Evaluating a Cloud-Based Reactive Architecture using *cloud-ATAM*

David Ebo Adjepon-Yamoah

Centre for Software Reliability
School of Computing Science
Newcastle University

Stakeholders' Group Work (July 2016)

Outline

Motivation

Present the ATAM/cloud-ATAM

Present the Business Drivers

Present the Architecture

Identify the Architectural Approaches

Generate the Quality Attribute Utility Tree and Scenarios

Analyse the Architectural Approaches

Present the Results

WHY?

Motivation

- The existing **architecture evaluation methods** have **limitations** when assessing architectures **interfacing** with **unpredictable environments** such as the *Cloud*.
- It is however, important to focus on holistic approaches combining aspects of both *dynamic* and *static* analysis of **architecture dependability attributes**.
- This exercise introduces an ATAM derived methodology - **cloud-ATAM** - for evaluating the **trade-off** between multiple dependability quality attributes (i.e. *availability* and *performance*) of a cloud-based *Reactive Architecture*.

Why Early Phases to System Development?

Present the ATAM/cloud-ATAM

- Identify potential **risks**, and verify the **quality** before building.
- All *design* involves **trade-offs**.
- A *software architecture* is the earliest life-cycle artefact that embodies significant **design decisions**: choices and trade-offs.

System Quality Attributes

Present the ATAM/cloud-ATAM

- Non-functional characteristics
- Desired combination of system *attributes*/*drivers*:
  1. *Maintainability*:
     corrections, improvements, adaptation
  2. *Modifiability*:
     change quickly and cost effectively
     (Extensibility, portability, restructuring)
  3. *Flexibility*:
     ease of modification for case NOT designed
  4. *Availability*:
     ratio of time a system is functional as expected
  5. *Performance*:
     total effectiveness of a system, ...

ⓘ: [Keller 1990], [Avizienis et al 2004] are very good starting places for quality attributes.

Architecture Evaluation Method

Present the ATAM/cloud-ATAM

We need a method in which the **right questions** are asked early to:

- Discover **sensitivity points** – alternatives for which a slight change makes a significant difference in some quality attribute
- Discover **risks** – alternatives that might create future problems in some quality attribute
- Discover **trade-offs** – decisions affecting more than one quality attribute

Architecture Evaluation Methods Comparison

Present the ATAM/cloud-ATAM

Comparison elements → / Evaluation Method	ATAM (Kazman et al, 1999)	SAAM (Kazman et al, 1994)	ALMA (Lassing et al, 1999 and Bengtsson et al, 2004)	SAAMCS (Lassing et al, 1999)	SBAR (Bengtsson and Bosch, 1998)	ALPSM (Bengtsson and Bosch, 1999)	ESAAMI (G. Molter, 1999)	SACAM (Bergey et al, 1999)
Method's Goal	Sensitivity and Trade-off analysis	Risk identification and architectural suitability	Maintenance cost prediction, risk assessment, SA selection	Risk assessment, developing complex scenarios for flexibility quality attribute	SA re-engineering to achieve quality attributes	Prediction of software maintainability	Integrates SAAM in a reuse based development & domain centric process	Comparison of several software architectures from different domains
Evaluation Approaches	Integrates questioning and measuring techniques	Scenario based functionality and change analysis	Depends on analysis objectives	Scenario-based (for complex scenarios)	Multiple approaches	Scenario-based	Preparation of analysis template to collect protoscenario	Collect comparison criteria
Quality Attributes	Multiple quality attributes	Modifiability	Maintainability	Flexibility	Multiple quality attributes	Maintainability	Modifiability	Multiple quality attributes
Applicable Project Stage	After SA /detailed or iterative improvement process	Once functions assigned to modules	During design phase	In the final version of the software architecture	System extension or re-engineering stage	During architectural design to predict perfective software maintenance	In the final version of the software architecture	During architectural design
Method's Activities	Six activities in four phases	Six activities with some activities in parallel	Five activities executed sequentially	Three activities, two in parallel executed	Three activities carried out iteratively	Six activities	Same as SAAM but use reusable knowledge base	Six activities, one executed repeatedly
Stakeholders' Involvement	All relevant stakeholders	All major stakeholders	Various for different activities	All major stakeholders	Software architect	Designer	All	Designer
Method's Application	Remote temperature sensor, Battlefield Control System and many more	GIS, WRCS, keyword in context (KWIC), and embedded audio system	Telecommunication systems, information systems, and embedded systems	Various business information system	Fire-alarm systems, measurement systems, and dialysis systems	Haemodialysis machine	Measurement system	Envisioned software proposed by various contractors
Tool Support	Not available	Partial, Ex:SAAMTOOL	Not available	Not available	Not available	Not available	Not available	Not available

Figure: Comparison of Architecture Evaluation Methods

Architecture Trade-off Analysis Method (ATAM)

Present the ATAM/cloud-ATAM

- Proposed by Kazman as a technique for understanding the **trade-offs points** or dependencies among the **attributes**, inherent to architecture **evaluation**.

  "The purpose of the ATAM is to assess the consequences of architectural decision alternatives in light of quality attribute requirements [Kazman 1998]"

- Provides a way to **evaluate software architecture's fitness** with respect to **multiple competing quality attributes**.
- Since these attributes interact, the method helps to **reason about architectural decisions** that affect **quality attribute interactions**.
- Follows a **spiral model of design**, postulating candidate architectures followed by analysis and risk mitigation, leading to refined architectures.

Spiral ATAM Model

Present the ATAM/cloud-ATAM

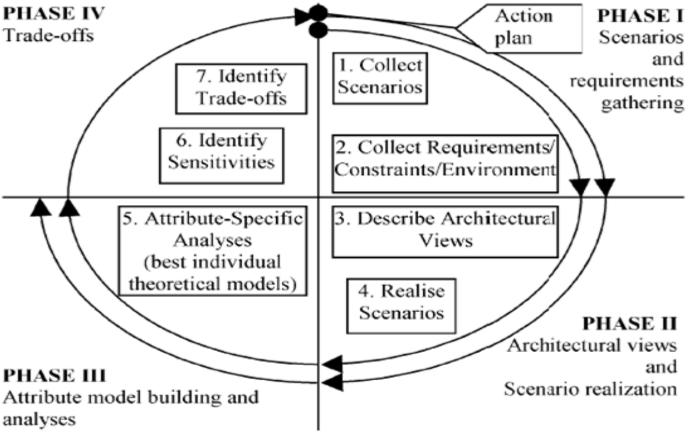

Figure: Spiral ATAM Model

Overview of ATAM

Present the ATAM/cloud-ATAM

- The purpose of an ATAM is **NOT to provide precise analyses**. The purpose IS to **discover risks** created by **architectural decisions**.
- We want to find trends: correlation between architectural decisions and predictions of system properties.
- Discovered risks can then be made the focus of mitigation activities: e.g. further design, further analysis, prototyping.
- **Examples of ATAM Projects:** US DOD Battlefield Control System, NASA Earthquake Monitoring System, etc.

ATAM: Scenario-Based Approach

Present the ATAM/cloud-ATAM

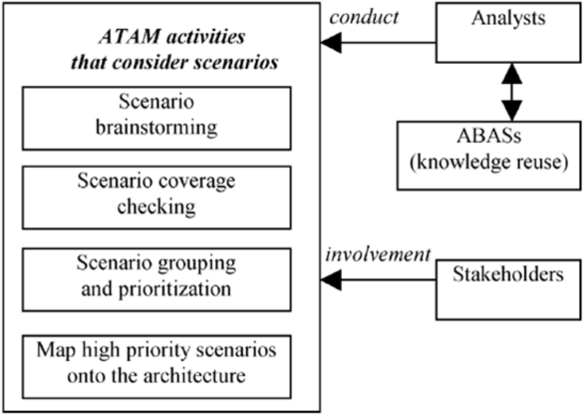

Figure: Scenario-Based Approach

Nine (9) ATAM Steps

Present the ATAM/cloud-ATAM

1. Present the ATAM
2. Present the Business Drivers
3. Present the Architecture
4. Identify the Architectural Approaches
5. Generate the Quality Attribute Utility Tree*
6. Analyse the Architectural Approaches**
7. Brainstorm and Prioritise Scenarios*
8. Analyse the Architectural Approaches**
9. Present the Results

(Steps 5 & 7 [*], and Steps 6 & 8 [**]: *Focus of adaptation*)

ATAM Concept Interactions

Present the ATAM/cloud-ATAM

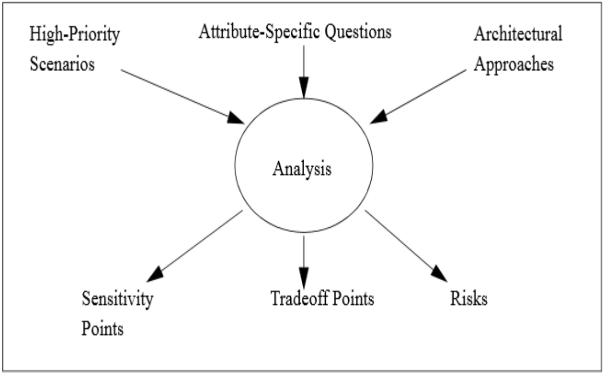

Figure: ATAM Concept Interactions

If ATAM, why *cloud-ATAM*?

Present the ATAM/cloud-ATAM

- Variation of ATAM for *small-to-medium size* (i.e. ISO/IEC 14143:1998 & COSMIC Full FP 2.2) cloud-based systems/architecture. Avoids repetitions in steps that are not required. (**Towards analysis automation**)
- Provision of guidelines for qualitative or quantitative reasoning of architectures using Attribute-Based Architectural Styles (ABAS) (i.e. Continuous Markov Chain Analysis).

cloud-ATAM Overview

Present the ATAM/cloud-ATAM

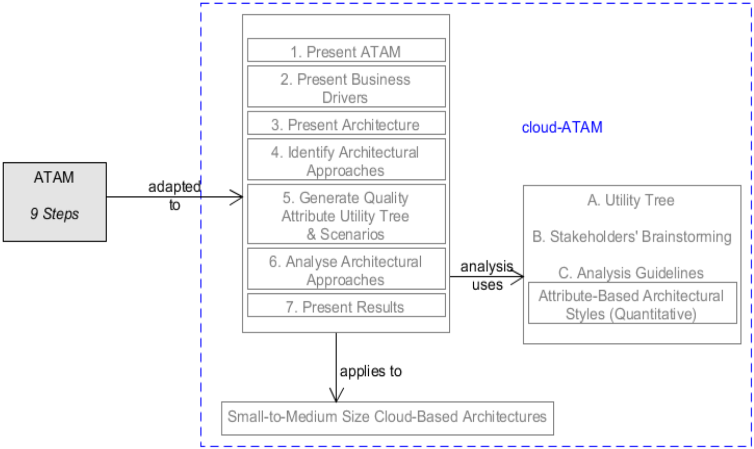

Figure: *cloud-ATAM*: Adapted ATAM with Defined 3-Step Analysis Approach

ATAM Phases*

Present the ATAM/cloud-ATAM

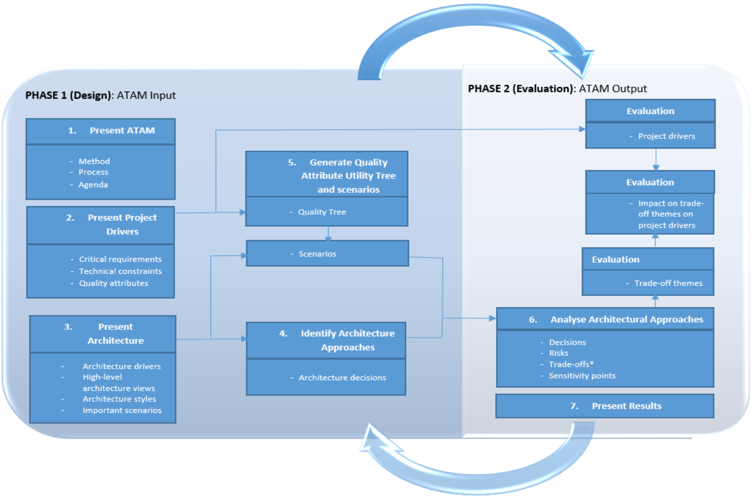

Figure: The cloud-ATAM Phases

ACTIVITY

Questionnaire: Section 1

DYNAMIC VIEW: Reactive Architecture

Present the Business Drivers

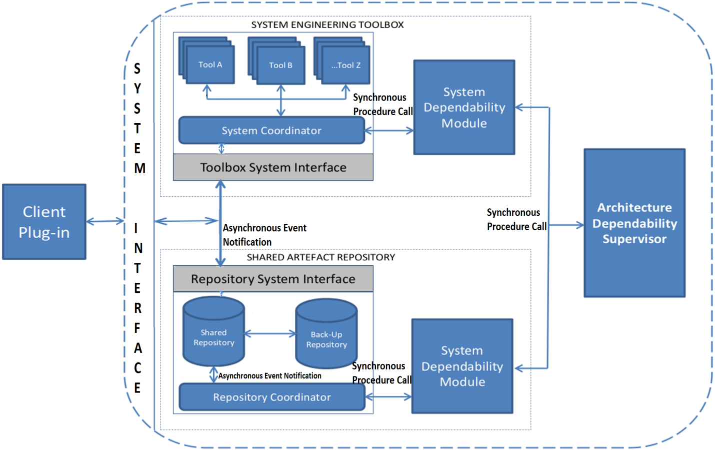

Figure: Dynamic View of Reactive Architecture

Requirements of Reactive Architecture

Present the Business Drivers

1. **The Reactive Architecture must store all artefacts created in all the components of the main composing systems.**
2. **It must monitor and trace all changes to these artefacts to inform system stakeholders.**
3. The *Reactive Architecture* must support multiple users concurrently.
4. The *Reactive Architecture* must provide capacity to scale quickly to accommodate changing demands of system stakeholders, and failures.
5. The *Reactive Middleware* must enable heterogeneous access and analysis operations on artefacts in the *Shared Artefacts Repository*.
6. The *Shared Artefacts Repository* must be backed up asynchronously to facilitate roll-back of repository artefacts.
7. Critical systems such as the *Reactive Middleware* must not constitute a single point of failure which will affect the uptime of the system and the architecture.
8. The *System Engineering Toolbox* must facilitate sequential and parallel execution of tools in a workflow manner.
9. The *Formal Decomposition System* must provided a high capacity and dedicated channel to coordinate real-time analysis on artefacts for local client computers and on remote cloud environment.
10. The *Cloud Accountability System* must gather dependability metrics from several virtual machines, and perform a synchronous analysis of these metrics.
11. Security mechanisms must not degrade defined performance threshold. Specifically, response time for create, delete, update, and display artefact/data operations should not exceed 5 seconds at peak cloud (i.e. architecture) period and less than 1 second during off-peak period.
12. The *Reactive Architecture* must do all this while meeting the *performance* and *availability* requirements to allow it to keep up with the sturdy stream of data and operations on artefacts from the system engineering processes.

Constraints of Reactive Architecture

Present the Business Drivers

The constraints affecting the Reactive Architecture are identified along two perspectives:

1. Reactive Architecture:
   - Cost of development.
   - Time to market, which is dictated by customer demands. (i.e. changing requirements)
   - Skill set of architect.
   - Technical standards.
2. Environment of Architecture (i.e. Cloud Environment):
   - Architecture is highly reliant on the dependability of the cloud infrastructure.
   - Co-location of potentially risky systems on a cloud server/platform.
   - Security of the architecture and data is largely out of the control of the architect.
   - Reliance on COTS products of the cloud service provider (CSP).

Quality Attributes Under Review

Present the Business Drivers

- Availability
- Performance

Reactive Architecture

Present the Architecture

Composed of:

- Reactive Middleware (RM)
- System Artefacts Repository (SAR)
- System Engineering Toolbox (SET)
- Formal Decomposition System (FDS)
- Cloud Accountability System (CAS)

SYSTEM VIEW: Reactive Architecture

Present the Architecture

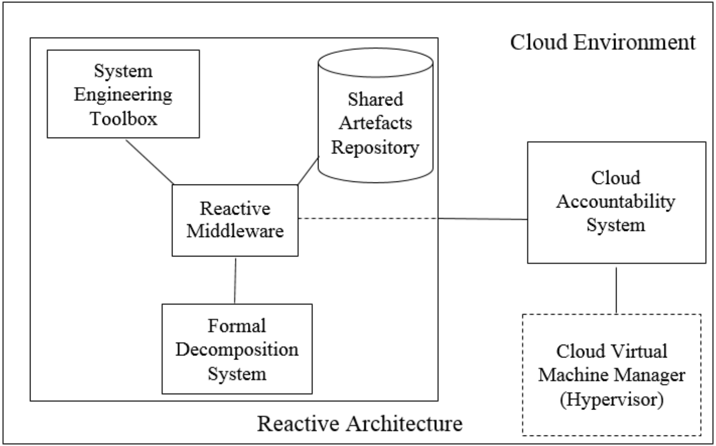

Figure: Overview of Reactive Architecture (**Redundancy**)

SYTEM VIEW: Reactive Architecture

Present the Architecture

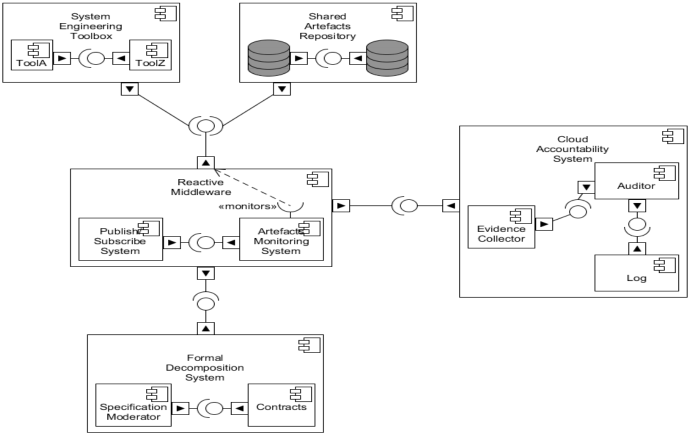

Figure: Component and Connector View of Reactive Architecture

Scenarios of Reactive Architecture

Present the Architecture

Initial....

- *Scenario 1:* User Management
- *Scenario 2:* Adding Tools to the Toolbox
- *Scenario 3:* Saving Artefacts
- *Scenario 4:* Downloading Artefacts
- *Scenario 5:* Sharing Artefacts
- *Scenario 6:* Change Management of Artefacts
- *Scenario 7:* **Traceability of Artefacts**
- *Scenario 8:* Primary Back-Up Repository

SEQUENCE DIAGRAM: Traceability of Artefacts

Present the Architecture

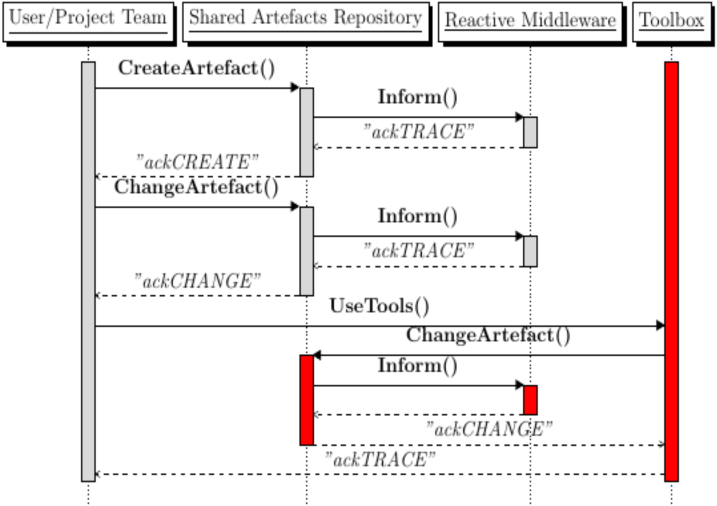

Figure: Traceability of Artefacts (Sequence Diagram)

Requirements of Reactive Architecture

Present the Architecture

Reactive Architecture Quality Attributes of Interest

Quality Attribute Goals	ID	Attribute-Specific Requirements
Operability	O1	The Reactive Architecture must store all artefacts created in all the components of the main composing systems.
	O2*	It must monitor and trace all changes to these artefacts to inform system developers (i.e. clients). (also P1)
	O3	The System Engineering Toolbox must facilitate sequential and parallel execution of tools in a workflow manner.
	O4	The Formal Decomposition System must provided a high capacity and dedicated channel to coordinate real-time analysis on artefacts for local client computers and on remote cloud environment. (also P3)
	O5	The Cloud Accountability System must gather dependability metrics from several virtual machines, and perform a synchronous analysis of these metrics.
Performance	P1*	It must monitor and trace all changes to these artefacts to inform system developers (i.e. clients). (also O2)
	P2*	The Reactive Middleware must enable heterogeneous access and analysis operations on artefacts in the Shared Artefacts Repository. (also A2)
	P3	The Formal Decomposition System must provided a high capacity and dedicated channel to coordinate real-time analysis on artefacts for local client computers and on remote cloud environment. (also O4)
	P4	Security mechanisms must not degrade defined performance threshold. Specifically, response time for create, delete, update, and display artefact/data operations should not exceed 5 seconds at peak cloud (i.e. architecture) period and less than 1 second during off-peak period. (also S1)
	P5	The Reactive Architecture must do all this while meeting the performance and availability requirements to allow it to keep up with the sturdy stream of data and operations on artefacts from the system engineering processes. (also A4)
Scalability	Sc1	The Reactive Architecture must support multiple users concurrently.
	Sc2*	The Reactive Architecture must provide capacity to scale quickly to accommodate changing demands of system developers, and failures. (also A1)
Availability	A1*	The Reactive Architecture must provide capacity to scale quickly to accommodate changing demands of system developers, and failures. (also Sc2)
	A2*	The Reactive Middleware must enable heterogeneous access and analysis operations on artefacts in the Shared Artefacts Repository. (also P2)
	A3	Critical systems such as the Reactive Middleware must not constitute a single point of failure which will affect the uptime of the system and the architecture. (also R1)
	A4	The Reactive Architecture must do all this while meeting the performance and availability requirements to allow it to keep up with the sturdy stream of data and operations on artefacts from the system engineering processes. (also P5)
Maintainability	M1	The Shared Artefacts Repository must be backed up asynchronously to facilitate roll-back of repository artefacts.
Reliability	R1	Critical systems such as the Reactive Middleware must not constitute a single point of failure which will affect the uptime of the system and the architecture. (also A3)
Security	S1	Security mechanisms must not degrade defined performance threshold. Specifically, response time for create, delete, update, and display artefact/data operations should not exceed 5 seconds at peak cloud (i.e. architecture) period and less than 1 second during off-peak period. (also P4)

Figure: Mapping Requirements to Quality Attributes

ACTIVITY

Questionnaire: Section 2

Quality Attribute Characterisation

Identify the Architectural Approaches

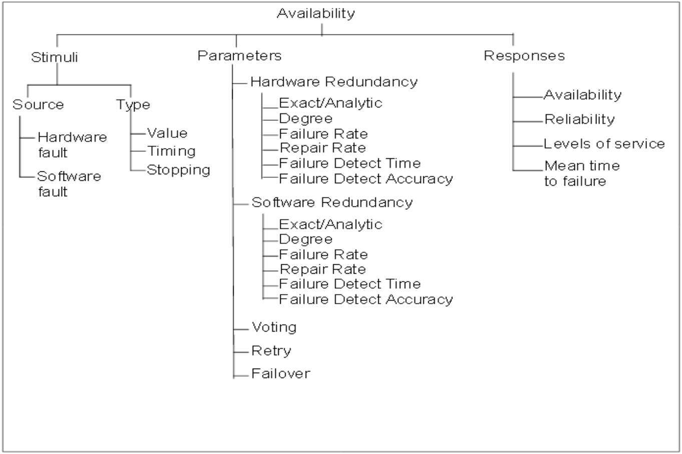

Figure: Availability Characterisation

Quality Attribute Characterisation

Identify the Architectural Approaches

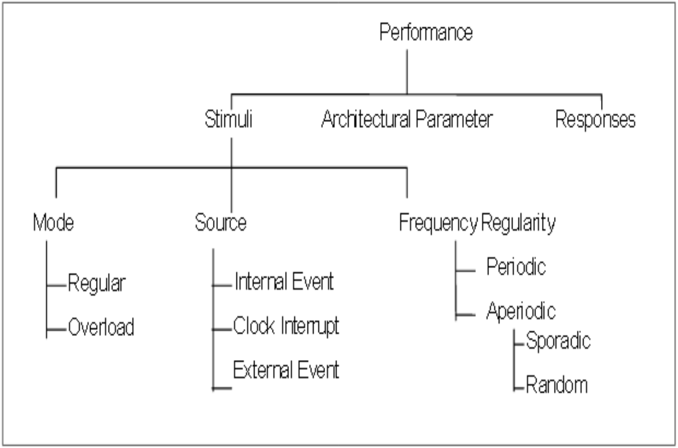

Figure: Performance Characterisation - Stimuli

Quality Attribute Characterisation

Identify the Architectural Approaches

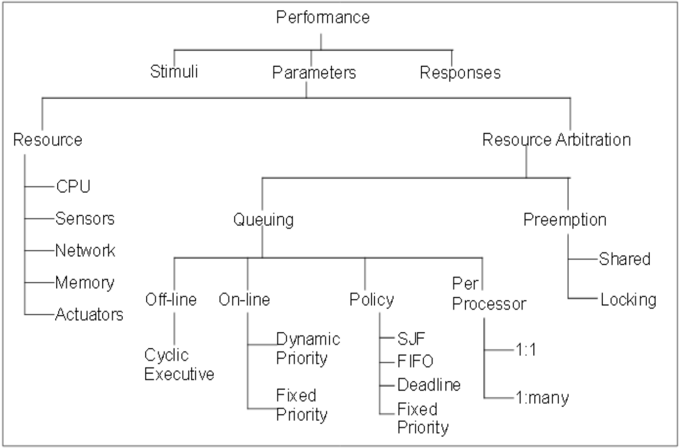

Figure: Performance Characterisation - Parameters

Quality Attribute Characterisation

Identify the Architectural Approaches

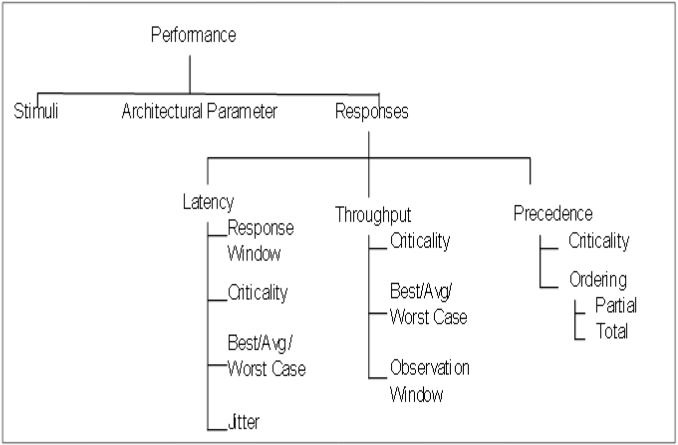

Figure: Performance Characterisation - Response

Attribute-Specific Questions

Identify the Architectural Approaches

Table: 1 Classified Attribute-Specific Questions

Attribute-Specific Questions ID	**Attribute-Specific Questions**
ASQ1	What **facilities** exist in the software architecture (if any) for **self-testing** and **monitoring** of software components? (Availability)
ASQ2	What **facilities** exist in the software architecture (if any) for **redundancy**, **liveness monitoring**, and **fail-over**? (Availability)
ASQ3	How is **data consistency** maintained so that one component can take over from another and be sure that it is in a consistent state with the **failed component**? (Availability)
ASQ4	What is the process and/or task view of the system, including **mapping** of these processes/tasks to **hardware** and **communication mechanisms** between them? (Performance)
ASQ5	What **functional dependencies** exist among the software components? (Performance)
ASQ6	What **data** is kept in the database? How **big** is it, how much does it **change**, who **reads/writes** it? (Performance)
ASQ7	How are **resources allocated** to **service requests**? (Performance)
ASQ8	What are the **anticipated frequency** and **volume of data** transmitted among the system components? (Performance)

Architectural Approaches

Identify the Architectural Approaches

Table: 2 Architectural Approaches for the Reactive Architecture

Architectural Approach ID	Architectural Approaches
AD1	We use the **component-and-connector architectural style** to represent the various components and connections/interfaces of the Reactive Architecture. This is particularly relevant because **it expresses the runtime behaviour of the architecture** under review. Also, **interfaces** are defined as **application programming interfaces** (APIs).
AD2	We AVOID the **distributed data repository** approach in designing the Shared Artefacts Repository. This prevents situations such as issues with **data/database consistency** and possible **modifiability concerns**.
AD3	The **client-server** approach is a best fit for the data-centric **Shared Artefacts Repository** system.
AD4	The **Reactive Middleware** will be adequately represented using the **client-server** approach.
  AD5 **AD6**	Since the **Reactive Middleware** and the **Shared Artefacts Repository** constitute a **single point of failure**, we present the following approaches: - **Back-up** of artefacts in the Shared Artefacts Repository. - **Distributed services** (or modular set of services) for the components of the Reactive Middleware.
AD7	**Schema-free NoSQL data management system** (DMS) is necessary for the Shared Artefacts Repository to minimise or remove **bottlenecks**.
AD8	An **independent communication components** approach (i.e. **high bandwidth**) for communication among the Reactive Middleware, Shared Artefacts Repository, Cloud Accountability System, and the Formal Decomposition System. Such communication approach is particularly relevant for the distributed components of the Cloud Accountability System.

Utility Tree

Generate the Quality Attribute Utility Tree and Scenarios

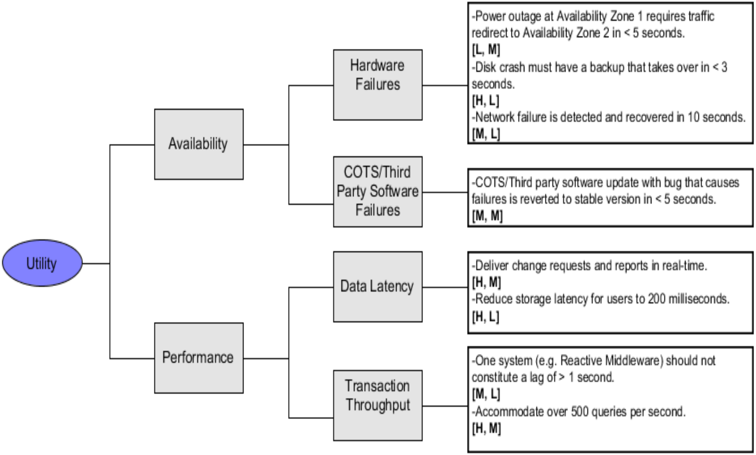

Figure: Attribute Utility Tree of Reactive Architecture

Quality Attribute Scenarios

Generate the Quality Attribute Utility Tree and Scenarios

Table: 3 Prioritised Quality Attribute Scenarios (Ordered)

No.	Quality Attribute Scenarios	Scenario ID	Numbered Value
1	Disk (i.e. data repository) crash must have a back-up that takes over in less than 3 seconds	A2	1
2	**Deliver change requests and reports in real-time**	**P1**	**1**
3	Reduce storage latency for users to 200 milliseconds	P2	**1**
4	Accommodate over 500 queries per second	P4	**1**
5	Network failure is detected and recovered in 10 seconds	A3	**2**
6	COTS/Third party software update with bug that causes failures is reverted to stable version in less than 5 seconds	A4	**2**
7	One system (e.g. Reactive Middleware) should not constitute a lag greater than 1 second	P3	**2**
8	Power outage at *Availability Zone 1** requires traffic redirect to *Availability Zone 2** in less than 5 seconds	A1	**3**

Architecture Components for P1 Analysis

Analyse the Architectural Approaches

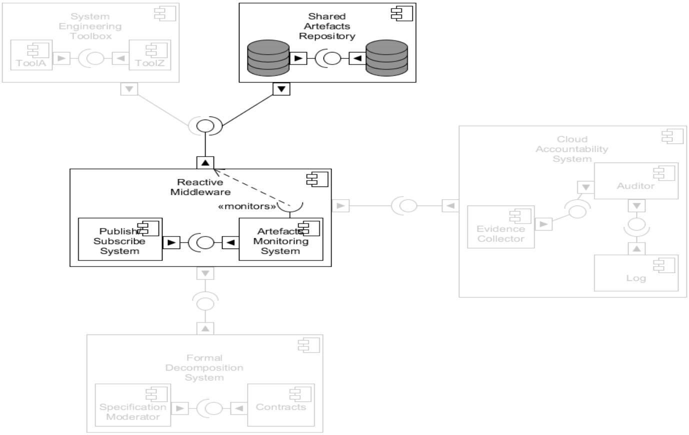

Figure: Component and Connector View of Reactive Architecture

Focal Table for P1 Analysis

Analyse the Architectural Approaches

Table: 4 Analysis of Sensitivities, Trade-offs, Risks & Non-Risks for Scenario **P1**

Sensitivities:	* **S1:** Concern over **network latency**. * **S2:** Using a **data-centric** and client-server approach for the **central repository** can facilitate **data integrity** and **consistency**, BUT it makes the **architecture sensitive** to its **faults** and **bottlenecks**. * **S3:** Similarly, the **central role** played by the Reactive Middleware makes the architecture sensitive to **faults**, **resource** (i.e. CPU, memory) **malfunctions** or **unavailability**.
Trade-offs:	* **T1:** Availability (+) vrs Performance (-) vrs Reliability (-): defining a **central artefacts repository** makes artefacts readily **available**, BUT may be faced with **bottlenecks** when there is a **burst of queries on the repository**. * **T2:** Availability (+) vrs Performance (+): *using APIs for component interfaces facilitate* **readily access** *to resources, and* **boosts performance**. * **T3:** Availability (+) vrs Performance (-): **client-server** approach for the Reactive Middleware allows for multi-client service, BUT there can be an **overwhelming network management performance constraint**. * **T4:** Availability (+) vrs Performance (-): **backing up** the artefacts in the primary Shared Artefacts Repository allows for **fail-over assurance**, BUT the **asynchronous** (i.e. not in real-time) **back-up process** can affect performance.
Risks:	* **R1:** Data integrity. * **R2:** The risk is that the **Reactive Middleware** and the **Shared Artefacts Repository** constitute a **single point of failure**.
Non-Risks:	* **N1:** The non-risk is the use of **application programming interface** (API) approach which should stay compatible. * **N2:** The **independent communication connections** should enable real-time data transfer.

Analysis of Performance Scenario - **P1**

Analyse the Architectural Approaches

Table: 5 Analysis of Performance Scenario - **P1** - (see Table 4 for the description of S1, S2, T1, etc.) and (**C&C + API**: Component-and-connector architectural style and API, **SAR**: Shared Artefacts Repository, **RM**: Reactive Middleware, and **ICC**: Independent Communication Components)

Analysis of Architectural Approach using a Performance-related Scenario				
Scenario ID : *P1*	**Scenario:** *Deliver change requests and reports in real-time*			
Attribute(s)	*Performance (influenced by Availability)*			
Environment	*Normal Operations*			
Stimulus	*Responsiveness to change events*			
Response	*real-time*			
Architectural Decisions	**Sensitivity**	**Trade-off**	**Risk**	**Non-Risk**
AD1 C&C + API	S1			N1
AD2				
AD3 Client-Server SAR	S2	**T1**	R1, R2	N1
AD4 Client-Server RM	S3	**T3**	R2	N1
AD5 Back-up	S1,S2	**T4**		N1
AD6 DS RM	S1		R1	N1
AD7 Schema-free-SAR			R2	
AD8 ICC	S1			N2

Overview of Results

Present the Results

- The *cloud-ATAM* delivers the main **products**: *sensitivities*, *trade-offs*, and *architectural risks* in Table 4.
- From Table 5, the *cloud-ATAM* completed a full cycle by linking the **architectural decisions** to the **quality attributes** (i.e. availability, performance), and back to the **business goals** of the Reactive Architecture.

Summary

- We have motivated the need for architecture evaluation methods suitable for the dynamic unpredictable cloud environments. In particular, we have presented an evaluation method - *cloud-ATAM* - derived from ATAM for evaluating the *availability* and *performance* quality attributes of a cloud-based Reactive Architecture.
- The results from Tables 4 and 5 indicate that the *cloud-ATAM* found some trade-offs (i.e. T1, T3, T4). This answers our **research question**, and validates our **hypothesis** that *the cloud-ATAM is able to identify trade-offs between the availability and performance quality attributes for the Reactive Architecture*.
- **To Do:** Attribute Based Architectural Styles (Continuous Markov Chain Analysis)

ACTIVITY

Questionnaire: Section 3