NEXOF-RA

NESSI Open Framework – Reference Architecture

IST- FP7-216446

Deliverable D10.2c
Assessment Criteria

Deliverable D10.2 defines the assessment criteria for the NEXOF-RA identifying the key aspects that require validation. The results of the assessment will be reported in D10.3.

CINI (leader)
BT, Logica, Moma, Siemens, Thales, TIE

Due date of deliverable: 16/07/2009
Actual submission date: 01/09/2009

Document Information

IST Project Number	FP7 – 216446	**Acronym**	NEXOF-RA
Full title	NESSI Open Framework – Reference Architecture		
Project URL	http://www.NEXOF-ra.eu		
EU Project officer	Arian Zwegers		

Deliverable	**Number**	D10.2c	**Title**	Assessment Criteria
Work package	**Number**	10	**Title**	Requirements and Assessment Criteria

Date of delivery	**Contractual**	01/09/2009	**Actual**	01/09/2009
Status	Version 1, dated 28/02/2009		final ☑	
Nature	Report ☑ Demonstrator ☐ Other ☐			
Abstract (for dissemination)	The document describes assessment criteria for the scenarios used to define the Reference Architecture.			
Keywords	Assessment criteria, scenarios, requirements			

Internal reviewers	ATO			
	ENG			
Authors (Partner)	CINI (leader), BT, Logica, Moma, Siemens, Thales, TIE			
Responsible Author	Alberto Sillitti		**Email**	asillitti@unibz.it
	Partner	CINI	**Phone**	

1 Executive summary

The goal of this document is to define the assessment criteria that will be used to validate the outcome of NEXOF-RA, the NEXOF reference architecture, against the requirements specified within D10.1 “Requirements Report”.

This document defines the definition of assessment criteria using the ATAM architecture assessment method [12], of which the first five steps are presented in this document, the remaining steps and the architectural evaluation are presented in D10.2 “Assessment results”.

2 TABLE OF CONTENTS

3 INTRODUCTION

Within the deliverable D10.1, the requirements report, the project partners collected possible scenarios of the use of software systems in which services play a central role to derive requirements for service oriented architectures.

These requirements are used by other work packages as a basis for their work to ensure that the results are aligned to what is important for NEXOF-RA stakeholders.

The goal of this document is to define the assessment criteria that will be used to validate the outcome of NEXOF-RA, the NEXOF reference architecture, against the requirements specified within D10.1 "Requirements Report".

Generally speaking, to assess means to make a judgment. To judge objectively, assessment means also to define criteria that are easier to interpret than the assessment itself.

For example, to judge whether to go hiking on a particular mountain next Sunday, the assessment criteria could be that the weather forecast should be for no rain and that all participants have the necessary equipment.

This example shows that assessment criteria have a twofold role: they **identify the specific characteristics by which the outcome is judged**, and at the same time they **indicate those characteristics that are not relevant** (by omission). In the previous example we see that the physical condition of the participants was not considered as a criterion.

This is the challenge in identifying assessment criteria: to pick aspects that are simple enough (if they are easy to understand and to verify it is more likely that their evaluation is objective[1]) and that at the same time they suffice to guarantee a correct judgment.

In our hiking example we forgot to consider the physical fitness of all participants, our assessment could lead to a wrong judgment, i.e., to go to hike and not reach the summit because the group is not fit enough.

In the case of reference architectures, it is necessary to evaluate if the reference architecture supports or obstructs the creation of software architectures that support and not obstruct the implementation of the stated requirements.

The assessment of software architectures involves "thought experiments", modeling and walking-through scenarios that embody the requirements, the development of prototypes or proofs-of-concept, as well as assessment by

[1] "Objective" means in this context that the outcome does not depend on *who* is judging.

experts who look for gaps and weaknesses in the architecture based on their experience [2].

The assessment of an architecture differs from the assessment of an implementation; an architecture represents just “the fundamental **organization** of a system embodied in its components, their relationships to each other and to the environment and the principles guiding its design and evolution [6]” – and not its implementation.

To assess the “organization of a system” with the goal to understand if the requirements are fulfilled, can be difficult: it is for example not possible to guarantee the fulfillment of quality attributes of the final system based on software architecture design [7]. This would imply that the detailed design and implementation represent a strict projection of the architecture [8]. E.g., “scalability”: just because an architecture is designed as scalable, this does not mean that any possible implementation is automatically scalable.

This shows the limits of the validation of software architectures: it is very hard to proof the fulfillment of requirements. Instead, it is feasible to **evaluate the architectural decisions represented by the software architecture** and to assess **if these decisions support or obstruct the implementation of the requirements.**

This rest of this document is structured as follows: section 4 - “Terminology” – defines a set of key words that are used throughout this document, section 5 – “State of the art in architectural assessment” – summarizes the state of the art in architectural assessment, section 6 presents the ATAM steps that will be performed in this assessment, and section 7 gives a conclusion.

4 TERMINOLOGY

This document is about validation and assessment. These terms convey different meanings when used in different contexts, which makes it necessary to state how they are understood within this text.

Validation: According to the IEEE Standard 1012-1998 [5], software validation is the "confirmation by examination and provisions of objective evidence that specified requirements have been fulfilled."

Assessment: In the context of this deliverable, "assessment" refers to the validation of the requirements stated in the deliverable D10.1 (Requirements report). The goal of the assessment is the validation, i.e., to demonstrate that specified requirements are sufficiently supported by the reference architecture.

Functional requirement: Functional requirements specify the functions of the system, how it records, computes, transforms, and transmits data [1].

Non-functional requirement (quality requirement): Quality requirements specify how well the system performs its intended functions [1].

Architectural property: A statement about a part of the reference architecture.

5 State of the art in architectural assessment

Eicker et al. [10] identified eighteen methods to assessment software architectures:

1. SAAM: Software Architecture Analysis Method [11]
2. ATAM: Architecture Analysis Tradeoff Method [12]
3. SBAR: Scenario-Based Architecture Reengineering [13]
4. CBAM: Cost Benefit Analysis Method [14]
5. SACAM: Software architecture Comparison Analysis Method [15]
6. ARID: Active Reviews for Intermediate Design [16]
7. SAAMCS: Software Architecture Analysis Method for Complex Scenarios [17]
8. ESAAMI/ISAAMCR: Extending SAAM by Integration in the Domain/Integrating SAAM in Domain-centric and Reuse-based Development Processes [18]
9. SAAMER: Software Architecture Analysis method for Evolution and reusability [19]
10. FAAM: Family Architecture Assessment Method [20]
11. ALPSM: Architecture Level Prediction of Software Maintenance [21]
12. PASA: Performance Assessment of Software Architecture (see [22])
13. ALMA: Architecture-Level Modifiability Analysis [23]
14. QAW: Quality Attribute Workshop [23]
15. SAEM: Software Architecture Evaluation Model [25]
16. AQA: Architecture Quality Assessment [24]
17. ABAS: Attribute-Based Architectural Styles [26]

The different methods can be analyzed considering two relationship types [10]:

- "Association": some aspects of one method where integrated into a new method or through a combination of several parts of different methods a new method was created
- "Inheritance": the initial method remains generally the same, one or more aspects were refined or specialized

The following picture shows assessment methods that were related by an "association" relationship, the labels next to arrows indicate the aspect that has been taken over by the associated method (the arrow shows in which direction this occurred). Italic labels next to the method names indicate new elements that were introduced in that method.

The methods ATAM, SBAR, CBAM, SACAM, and ARID build on scenarios to identify quality attributes, and are therefore related to SAAM.

A SAAM's goal is to verify basic architectural assumptions and principles against the documents describing the desired properties of an application. Scenarios represent the starting point for analyzing the properties of a software architecture. They illustrate the kinds of activities that the system must support and the kinds of anticipated changes that will be made to the system [24].

The ATAM method assesses an architecture with respect to multiple competing quality attributes focusing on "sensitivity points," i.e., properties of a component that are critical to the success of system, and trade-off points, i.e., properties that affect more than one attribute or sensitivity points.

SBAR uses scenarios, simulation, mathematical modeling, and experience-based reasoning to estimate the potential of the designed architecture to reach the software quality requirements [8].

CBAM is a method that - together with the assessment of quality attributes - also analyses the software architecture in respect to its cost effectiveness [10].

SACAM is used to compare different architectural alternatives analyzing their suitability for a set of scenarios. SACAM defines criteria that represent the degree of how good a given architecture candidate matches the requirement of a given scenario [10].

ARID is a combination of the scenario technique proposed in ATAM and Active Design Reviews [28] and is intended to assess partial architectural drafts during an early stage of development [10].

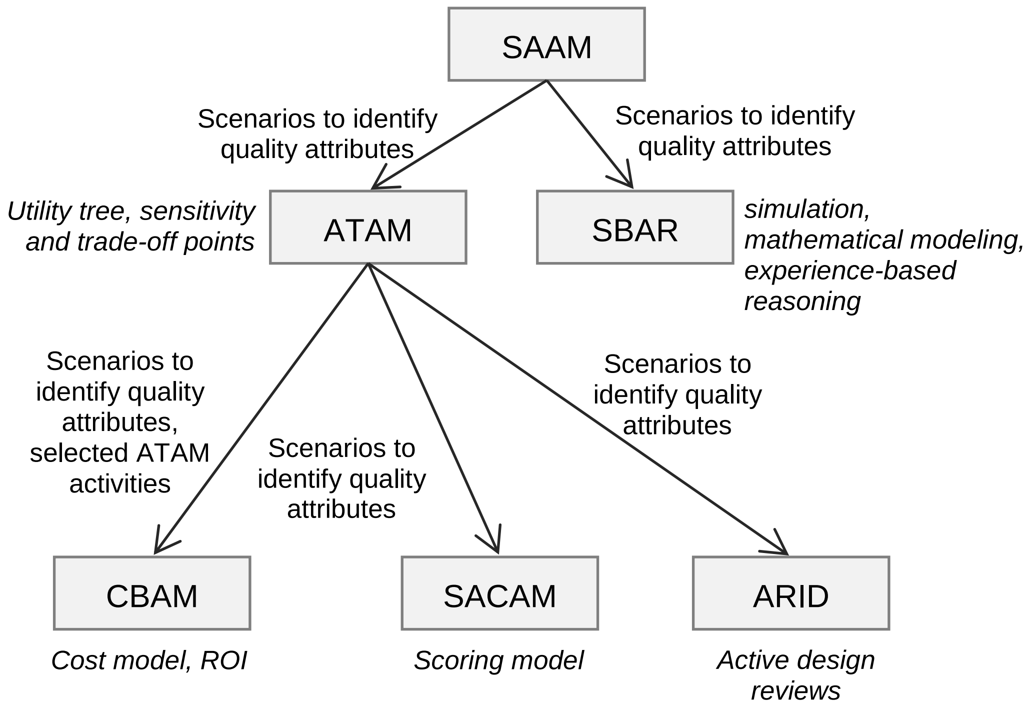
SAAM
Scenarios to identify quality attributes
Scenarios to identify quality attributes
Utility tree, sensitivity and trade-off points
ATAM
SBAR
simulation, mathematical modeling, experience-based reasoning
Scenarios to identify quality attributes, selected ATAM activities
Scenarios to identify quality attributes
Scenarios to identify quality attributes
CBAM
SACAM
ARID
Cost model, ROI
Scoring model
Active design reviews

Figure 1: Association relationships between architectural assessment methods (adapted from [10])

The following figure shows the inheritance relationship, i.e., methods that extend one or more aspects of other methods. SAAMCS, ESAAMI/ISAMCR, SAAMER, FAAM, ALPSM, PASA, and ALMA extend or refine SAAM and are therefore shown with an inheritance relationship. QAW is a specialization of ATAM.

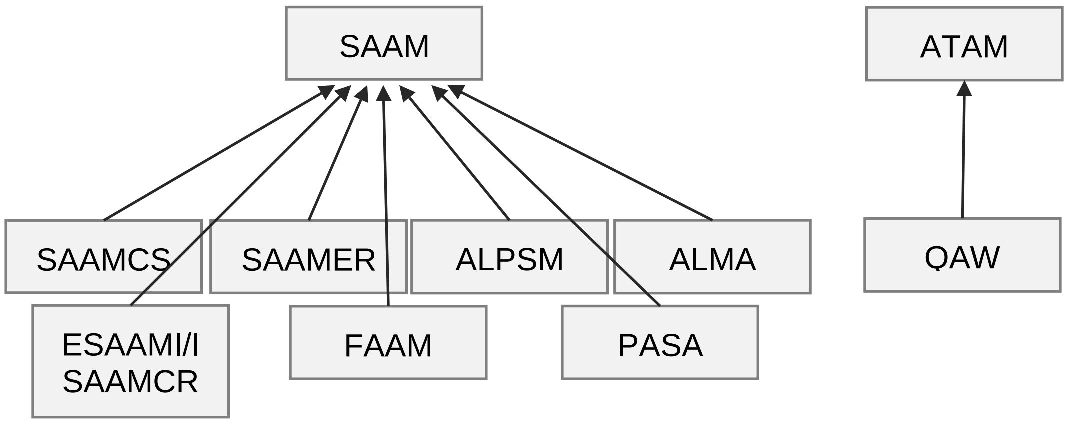

Figure 2: Inheritance relationships between architectural assessment methods [10]

The methods ALMA, FAAM, SAAMCS, ALMA, and PASA analyse particular quality attributes in detail [10]:

- focussing on the extension of a system
    - ALMA: Modifiability
    - FAAM: Extensibility
    - SAAMCS: Flexibility
    - ALPSM: Maintainability
    - PASA: Performance
- Focussing on the extension and on the maintenance of a system:
    - SAAMER: Evolution and reusability [8]
    - ESAAMI/ISAAMCR: Reuse [8]

The methods SAAM, AQA, ABAS, and SAEM are independent methods and are not based on other methods [10].

AQA studies aspects of maintainability, considering the extension of a system and also its maintenance [8].

ABAS helps to move from the notion of architectural styles toward the ability to reason based on quality attribute-specific models [8].

SAEM defines metrics of quality based on the goal-question-metric [9] technique. The goal of metrics is to discover whether certain attributes meet the values specified in the quality specification for each software characteristic [8].

The QAW is a facilitated, early intervention method used to generate, prioritize, and refine quality attribute scenarios before the software architecture is completed [23].

5.1 Assessment method selection

The selection of the method to assess the NEXOF reference architecture was based on the results described in [8] and [10] and on the selected assessment method in the “Proof of Concept” track.

The following methods were excluded:

1. methods that **focus on a single quality attribute** [8]: SAAM, SAAMCS, SAAMER, ALPSM, ALMA, ESAAMI/ISAAMCR, FAAM, PASA, ARID, CBAM, and AQA.
2. methods that **evaluate different alternatives** instead of one: In our case we do not have several architectural alternatives, therefore SACAM was also excluded
3. methods that **expect a fully implemented architecture** [10]: SAEM, AQA, and SBAR
4. methods that **depend on the availability of source code**: SAEM and AQA
5. methods that **do not assess adherence to quality requirements**: ABAS assesses the architecture to extract patterns that can be used as building blocks for designing software architectures, the QAW method is an approach for discovering quality attributes before the software architecture is created. Both methods support other methods, e.g., ABAS is mentioned in [12] to support ATAM, QAW in [15] to support SACAM to extract the needed information to perform the assessment.

For the afore mentioned reasons, also since ATAM was already used in the context of evaluating a reference architecture by the Software Engineering Institute (see [27]), and also **to align our work with the “Proof of Concept” track** we choose ATAM as the assessment method.

6 ATAM ASSESSMENT OF THE NEXOF REFERENCE ARCHITECTURE

6.1 Step 1: Present the ATAM

In this step the evaluation team lead presents the ATAM to the assembled stakeholders.

The ATAM is performed within 9 steps [12]:

- Presentation
  1. **Present the ATAM**: the method is presented to the stakeholders
  2. **Present business drivers**: the primary architectural drivers are presented
  3. **Present architecture**: the proposed architecture, focusing on how it addresses the business drivers it presented
- Investigation and Analysis
  4. **Identify architectural approaches**: architectural approaches are identified but not analyzed
  5. **Generate quality attribute utility tree**: the quality factors that comprise system utility are elicited, specified, annotated with stimuli and responses, and prioritized.
  6. **Analyze architectural approaches**: Based upon the high-priority factors identified in the previous step, the architectural approaches that address those factors are elicited and analyzed
- Testing
  7. **Brainstorm and prioritize scenarios**[2]: an additional set of scenarios is elicited from the entire group of stakeholders. This set of scenarios is prioritized via a voting process involving the entire stakeholder group.
  8. **Analyze architectural approaches**: this step reiterates step 6, but here the highly ranked scenarios from Step 7 are considered to be test cases for the analysis of the architectural approaches determined thus far.
- Reporting
  9. **Present results**: Based upon the information collected in the ATAM (styles, scenarios, attribute-specific questions, the utility tree, risks, sensitivity points, tradeoffs) the ATAM team presents the findings to the assembled stakeholders and potentially writes a report detailing this information along with any proposed mitigation strategies.

[2] In ATAM the term “scenario” is defined as “short statement describing an interaction of one of the stakeholders with the system”, which differs from the definition used in the requirements report (D10.1), in which scenarios are “facts describing an existing system and its environment including the behaviors of agents and sufficient context information to allow discovery and validation of system requirements”. In ATAM, a “scenario” if described from a user perspective “would very much resemble use cases in object-oriented parlance [12]”.

As suggested in [12], we divide the ATAM process into two phases: the first phase (steps 1-5) is about gathering information and to define what will be assessed, how. The second phase (steps 6-9) performs the evaluation itself.

The output of the evaluation will be "the scenarios elicited and prioritized, the questions used to understand/evaluate the architecture, a “utility” tree, describing and prioritizing the driving architectural requirements, the set of identified architectural approaches and styles, the set of risks and non-risks discovered, the set of sensitivity points and tradeoffs discovered [12]".

This deliverable will report the results of phase 1, D10.2 ("Assessment results") will report the results of phase 2.

6.2 Step 2: Present business drivers

In this step a system overview from a business perspective is given, i.e., "its most important functional requirements, its technical, managerial, economic, or political constraints, its business goals and context, its major stakeholders, the major quality attribute goals that shape the architecture [12]."

The NEXOF reference architecture is the reference architecture for a generic open platform for creating and delivering applications enabling the creation of service based ecosystems where service providers and third parties easy collaborate [32].

The overall goal of NEXOF-RA is independence such that NEXOF can be implemented into a broad range of application domains supporting any business size by all user communities using different technologies [32]. In other words, the NEXOF reference architecture is designed with the goal to be technology neutral, application domain independent, and business-scale independent.

The requirements report [31], contains 21 possible applications of service oriented architectures which are provided in form of scenarios[3] including their constraints, stakeholders, business goals and context. The 21 scenarios were used to generate 42 high level requirements, and finally the system requirements, used to define the reference architecture model.

In the context of architectural assessment the 21 scenarios will be used to assess whether the proposed reference architecture is able to fulfill the scenarios, i.e., as test cases for the reference architecture.

As major stakeholder groups we consider service consumers, service providers, and service integrators or developers.

[3] Scenarios now intended as "facts describing an existing system and its environment including the behaviours of agents and sufficient context information to allow discovery and validation of system requirements".

6.3 Step 3: Present architecture

In this step the architecture team presents the proposed architecture, focusing on how it addresses the business drivers.

The NEXOF-RA reference architecture is structured into the following parts:

- The standards catalog
- The component catalog, consisting of
    - Concrete components
    - Abstract components
- The architecture specification, consisting of
    - Top level patterns (e.g., Enterprise SOA)
    - Abstract design patterns
    - Implementation design patterns
- Guidelines and principles
- The conceptual model

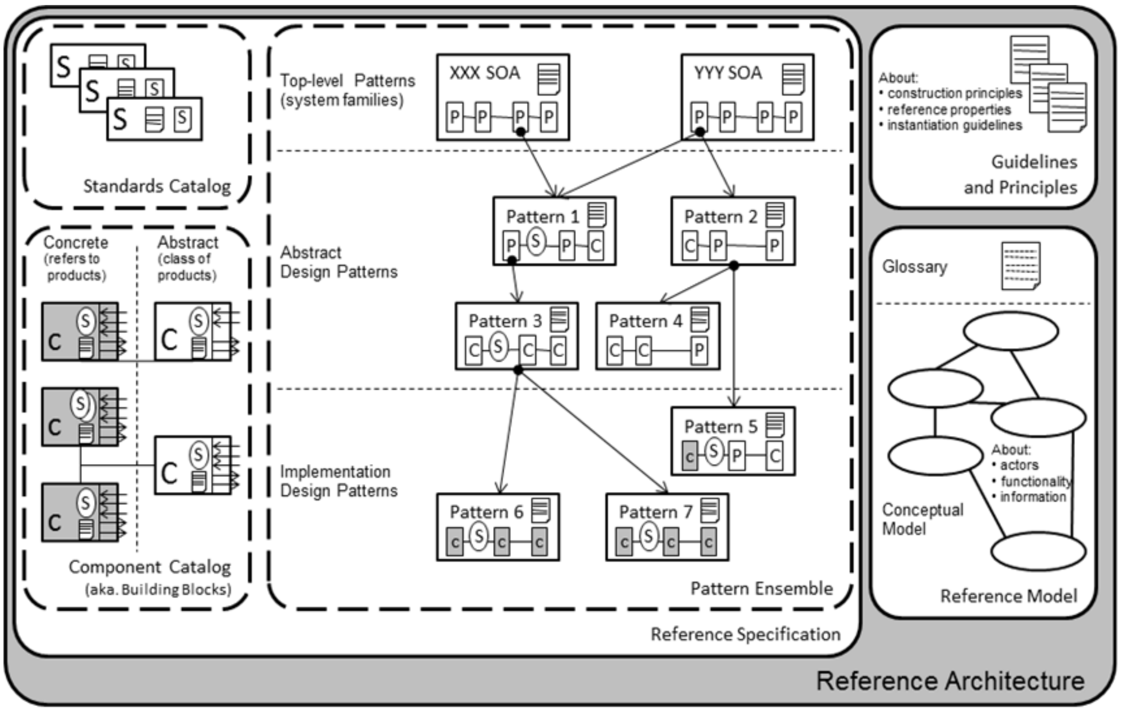

Figure 3: The NEXOF reference architecture

The parts are described in the deliverables shown in **table 1**.

The business drivers are addressed by the different components of the architecture:

- functional requirements, functional constraints, and stakeholders – in form of agents (entities of the real world) and actors (roles the real world entities may play) – are addressed in the reference model.

- quality requirements and quality constraints are addressed in the reference specification and influence the design decisions that define the patterns.

Table 1: The main deliverables used during the assessment

Part of the reference architecture	D7.2, "Definition of an architectural framework and principles"	D6.1, "NEXOF Reference Architecture Model"	D6.2, "NEXOF Reference Architecture Model"	D7.5 "RA Specification"	D7.1 "State of the art report"
Guidelines and principles about structure, properties, and instantiation guidelines	●				
The conceptual model		●	●		
Top-level patterns of architecture specification				●	
Abstract design patterns of the architecture specification				●	
Abstract components of the component catalog				●	
Implementation design patterns of the architecture specification				●	
Concrete components of the component catalog				●	
The standards catalog				●	●

6.4 Step 4: Identify architectural approaches

Architectural approaches define the important structures of the system and describe the ways in which the system can grow, respond to changes, withstand attacks, integrate with other systems, and so forth [12].

In our case the architectural approaches are formed by the patterns and it's building blocks, the components. We consider only those patterns that are at least in the formally-consistent state (see [30]) and omit patterns that are in the in-conception or in-elaboration state. The patterns are [30]:

- Internet of Services
  - Assisted Composition Designer

- Authorization
- IaaS
- Internet of Service
- Mash up as a service (MaaS)
- Multi-phase Discovery
- Service Discovery
- Service Matchmaking and Ranking
- Template-based Discovery
- Trust Based model registry

* Cloud computing
    - Authorization
    - Cloud migration enabled by OSGi – step one
    - Cloudified Application Servers
    - Elastic and Reliable Filesystems
    - IaaS
    - Map-Reduce
    - NoSQL Storage
    - Platform as a Service
* Enterprise SOA
    - Active Replication
    - Assisted Composition Designer
    - Authorization
    - Black-Box DB Replication
    - Cloud migration enabled by OSGi – step one
    - Designer and Runtime Tools for E-SOA
    - Distributed ESB in E-SOA
    - Enterprise SOA
    - Front End in E-SOA
    - Generic Group Communication
    - Gray-Box DB Replication
    - Horizontal Replication with Replication Awareness
    - IaaS
    - Log Mining Writeset Extraction
    - Models Manager
    - Monitoring in E-SOA
    - Multicast-Based Replica Discovery
    - Multi-phase Discovery
    - Multi Tenancy Enabler
    - Multi-Tier Transactional Service Runtime
    - Non-Repudiation
    - OSGi-based SCA Container
    - Reflective Database Replication
    - Registry-Based Replica Discovery
    - Security in E-SOA
    - Service Discovery
    - Service Matchmaking and Ranking
    - Session Replication with multi-tier coordination
    - Template-based Discovery
    - Transparent Replication Proxy

- Trigger Writeset Extraction
- Trusted Timestamping
- Vertical Replication
- Virtual ESB
- Virtualization of Computational Resources in E-SOA
- White-Box DB Replication
- Writeset Extraction Based on Extended Interfaces

6.5 Step 5: Generate Quality Attribute Tree (Define assessment criteria)

The quality attribute tree serves to "concretize the quality attribute requirements". This section will describe the quality attribute requirements using the requirements stated in the requirements report [31] identifying the key aspects that require validation.

The quality attribute goals considered relevant by the authors of each scenario are shown in **table 2**.

Table 2: Quality attributes relevant for each scenario

Scenario	Availability	Buildability	Integrability	Interoperability	Modifiability	Performance	Recoverability	Reliability	Resource efficiency	Reusability	Scalability	Security	Testability	Usability
S1	●			●	●	●		●				●		●
S2	●	●			●	●	●	●	●	●	●	●	●	●
S3	●			●				●				●		
S4	●							●						●
S5	●			●		●		●	●		●			●
S6		●	●	●	●		●				●		●	
S7	●		●	●		●		●			●	●		
S8	●		●	●				●				●		●
S9	●		●	●	●	●		●		●	●	●		
S10	●		●	●				●	●			●	●	
S11	●		●	●		●		●				●	●	
S12				●				●				●	●	●
S13			●	●								●		●
S14	●			●			●	●					●	●
S15					●							●	●	●
S16	●		●	●								●	●	●
S17	●		●	●		●		●						
S18	●	●	●	●	●	●								
S19					●							●	●	
S20	●	●			●	●		●			●	●	●	
S21	●				●	●					●	●	●	
Sum	16	4	10	15	9	10	3	14	3	2	7	15	11	10

Scenario	Availability	Buildability	Integrability	Interoperability	Modifiability	Performance	Recoverability	Reliability	Resource efficiency	Reusability	Scalability	Security	Testability	Usability
Rank	1	8	5	2	6	5	9	3	9	10	7	2	4	5

We use the number of scenarios for which each quality attribute was mentioned to be relevant as an indicator of priority, to obtain the following ranking of quality attributes, the most important first.

1. Availability
2. Interoperability, Security
3. Reliability
4. Testability
5. Integrability, Performance, Usability
6. Modifiability
7. Scalability
8. Buildability
9. Recoverability, Resource efficiency
10. Reusability

The quality attribute tree further elaborates the quality attributes into scenarios, in the ATAM case, intended as "a short statement describing an interaction of one of the stakeholders with the system".

For every quality attribute we provide a generalized scenario (one summarizing all scenarios of the current quality attribute) consisting of (see [33]):

- **Source of stimulus**. This is some entity (a human, a computer system, or any other actuator) that generated the stimulus.
- **Stimulus**. The stimulus is a condition that needs to be considered when it arrives at a system.
- **Environment**. The stimulus occurs within certain conditions.
- **Artifact**. Some artifact is stimulated. This may be the whole system or some pieces of it.
- **Response**. The response is the activity undertaken after the arrival of the stimulus.
- **Response measure**. When the response occurs, it should be measurable in some fashion so that the requirement can be tested.

We extract these ATAM-scenarios from the scenarios described in the requirements document. For reasons of readability, we show the first two levels of the utility tree in figure 4, the leafs (the scenarios for every quality attribute) in the following subsections.

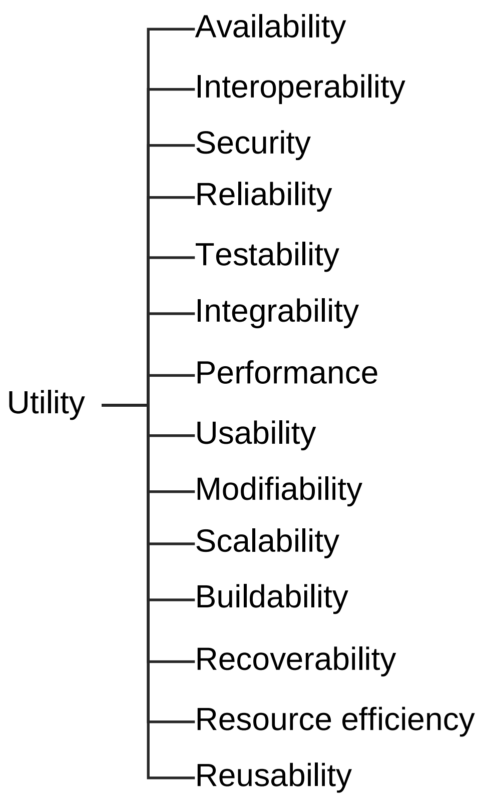

Figure 4: The first two levels of the ATAM utility tree considering the NEXOF-RA quality attributes

6.5.1 Availability

The scenarios for the availability quality attribute are as follows:

U1: Functionality is maintained with a high mean time between failure and a low mean time to recovery (S1)(S2)(S3)(S20)
U2: It is possible to achieve high availability through backup mechanisms (S4)
U3: High availability is achieved through a fail-secure and self-healing technical service platform (S4)(S7)(S8)(S9)(S14)(S17)
U4: High availability is achieved through self-configuration and adaptation (S14)
U5: High availability of access to computational resources (S5) is ensured
U6: High availability of access to data (S5)(S7)(S10)(S11) is ensured
U7: Reconfiguration does not disrupt functionality (S9)(S18)
U8: High availability is ensured also under high workload (S16)(S21)
U9: Safety is ensured (S14)(S4)(S7)(S9)

The **generalized scenario for availability scenarios (U1-U9)** is as follows:

- **Source of stimulus**: Internal or external

- **Stimulus**: Hardware or software fault, reconfiguration
- **Environment**: Normal, degraded, under high work load
- **Artifact**: Service
- **Response**: Mechanisms (self-healing, self-configuration, backup, or adaptation mechanisms) can be put in place to avoid the interruption to the access of computational resources or data, also ensuring safety (e.g., through the use of fail-secure mechanisms)
- **Response measure**: mean time between failure and mean time to recovery

6.5.2 Interoperability

The scenarios for the interoperability quality attribute are as follows:

U10: Through a high interoperability, vendor independency is achieved (S6)(S18)
U11: Interoperability of heterogeneous technical devices is possible (S7)(S8)(S9)(S10)(S11)(S14)(S17)
U12: Interoperability using heterogeneous data formats and protocols is possible (S7)(S8)(S9)(S10)(S11)(S14)(S17)(S5)
U13: Interoperability between organizations is possible (S12)(S16)(S3)
U14: Interoperability to implement ambient intelligence using protocols such as Wi-Fi or GSM is possible (S13)
U15: Services running on multiple different platforms can communicate (S18)
U16: Integration of heterogeneous information systems is possible (S1)(S18)(S5)

The **generalized scenario for interoperability scenarios (U10-U16)** is as follows:

- **Source of stimulus**: Internal, External
- **Stimulus**: Connection from/to an external system
- **Environment**: Normal
- **Artifact**: Application
- **Response**: Data exchange using heterogeneous data formats and protocols
- **Response measure**: Ability to interoperate with other applications, ability to extend an existing application to enhance interoperability

6.5.3 Security

The scenarios for the security quality attribute are as follows:

U17: Confidentiality is ensured (S1)(S2)(S3)(S7)(S8)(S15) (S16) (S21)
U18: The system can be locked to prevent tampering (S9)
U19: The security mechanisms address identified risks (S10) (S11)
U20: Encryption is used when transferring data (S12)(S15)

U21: There is a single entry point for authentication (S13) (S16)
U22: The integrity of data is ensured (S15)
U23: The authenticity of users is verified (S15) (S20)
U24: Data can be transmitted in a non-repudiable way (S15)
U25: The compliance to legal requirements can be verified (S19)(S21)
U26: It is possible to log the performed operations (S20)

The **generalized scenario for security scenarios (U17-U26)** is as follows:

- **Source of stimulus**: External system or user
- **Stimulus**: Access to data or computational resources
- **Environment**: Normal
- **Artifact**: Single entry point
- **Response**: Security mechanisms, selected and adapted according to the identified risks (e.g., encryption, nonrepudiation, confidentiality, integrity, authentication, etc.) also providing auditing functionality
- **Response measure**: provided functionality, legal compliance

6.5.4 Reliability

The scenarios for the interoperability quality attribute are as follows:

U27: Services can be managed against SLAs (S2)
U28: Services are provided reliably (S3) (S1)(S10) (S11) (S17)
U29: Safety is ensured (S4)(S7)(S9)(S8)(S14)(S20)
U30: Interruptions are avoided (S5)
U31: Monitoring and alerting are used to increase reliability (S12)

The **generalized scenario for reliability scenarios (U27-U31)** is as follows:

- **Source of stimulus**: External system
- **Stimulus**: Access to a service
- **Environment**: Normal
- **Artifact**: Service
- **Response**: Reliability and safety (in the context of reliability) ensuring mechanisms are put in place, such as a verification against SLAs or monitoring and alerting.
- **Response measure**: reliability ratio (percentage of reliably provided services over all services)

6.5.5 Testability

The scenarios for the testability quality attribute are as follows:

U32: Testing can be used to verify the correct working of the system before deployment (on design time) (S2)(S14)(S6) (S20)

U33: Testing can be used to verify the correct working of the system while it is running (on run time) (S15)(S16) (S20)
U34: The configuration can be verified through testing (S6)
U35: Simulation can be used to identify problems (S10) (S11)
U36: Ensure continuous operation (S12)
U37: Testing can be used to prove compliance to legal requirements (S19) (S21)
U38: Tests verify the compliance to SLAs (S21)

The **generalized scenario for testability scenarios (U32-U38)** is as follows:

- **Source of stimulus**: External system
- **Stimulus**: Access to a service auditing interface
- **Environment**: On runtime, on design time
- **Artifact**: Service auditing interface
- **Response**: Access to auditing data, simulation of user behavior
- **Response measure**: ability to demonstrate faults through testing, path coverage, configuration testing

6.5.6 Integrability

The scenarios for the integrability quality attribute are as follows:

U39: Through a high Integrability, vendor independency is achieved (S6)(S18)
U40: Integration of heterogeneous technical devices is possible (S7)(S8)(S9)(S10)(S11)(S17)
U41: Integration of data coming from different sources is possible (S7)(S8)(S9)(S10)(S11)(S13)
U42: Integration of heterogeneous information systems is possible (S16)(S17)

The **generalized scenario for integrability scenarios (U39-U42)** is as follows:

- **Source of stimulus**: Internal, External
- **Stimulus**: Integration of functionality between internal and external system
- **Environment**: Normal
- **Artifact**: Application
- **Response**: Data exchange using heterogeneous data formats and protocols
- **Response measure**: Ability to integrate with other applications, ability to extend an existing application to enhance integrability

6.5.7 Performance

The scenarios for the performance quality attribute are as follows:

U43: A service has a timely response (S1)(S5)(S11)(S18)(S20)
U44: Services have a low management overhead (S2)
U45: The platform can cope with large data (S7)(S17)
U46: The platform can identify the needed reusable component and timely deploy it (S9)
U47: The platform can cope with many users (S21)

The **generalized scenario for performance scenarios (U43-U47)** is as follows:

- **Source of stimulus**: External
- **Stimulus**: Invocation of service
- **Environment**: Normal
- **Artifact**: Service
- **Response**: Requested functionality, not degrading with increasing data size and amount of users
- **Response measure**: response time, throughput

6.5.8 Usability

The scenarios for the usability quality attribute are as follows:

U48: The user interface is adapted to the needs of the users of the system (S1)(S2)(S4)(S5)(S8)(S13)(S15)(S16)
U49: Services can be integrated in a graphical way (S12)
U50: Human-based process steps can be executed in combination with automated steps and work flows (S14)

The **generalized scenario for usability scenarios (U48-U50)** is as follows:

- **Source of stimulus**: End user
- **Stimulus**: Invocation of service
- **Environment**: At runtime, at design time
- **Artifact**: System
- **Response**: provide requested functionality, integrating human-based process steps
- **Response measure**: task time, number of errors, number of problems solved, user satisfaction, gain of user knowledge, ratio of successful operations to total operations, or amount of time/data lost when an error occurs

6.5.9 Modifiability

The scenarios for the modifiability quality attribute are as follows:

U51: Addition, modification, removal of services is possible (S1)
U52: Accommodate to specific configuration requirements is possible (S2)(S6)(S9)(S21)
U53: Accommodate to specific deployment requirements is possible (S6)(S9)
U54: Services can be ported between different environments (S18)
U55: Compliance to legal requirements is achieved through modifications (S15)(S19)(S20)(S21)

The **generalized scenario for modifiability scenarios (U51-U55)** is as follows:

- **Source of stimulus**: User (developer, administrator, end user)
- **Stimulus**: Changes (addition, removal, modification, deployment, porting, configuration of service) to be made
- **Environment**: At runtime, at design time
- **Artifact**: Part of system to be changed
- **Response**: Modification of the system with verification of side effects (testing)
- **Response measure**: time to perform modification, time to demonstrate correct behaviour after modification (also legal compliance)

6.5.10 Scalability

The scenarios for the scalability quality attribute are as follows:

U56: The number of users is scalable (S2)(S5)(S7) (S20)(S21)
U57: The data size is scalable (S2)(S5)(S7)(S21)
U58: The geographical distribution is scalable (S2)
U59: The different approaches for deployment are scalable (S6)
U60: The number of involved systems is scalable (S9)(S20)

The **generalized scenario for scalability scenarios (U56-U60)** is as follows:

- **Source of stimulus**: User (developer, administrator)
- **Stimulus**: Scalability requirement
- **Environment**: At design time
- **Artifact**: System
- **Response**: System can be adapted to support a different amount of users, data size, communication needs, number of involved systems
- **Response measure**: time to adapt scalability

6.5.11 Buildability

The scenarios for the buildability quality attribute are as follows:

U61: Assistance to create services is provided (S2)(S6)(S20)
U62: Assistance to deploy services is provided (S6)
U63: Assistance to migrate services to new platforms is provided (S18)

The **generalized scenario for buildability scenarios (U61-U63)** is as follows:

- **Source of stimulus**: Developer, build system
- **Stimulus**: Building of system, design of system
- **Environment**: At runtime, at design time
- **Artifact**: System
- **Response**: Interfaces to build system
- **Response measure**: time to build, modularity (degree of dependencies between objects)

6.5.12 Recoverability

The scenarios for the recoverability quality attribute are as follows:

U64: It is possible to preserve session and lifecycle state in the event of failure (S2)
U65: It is possible to preserve configuration (S6)
U66: Safety is ensured (S14)

The **generalized scenario for recoverability scenarios (U64-U66)** is as follows:

- **Source of stimulus**: Internal or external
- **Stimulus**: Hardware or software fault, reconfiguration
- **Environment**: Normal, degraded, under high work load
- **Artifact**: Service
- **Response**: Mechanisms that can be put in place to recover the system state (session and lifecycle state, configuration) in the event of failure, also ensuring safety (e.g., through the use of fail-secure mechanisms)
- **Response measure**: mean time between failure and mean time to recovery

6.5.13 Resource efficiency

The scenarios for the resource efficiency quality attribute are as follows:

U67: Services have a low management overhead (S2)
U68: Interruptions are avoided (S5)
U69: The flow of goods, information and resources are optimized (S10)

The **generalized scenario for resource efficiency scenarios (U67-U69)** is as follows:

- **Source of stimulus**: External system, user
- **Stimulus**: Invocation of service, management of services
- **Environment**: At runtime, at design time
- **Artifact**: System
- **Response**: provide requested functionality, minimize overhead, avoiding interruptions, optimizing the flow of goods, information and resources
- **Response measure**: task time, memory requirements, efficiency

6.5.14 Reusability

The scenarios for the reusability quality attribute are as follows:

U70: Components can be reused (S2)
U71: Strategies can be reused (S9)

The **generalized scenario for resource reusability scenarios (U70-U71)** is as follows:

- **Source of stimulus**: Developer
- **Stimulus**: Reuse of a component
- **Environment**: At runtime, at design time
- **Artifact**: Component, Source code
- **Response**: Evaluate the behaviour of a component, reuse components, strategies on runtime and design time
- **Response measure**: time to reuse, modularity (degree of dependencies between objects), time to validate and verify reused component

7 CONCLUSION

The goal of this document is to define the assessment criteria that will be used to validate the outcome of NEXOF-RA, the NEXOF reference architecture, against the requirements specified within D10.1 “Requirements Report”.

The assessment criteria are represented by the ATAM scenarios of the ATAM utility tree described from section 6.5.1 to 6.5.14.

The approach described in this document uses ATAM [12] to generate assessment criteria (in ATAM the “quality attribute tree”). In D10.2 “Assessment results”, this quality attribute tree will be evaluated using the identified architectural approaches.

8 REFERENCES

[1] Lauesen, Soren. 2002. Software Requirements: Styles & Techniques. Addison-Wesley Professional, January 26.
[2] Crnkovic, I. and M. P. H. Larsson, *Building Reliable Component-based Software Systems,* Artech House, 2002
[3] You, C., Zhou, J., and Lam, K. 1998. On the efficient implementation of fair non-repudiation. SIGCOMM Comput. Commun. Rev. 28, 5 (Oct. 1998), 50-60.
[4] Onieva, J. A., Zhou, J., and Lopez, J. 2008. Multiparty nonrepudiation: A survey. ACM Comput. Surv. 41, 1 (Dec. 2008), 1-43.
[5] Software Engineering Standards Committee of the IEEE Computer Society, Standard for Software Verification and Validation, IEEE Std 1012-1998, 1998.
[6] Software Engineering Standards Committee of the IEEE Computer Society, Recommended Practice for Architecture Description of Software-Intensive Systems, IEEE 1471-2000, 2000.
[7] Bosch J. and P. Molin, "Software Architecture Design: Evaluation and Transformation," *Proc. IEEE Eng. of Computer Based Systems Symp. (ECBS `99),* Dec. 1999.
[8] Dobrica, L., and E. Niemela, "A survey on software architecture analysis methods, " *IEEE Transactions on Software Engineering,* Volume 28, Issue 7, Jul 2002 Page(s): 638 - 653
[9] Basili, V, G. Caldiera, H. D. Rombach. "The Goal Question Metric Approach," In: Encyclopedia of Software Engineering. John Wiley & Sons, 1994, pp. 528–532.
[10] Eicker, S., Hegmanns C, Malich, S. "Auswahl von Bewertungsmethoden für Softwarearchitecturen," ICB-Research Report No. 14, Universität Duisburg Essen, March 2007
[11] Kazman, R., Bass, L., Abowd, G., Webb M. "SAAM: A Method for Analyzing the Properties of Software Architectures," Proceedings of the 16th International Conference on Software Engineering, 1994
[12] Kazman, R., Klein, M., Barbacci, M., Longstaff, T., Lipson, H., Carriere, H. "The Architecture Tradeoff Analysis method," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, 1997, available online: http://www.sei.cmu.edu/library/abstracts/reports/00tr004.cfm
[13] Bengtsson, P., Bosch, J. "Scenario-Based Software Architecture Reeingineering," Proceedings of the 5th International Conference on Software reuse. IEEE Computer Socienty Press, 1998
[14] Kazman, R., Asundi, J., Klein, M. "Making Architecture Design Decisions: An Economic Approach," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, 2002, available online: http://www.sei.cmu.edu/reports/02tr035.pdf
[15] Stoemer, C., Bachmann, F., Verhoef, C. "SACAM: The Software Architecture Comparison Analysis method," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, 2003, available online: http://www.sei.cmu.edu/library/abstracts/reports/03tr006.cfm

[16] Clements, P. “Active Reviews for Intermediate Design.” Software Engineering Institute, Carnegie Mellon University, Pittsburgh, 2000, available online: http://www.sei.cmu.edu/reports/00tn009.pdf
[17] Lassing, N., Rijsenbrij, D., van Vliet, H. “On Software Architecture Analysis of Flexibility. Complexity of Changes: Size isn’t Everything.” Proceedings of the 2nd Nordic Workshop on Software Architecture, Ronneby 1999
[18] Molter, G. “Integrating SAAM in Domain-Centric and reuse-based Development Processes.” Proceedings of the 2nd Nordic Workshop on Software Architecture, Ronneby 1999
[19] Lung, C., Bot, S., Kalaichelvan, K., Kazman, R. „An Approach to Software Architecture Analysis for Evolution and Reusability.“ Proceedings Cascon, 1997
[20] Dolan, T. J. “Architecture Assessment of Information-Systems Families, a Practical Perspective.” Library Technische Universiteit Eindhoven, Proefschrift 2002
[21] Bengtsson, P., Bosch, J. “Architecture Level prediction of Software Maintenance.” Proceedings of the 3rd European Conference on Maintenance and reengineering, IEEE Computer Society Press 1999
[22] Thiel, S. “A Framework to Improve the Archtiecture Quality of Software-intensive Systems. Dissertation, Universität Duisburg-Essen, Essen 2005
[23] Barbacci, M., Ellison, R., Stafford, J., Weinstock, C., Wood, W. “Quality Attribute Workshops.” Software Engineering Institute, Carnegie Mellon University, Pittsburgh, 2001, available online: http://www.sei.cmu.edu/reports/03tr016.pdf
[24] Hilliard II, R. F., Kurland, M. J. Litvintchouk, S. D. „MITRE’S Architecture Quality Assessment.“ Proceedings of Software Engineering & Economics Conference, 1997
[25] Duenas, J.C. de Oliveira, W.L., de la Puente, J.A. “A Software Architecture Evaluation Model,” Proceedings of the 2nd International ESPRIT ARES Workshop, 1998
[26] Klein, M., Kazman, R. “Attribute-Based Architectural Styles.” Software Engineering Institute, Carnegie Mellon University, Pittsburgh, 1999, available online: http://www.sei.cmu.edu/library/abstracts/reports/99tr022.cfm
[27] Gallagher, B. “Using the Architecture Tradeoff Analysis Method to Evaluate a Reference Architecture: A Case Study,” Software Engineering Institute, Carnegie Mellon University, Pittsburgh, 1999, available online: http://www.sei.cmu.edu/reports/00tn007.pdf
[28] Parnas, D., Weiss, D. “Active Design Reviews: Principles and Practice,” Proceedings of the Eight International Conference on Software Engineering, 1985
[29] NEXOF-RA, 2009. Quality Model for NEXOF-RA Pattern Designing, Report.
[30] NEXOF-RA, 2009. RA Specification V1.0, Deliverable D7.5b (Pattern Compass)
[31] NEXOF-RA, 2010. Requirements Report, Deliverable D10.1b
[32] NEXOF-RA, 2010, Website, available online: http://www.nexof-ra.eu
[33] Bass, L., Clements, P., Kazman R. “Software Architecture in Practice.” Second Edition. Addison Wesley, 2003