectural views that are relevant to be carried through to the subsequent assessment steps. The challenge here is to clarify the business or logical constraints that may influence the assessment continuation.

FAAM Step 5 – Assess Architecture Conformance
The architecture description is verified against the specified requirements with focus on the ability and easiness to integrate or to satisfy the change-cases specified in Step 2.

FAAM Step 6 – Report Results and Proposals
In this phase the assessment results are recorded and communicated back to the stakeholders. Based on the results of Step 5 the facilitator together with the architecting team draws the lessons learned for the assessment exercise.

2.5.6. FAAM Roles

As in the previous methods the same classification, in general groups, can be adopted for FAAM active roles:

a. Family stakeholders (the assessors), have no direct involvement in the software architecture development process. Their role is to present the business context of the project, to provide and rank the system requirements, and decide upon the assessment results. Examples of external stakeholders are business-management, product-management, customer-support management, development-management, etc.

b. Internal stakeholders are directly involved in the software architecting process. They have the role of analyzing, defining and presenting the concepts and the architectural views during the FAAM session. Examples of internal stakeholders are software architect or the architecture team.

c. The FAAM team (facilitators) has no direct stake in the system's software architecture but conducts the evaluation process. The facilitators have the role of supporting the stakeholders in requirements and change-case generation and the architects in the software architecture presentation (if necessary). FAAM evaluation team consists of an assessment facilitator (team leader or spokesperson), application domain experts, external architecture experts (optional, for a more formal evaluation), administrative and logistical support (secretary, also optional).

2.5.7. Effort Estimate in Applying FAAM

Since the effort in applying FAAM is given by the nature of the assessment, the number of participants and the level of experience in dealing with the assessment sessions, the necessary time interval can be estimated at most 3 session days. Form a comparison with other methods and the case studies where FAAM has been applied one can say that the effort is relatively small. The complexity is always given by the family-system under assessment.

2.5.8. FAAM Tool Support

As indicated earlier, FAAM is supported only by family related techniques, guidelines and templates in generating change-case-guidelines and templates, requirements-ranking criteria, family-feature maps, migration-maps, family-context diagrams. There is no automatic tool-support generation specified in FAAM.

2.5.9. Alternatives for FAAM

FAAM builds on the experience of SAAM by adding a family perspective and practical advanced techniques for assessment facilitation. ATAM can be an alternative for FAAM. The difference between FAAM and ATAM is the scope of the evaluation. ATAM hasn't been applied yet for family information-systems but in principle can support it. FAAM has been designed for assessing the interoperability and extensibility of the family systems only. Thus, the method is supported by dedicated techniques, templates and process guidelines accordingly.

2.5.10. FAAM Outcomes and Strengths

FAAM general strengths and outputs are:

- The method provides *how-to* advice to enable development teams to conduct their own self-assessment as a means towards continuous improvement.
- The general assessment process is tailored for the domain of information-systems families.
- FAAM has a well-defined process workbench description. This is useful for supporting the participants with practical techniques in generating the necessary process artifacts for the evaluating interoperability and extensibility attributes
- FAAM is built on the general architecture assessment methods like SAAM and ATAM, thus inheriting their benefits with respect to efficiency.

3. Discussions and Conclusions

In this paper have been presented five existing techniques for assessing 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.

Future work directly related with this analysis regards the integration of the scenario-based techniques requirements discipline.

References

[1] Paul Clements, Rick Kazman and Mark Klein, *Evaluating Software Architectures: Methods and Case Studies*, Addison Wesley, 2002.

[2] *"ATAM: Method for architecture evaluation"*: ATAM - Architecture Trade-off Analysis Method report: http://www.sei.cmu.edu/ata/ata_method.html

[3] Rick Kazman, Len Bass, Gregory Abowd, and Mike Webb, "SAAM: A Method for Analyzing the Properties Software Architectures," Proceedings of the 16th International Conference on Software Engineering, Sorrento, Italy, May 1994, pp. 81-90. http://www.sei.cmu.edu/ata/publications.html#reports

[4] *"CBAM: Cost Benefit Analysis Method* http://www.sei.cmu.edu/ata/products_services/cbam.html

[5] Lassing, Nico, Ph.D. Thesis, "*Architecture-Level Modifiability Analysis"*, Ph.D. thesis, Free University Amsterdam, February 2002.

[6] Bengtsson, PerOlof, Ph.D. Thesis, *"Architecture-Level Modifiability Analysis"*, Department of Software Engineering and Computer Science, Blekinge Institute of Technology, Sweden 2002.

[7] Thomas J. Dolan, Ph.D. Thesis, *"Architecture Assessment of Information-System Families"*, Department of Technology Management, Eindhoven University of Technology, February 2002.

[8] Software Architecture Review and Assessment (SARA) Report Version 1.0 available in electronic form at: http://www.rational.com/media/products/rup/sara_report.pdf

[9] Abowd, G., Georgia Institute of Technology, Bass, Clements, Kazman, Northrop and Zaremski, SEI, *"Recommended Best Industrial Practice for Software Architecture Evaluation" 13Jan, 1997.* www.sei.cmu.edu/pub/documents/96.reports/pdf/tr025.96.pdf

[10] Stephan Kurpjuweit, Ph.D. Thesis, *"A Family of Tools to Integrate Software Architecture Analysis and Design"*, Final Draft Version, to be published 2002.

Appendix

Method	Assessed Quality	Metrics and Tools Support	Process Description	Strengths	Weaknesses	Systems Type Applicable for
SAAM	Modifiability	Scenario classification (direct vs. indirect ones)	Reasonable	Identifying the areas of high potential complexity Open for any architectural description	Not a clear quality metric Not supported by techniques for performing the steps	All
ATAM	Modifiability	Sensitivity Points, Tradeoff Points Supported by ATA Tool [10]	Good	Scenarios generation based on Requirements Applicable for static and dynamic properties Quality utility tree	Requires detailed technical knowledge	All
CBAM	Costs, Benefits, and Schedule Implications	Time and Costs	Reasonable	Provide business measures for particular system changes Make explicit the uncertainty associated with the estimates	Identifying and trading costs and benefits can be done by the participants in an open manner	All
ALMA	Modifiability	Impact estimation, Modifiability prediction Model,	Reasonable	Scenario generation stopping criterion	Restricted set of case studies Concentrates on static properties	Business Information Systems
FAAM	Interoperability and Extensibility	Various specialized tables and Diagrams	Very good Detailed process flow	Emphasis on empowering the teams in applying the FAAM session	Only partially proven in one particular environment Concentrates o static properties	System Families

Table A**. - The relevant aspect of all different software architecture evaluation methods.**