QRS

{insert client logo here}

Client – MVP

ATAM Architectural Approach Analysis Document

Version 1.0

CONTENTS

1 Document Administration

1.1 Document Purpose

This step sizes up how well suited steps 4 & 5 are to each other. Here the architectural decisions are done and identifying their risks, non-risks, sensitivity points and trade-offs.

Each architectural approach or decision is relevant to each high-priority utility tree scenario. These were identified in step 4. If not, why the discrepancy? This section should identify the approach and the components, connectors and constraints involved.

1.2 Project Reference

Project Sponsor	
Project Manager	
Product Owner	
Operational Lead	
Summary of Impacted Business Groups / Sites	
Impacted Current-State Information Systems / Applications	

1.3 Revision History

Version	Date	Author	Revisions
1	11/4/2015	Robin G Coles	

1.4 Document Contributors

The following people made significant contributions to the content of this document:

Contributor	Contribution
QRS Team	*Authoring and overall document management*

1.5 Related Documents

Name	ATAM Quality Attribute Library
Description	
Path or Source	
Name	ATAM Utility Tree
Description	
Path or Source	

2 Acronyms & Definitions

The following critical abbreviations and acronyms appear throughout the remaining document sections.

Abbreviation or Term	Definition
Architectural Decision	One of three categories in the quality attribute characterization. An architectural decision is the aspect of an architecture – components, relations, and their properties – that have a direct impact on achieving attribute responses.
Availability	Availability is the proportion of time the system is up and running. It is measured by the length of time between failures as well as how quickly the system is able to resume operation in the event of failure.
Business Goals	Goals that motivate the development effort and hence the primary architectural drivers presented. These are high reliability and high performance.
Compatibility	The ability to work with other systems
Conceptual integrity	Conceptual integrity is the underlying theme or vision that unifies the design of the system at all levels. The architecture should do similar things in similar ways. Conceptual integrity is exemplified in an architecture that exhibits consistency, has a small number of data and control mechanisms, and uses a small number of patterns throughout to get the job done.
Cost and Schedule	The cost of the system with respect to time to market, expected project lifetime, and utilization of legacy and COTS systems.
Efficiency	Efficiency is the ability of the software to do the required processing on least amount of hardware.
Environment	Describes what's going on at time of stimulus. Asks the question, what is the system's state, unusual conditions in effect, system heavy load, is processer down, are communication channels flooded? Any ambient condition relevant to understanding scenario if environment is "under normal conditions" then it is omitted.
Flexibility	The ease with which a system or component can be modified for use in applications or environments, other than those for which it was specifically designed. [Barbacci, 1995]
Interoperability	Interoperability means the ability to work on any component or application with other components or application without making any special effort. The ability of a collection of communicating entities to share specific information and operate on it according to an agreed-upon operational semantics. It is determined by factors such as: • Number of different platforms (Microsoft .Net, Java, Perl, PHP) • Complexity of platforms (due to non-availability of universal standard for interoperability) • Coupling between platforms (WSDL), SOAP, Web services standards (e.g., BPEL, ebXML, WS-Security). Web services-interoperability organization was chartered in 2002, to promote the interoperability of Web services across platforms, applications, and programming languages. Still more research work with respect to Data Access, Encoding Style and Conversion Format is desired to achieve smooth interoperability between systems. High interoperability → multiple platforms, interface format, communication protocols

Abbreviation or Term	Definition
Maintainability	Maintainability is the ability of the system to undergo changes with a degree of ease. These changes could impact components, services, features, and interfaces when adding or changing the functionality, fixing errors, and meeting new business requirements.
Manageability	Manageability defines how easy it is for system administrators to manage the application, usually through sufficient and useful instrumentation exposed for use in monitoring systems and for debugging and performance tuning.

Modifiability	Modifiability is the ability to make changes to a system quickly and cost effectively. It is measured by using specific changes as benchmarks and recording how expensive those changes are to make.
	Two factors that affect modifiability are:
	• Extensibility
	  ○ How many new services have been added or modified with/without new interfaces?
	  ○ How many old services deleted/discounted?
	• Changeability
	  ○ How many changes have been added to existing services?
	SOA is loose-coupled in nature, which results in reduction of cost of modifiability. In SOA, service capability can be extended without affecting other parts of the system. Extension in SOA includes addition of new services according to web standards and extension of existing services with/without changing the interfaces. A great research is required to classify the processes and techniques which deal with identifying the impact of services and incorporating new versions of service within the existing SOA environment.

Non-risk	Non-risks are good decisions that rely on assumptions that are frequently implicit in the architecture. A non-risk remains a non-risk when the assumption has not changed (or at least if it changes, the designation of non-risk will need to be re-justified). Documenting of non-risks consists of: • An architectural decision (or a decision that has not been made) • A specific quality attribute response that is being addressed by that decision along with the consequences of the predicted level of the response A rationale for the positive or negative effect that decision has on meeting the quality attribute requirement.

Abbreviation or Term	Definition
Performance	Performance refers to the responsiveness of the system – the time required to respond to stimuli (events) or the number of events processed in some interval of time. Performance qualities are often expressed by the number of transactions per unit time or by the amount of time it takes to complete a transaction with the system. Performance measures are often cited using benchmarks, which are specific transaction sets or work-load conditions under which the performance is measured. Performance of a SOA can be measured with respect to factors like: • Response time (how long does it take to process a request?) • Throughput (how many requests can be processed pre unit of time?) • Timeliness (desired time to process a request)
Portability	Portability is the ability of the system to run under different computing environments. These environments can be hardware, software, or a combination of the two. A system is portable to the extent that all of the assumptions about any particular computing environment are confined to one component (or at worst, a small number of easily changed components). If porting to a new system requires change, then portability is simply a special kind of modifiability.
Reliability	Reliability is the ability of the system to keep operating over time. Reliability is usually measured by mean time to failure.
Response	Tells how the system, through its architecture should respond to the stimulus. For example: is the function carried out, tested successful, reconfiguration happen, how much effort did the maintenance change require? It's often the key to understanding what quality attribute the stakeholder who proposed the scenario is concerned about. End-to-end, worst-case latency. Requirements expressed in terms that are concrete and measurable or observable.
Reusability	Reusability defines the capability for components and subsystems to be suitable for use in other applications and in other scenarios. Reusability minimizes the duplication of components and also the implementation time.
Risk	Risks are potentially problematic architectural decisions. It should be understood and explicitly recorded. Documenting of risks consists of: • An architectural decision (or a decision that has not been made) • A specific quality attribute response that is being addressed by that decision along with the consequences of the predicted level of the response • A rationale for the positive or negative effect that decision has on meeting the quality attribute requirement.

Abbreviation or Term	Definition
Scalability	The ability of SOA to work well when the change in size or in volume occur in the system to meet user's need is known as scalability. Web service technology do not consist any inherent scalability features. Platform vendors provide the mechanisms like: • Horizontal scaling (addition in load-balanced servers) • Vertical scaling (increment in the capacity of the server) • Stateless services (avoid session management) • Service scope (creation of an instance of a service) Scalability can be determined by sources like type of transport protocol, the XML parser, the load-balancing algorithm and SOAP runtime. The performance of the system depends upon the magnitude of the scalability like how many service users like 10, 100, 1000, or 10,000 are handled by the system. The strategies which can affect the quality of system are needed to develop for making system more scalable.
Scenario	A scenario is a short statement describing an interaction of one of the stakeholders with the system. They resemble use cases. Each scenario is associated with: • A particular stakeholder • Addresses a particular quality, in specific terms. For example: • A maintenance stakeholder would describe making a change to the system • A developer might involve using the architecture to build the system or predict its performance • A customer might describe the architecture reused for a second product in a product line or might assert that the system is buildable given certain resources.
Scenario Structure	Scenarios tell a brief story about an interaction with the system from the point of view of a stakeholder. This has three parts: • Stimulus • Environment • Response There are also three types of scenarios: • Use case • Growth • Exploratory

Abbreviation or Term	Definition
Security	Security is a measure of the system's ability to resist unauthorized attempts at usage and denial of service while still providing its services to legitimate users. Security is categorized in terms of the types of threats that might be made to the system. Security of SOA and web services is linked with four factors: • Confidentiality • Authenticity • Integrity • Availability
Sensitivity	An architectural decision involving one or more architectural components that is critical for achieving a particular quality attribute response measure. This serves as a "Yellow Flag".
Stimulus	Explains or describes what the stakeholder does to initiate the interaction with the system. A user may invoke a function; maintainer makes a charge; tester runs tests; operator reconfigures the system. A characterization of the stimuli that this ABAS is designed to respond to and a description of the quality-attribute-specific measures of the response. Stimuli is two or more periodic or sporadic input streams of arrivals.
Subsetability	Sub-set-ability is the ability to support the production of a subset of the system. This is the most useful and most overlooked property. Subsetability can spell the difference between being able to deliver nothing when schedules slip versus being able to deliver a substantial part of the product. Subsetability also enables incremental development, a powerful development paradigm in which a minimal system is made to run early on and functions are added to it over time until the whole system is ready. Subsetability is a special kind of variability, mentioned above.
Supportability	Supportability is the ability of the system to provide information helpful for identifying and resolving issues when it fails to work correctly.
Sustainability	Sustainability refers to the cost efficient maintenance and evolution of a long-living software system over the entire life-cycle. The quality of software architectures determines sustainability to a large extent.
Tradeoff	An architectural decision that effects more than one attribute and is a sensitivity point for more than one attribute. If you can answer a question with "We haven't made that decision yet" then you cannot point to a component or property in the architecture and call it out as a sensitivity point because the component or property might not exist yet.
UI	User Interface

Abbreviation or Term	Definition
Usability	When a service interacts with the user, then its level of quality determines usability. It depends upon • Availability of services • Complexity of Interface • Coupling of services • Reliability of services Due to distributed nature of SOA, user actions involve calls to remote service providers. If the service call takes a long time to response then it is good to move the service communication on the separate thread which contains service-users. But in the case of web-services solutions, there is no option to give the user-effective feedback or control over the communication. The SOAP protocol does not have the option for progress notification or cancellation of an active call. These two options are required when a lengthy process is requested. There is still a need to find the mechanisms which provide user's effective feedback or control over communications.

3 Analyze Architectural Approaches

Once the scenarios are collected, the next step is to analyze their relevancy to the architectural decision; for the service integrator. Since each scenario in the utility tree is ranked "high", they are all listed in the table below; one at a time.

In this step, each scenario includes a business goal and quality attribute. Plus a refinement broken down into three parts as follows:

- Stimulus
- Environment
- Response.

During this process, the table lists at least one architectural decision to support its scenario. It also includes any risks, non-risks, sensitivity points, and tradeoffs that may occur.

4 Scenario

4.1 System experiences system upgrades/maintenance issues.

Scenario	System experiences system upgrades/maintenance issues.	
Business Goal(s)	Business wants to make the application highly available and minimize downtime.	
Attribute	Availability	
Scenario Refinement	Stimulus	System upgrade/maintenance issues
	Environment	Upgrade/Maintenance schedule
	Response	System is fixed and updated based on a specific standard.
Architectural Decisions/architectural Reasoning	Create a clustered environment. So that there is a fail-over cluster and the next cluster can take over.	
Risks	Low/None	
Sensitivities	Low	
Tradeoffs	None	
Non-risks		
Other Issues		
Reference	Mapped to WDFM clustering (5.2, 5.5, 5.8)	

4.2 System cannot support incompatible features.

Scenario	System cannot support incompatible features.	
Business Goal(s)	Business wants to mainstream features that support and enhance customer satisfaction.	
Attribute	Compatibility	
Scenario Refinement	Stimulus	System cannot support unknown features.
	Environment	Normal working environment
	Response	Communicate with users regarding compatibility issues.
Architectural Decisions and Reasoning	Apply more modular design techniques for an architecture.	
Risks	High	
Sensitivities	High	
Tradeoffs	High	
Non-risks		
Other Issues		

4.3 System has gone through multiple iterations w/diverging architecture

Scenario	System has gone through multiple iterations with a diverging architecture over time.	
Business Goal(s)	Business wants to improve the application at appropriate intervals.	
Attribute	Conceptual Integrity	
Scenario Refinement	Stimulus	System has conflicting architecture.
	Environment	System evaluation/assessment
	Response	Business plans to overhaul and uniformize the architecture.
Architectural Decisions and Reasoning	Evaluate risks and trade-offs between conflicting architectures. Setting expectations with stakeholders.	
Risks	Low	
Sensitivities	Low	
Tradeoffs	Low	
Non-risks		
Other Issues		

4.4 System is handled by less optimal Professional Services

Scenario	System is handled by less optimal Professional Services engagement (SLA) and affects both product development and delivery schedule.	
Business Goal(s)	Business wants to provide a highly updated, optimized and intuitive system to the users.	
Attribute	Cost and Schedule	
Scenario Refinement	Stimulus	Less than optimal SLA impacts product development and throughput.
	Environment	Product development.
	Response	Set expectations with upper level management and stakeholders. Re-negotiation of SLA's for product development.
Architectural Decisions and Reasoning	Plan for a well-documented system Execute documentation, system back-ups, and institute system audits Search for better product development options	
Risks	Low	
Sensitivities	Low	
Tradeoffs	Low	
Non-risks		
Other Issues		

4.5 System experiences extreme time lapse

Scenario	System experiences extreme time lapse due to user overload.	
Business Goal(s)	Business wants to provide a high performance application at all times.	
Attribute	Efficiency	
Scenario Refinement	Stimulus	System experiences user overload
	Environment	Peak time (hours) working conditions
	Response	System capacity is augmented to accommodate peak time volume of users.
Architectural Decisions and Reasoning	Employ an elastic, scalable, infrastructure architecture.	
Risks	Low	
Sensitivities	High	
Tradeoffs	Low	
Non-risks		
Other Issues		

4.6 Single or current vendor goes out of business.

Scenario	Single or current vendor goes out of business or is failing to meet minimum requirements.	
Business Goal(s)	Business needs to explore additional vendors.	
Attribute	Flexibility	
Scenario Refinement	Stimulus	Business has to find another vendor to meet their needs
	Environment	Normal working environment
	Response	Business concentrates on researching new vendors Business plans to branch out with another, more mainstream application
Architectural Decisions and Reasoning	Architect for a highly compatible, portable application.	
Risks	Low	
Sensitivities	Low	
Tradeoffs	Low	
Non-risks		
Other Issues		

4.7 System experiences interoperability protocol changes

Scenario	System experiences interoperability protocol changes due to partial system upgrades.	
Business Goal(s)	Business wants to be able to exchange data with third party interfaces, as needed.	
Attribute	Interoperability	
Scenario Refinement	Stimulus	Interoperability protocol changes occur, prevents data exchange, and communication with other systems.
	Environment	System upgrade schedule
	Response	Interoperability protocols are re-aligned and tested.
Architectural Decisions and Reasoning	Create adapter to program and/or implement abstraction layer protocol interfaces to create abstraction layer for dynamic translation. Set-up test environment to conduct testing All upgrades and impacts must be tested Conduct regression testing To avoid such situations	
Risks	High/none Low/none Medium	
Sensitivities	Low	
Tradeoffs	none none low	
Non-risks		
Other Issues		
Reference	HL7 Protocol	

4.8 Security flaw is identified in the database software

Scenario	A security flaw is identified in the database software causing all services accessing the database to be unable to connect after the security patch is applied.	
Business Goal(s)	Business wants to be able to exchange data with third party interfaces, as needed.	
Attribute	Interoperability	
Scenario Refinement	Stimulus	Security patch is applied
	Environment	Security flaw in system
	Response	Correct connection issue with database access component
Architectural Decisions and Reasoning	Create single access point into database that is used by all other services, reducing the changes required to only one component	
Risks	Low	
Sensitivities	High	
Tradeoffs	High	
Non-risks		
Other Issues		

4.9 Unavailability of source code for the purpose of modifiability.

Scenario	Unavailability of source code for the purpose of modifiability.	
Business Goal(s)	Business wants to preserve all source codes and other information needed to allow their system to remain modifiable.	
Attribute	Manageability	
Scenario Refinement	Stimulus	Unavailability of source code
	Environment	Assessment mode of plans for modifications
	Response	Implementation of more effective source code preservation methods Create an inventory of applications without a source code Communicate to stakeholders regarding applications with unavailable source codes Explore tools and software to reverse-engineer source code from binary code
Architectural Decisions and Reasoning	Re-engineer parts of the application with unavailable source code. To have some kind of back-up.	
Risks	Medium	
Sensitivities	High	
Tradeoffs	High	
Non-risks		
Other Issues		

4.10 Client acquires additional companies

Scenario	Client acquires additional companies who already have multiple systems.	
Business Goal(s)	Business needs to integrate all the systems together.	
Attribute	Modifiability	
Scenario Refinement	Stimulus	Acquisition outside Client's entity
	Environment	Assessment mode of plans for modifications
	Response	Implementation of more effective acquisition methods Create an inventory of applications that don't work together Communicate to stakeholders regarding applications outside Client's entity Explore tools and software to reverse-engineer applications
Architectural Decisions and Reasoning	Create an architecture that allows additional systems to be added to allow inter-application data transfer	
Risks	High	
Sensitivities	High	
Tradeoffs	High	
Non-risks		
Other Issues		

4.11 System experiences limit or stretch of scalability

Scenario	System experiences limit or stretch of scalability extent due to volume of users.	
Business Goal(s)	Business wants to provide a high performance application at all times	
Attribute	Performance	
Scenario Refinement	Stimulus	Scalability limit is stretched due to high volume of users
	Environment	Normal working environment
	Response	System should apply the graceful degradation plan
Architectural Decisions and Reasoning	Plan on highly scalable elastic architecture that auto detects loads, spins up additional resources, and/or expands accordingly.	
Risks	Medium	
Sensitivities	Medium	
Tradeoffs	None	
Non-risks		
Other Issues		

4.12 Integration service provider fails

Scenario	Integration service provider fails and Client moves to a different vendor.	
Business Goal(s)	Business wants to provide a highly compatible application to users at all times.	
Attribute	Portability	
Scenario Refinement	Stimulus	Application becomes incompatible with other systems/devices, etc.
	Environment	System upgrades need to be scheduled
	Response	Communicate issues with user and setting expectations with users regarding application availability. Execute a roll-back
Architectural Decisions and Reasoning	Create a statement of procedures around major updates.	
Risks	High	
Sensitivities	High	
Tradeoffs	High	
Non-risks		
Other Issues		

4.13 System experiences major updates

Scenario	System experiences major downtime.	
Business Goal(s)	Business wants to provide a reliable system without any gaps or outages in service	
Attribute	Reliability	
Scenario Refinement	Stimulus	System experiences major downtime
	Environment	Normal working environment
	Response	Communication to all users about major issues Execute a roll-back plan or expedited update
Architectural Decisions and Reasoning	Create statement of procedures around major downtime.	
Risks	High	
Sensitivities	High	
Tradeoffs	High	
Non-risks		
Other Issues		

4.14 System does not have self-documenting capability

Scenario	System does not have self-documenting capability.	
Business Goal(s)	Business wants to leverage all resources of/from the system as much as possible	
Attribute	Reusability	
Scenario Refinement	Stimulus	System does not have self-documenting capability
	Environment	Normal working environment
	Response	Business plans to enhance system with documenting capabilities
Architectural Decisions and Reasoning	Explore development studios, add-ins, and third party applications which aid self-documenting source code.	
Risks	Low	
Sensitivities	Low	
Tradeoffs	Low	
Non-risks		
Other Issues		

4.15 Transaction latency increases

Scenario	Transaction latency increases due to increase in volume between both users and transactions.	
Business Goal(s)	Business wants to provide a high performance application at all times.	
Attribute	Scalability	
Scenario Refinement	Stimulus	Scalability limit is stretched due to high volume of users.
	Environment	Normal working environment
	Response	System automatically scales to maintenance acceptable average latency times.
Architectural Decisions and Reasoning	Plan to on-board additional capacity when scalability is stretched. To architect an elastic, scalable environment.	
Risks	Medium	
Sensitivities	Medium	
Tradeoffs	None	
Non-risks		
Other Issues		

4.16 Incorrect vendor or design pattern prevents effective load

Scenario	Incorrect vendor or design pattern prevents effective load balancing for key architectural areas	
Business Goal(s)	Business wants to provide a high performance application at all times.	
Attribute	Scalability	
Scenario Refinement	Stimulus	Poor design-time decision for architecture
	Environment	Normal working time
	Response	Re-architect system
Architectural Decisions and Reasoning	Create stateless services layer with the ability to shift load to alternative resources	
Risks	Low	
Sensitivities	High	
Tradeoffs	Medium	
Non-risks		
Other Issues		

4.17 System experiences security breach

Scenario	System experiences security breach due to a flaw in the system configuration.	
Business Goal(s)	Business wants to have a highly secure application to protect the application against unauthorized activity.	
Attribute	Security	
Scenario Refinement	Stimulus	Security flaw is exposed and leveraged for unauthorized usage due to configurations saved in plain text with passwords
	Environment	Compromised application
	Response	Security flaw is addressed. Configurations and passwords are encrypted. Overall security is re-examined and upgraded.
Architectural Decisions and Reasoning	Business considers a configuration vault. To restore configurations.	
Risks	Low	
Sensitivities	High	
Tradeoffs	Medium	
Non-risks		
Other Issues		

4.18 Ability to operate part of the system as its own system

Scenario	Ability to operate part of the system as its own complete sustainable system.	
Business Goal(s)	The ability to separate systems and have them operate independently	
Attribute	Subsetability	
Scenario Refinement	Stimulus	Change in enterprise IT strategy
	Environment	Normal
	Response	Separate out the undesired systems
Architectural Decisions and Reasoning	Create a publish and subscriber-based architecture thus allowing systems to provide and consume data from whatever systems exist within its own subsystem	
Risks	Low	
Sensitivities	High	
Tradeoffs	High	
Non-risks		
Other Issues		

4.19 System component experiences catastrophic failure

Scenario	System component experiences catastrophic failure which impacts exemption handling and logging activity.	
Business Goal(s)	Business wants to have a self-diagnosing/self-monitoring application that will provide guidance for exception handling, etc. through its comprehensive activity logging capability.	
Attribute	Supportability	
Scenario Refinement	Stimulus	System's exception handling and activity logging capability is compromised due to catastrophic failure.
	Environment	Catastrophic failure of system
	Response	System operates redundant mode Auto startup of additional backup system Notification to system administrator when an error occurs
Architectural Decisions and Reasoning	To architect application monitoring system that runs outside of the system.	
Risks	Low	
Sensitivities	Low	
Tradeoffs	Low	
Non-risks		
Other Issues		

4.20 System maintenance becomes expensive

Scenario	System maintenance becomes expensive due to human capital investment.	
Business Goal(s)	Business wants to maximize human capital investment while still minimizing system maintenance cost.	
Attribute	Sustainability	
Scenario Refinement	Stimulus	Expensive human labor for system maintenance
	Environment	Normal working conditions
	Response	Proper vetting of costs for all stages of the application in the future Negotiate with third party to have staff on demand
Architectural Decisions and Reasoning	Explore modularity in architectural design Employ automated tests and methodologies	
Risks	High	
Sensitivities	High	
Tradeoffs	Medium	
Non-risks		
Other Issues		

4.21 Management desires to minimize down time

Scenario	Management desires to minimize down time during normal business hours	
Business Goal(s)	Business wants to ensure user is getting maximum usability out of the application	
Attribute	Usability	
Scenario Refinement	Stimulus	Platform requires updates and maintenance
	Environment	Normal working conditions
	Response	System maintenance should not affect usability of system.
Architectural Decisions and Reasoning	Modular multi-server environment	
Risks	High	
Sensitivities	High	
Tradeoffs	High	
Non-risks		
Other Issues		

5 References

[Barbacci 1995] https://msdn.microsoft.com/en-us/library/bb402962.aspx

[Clements, Kazman, Klein 2002] Book: Evaluating Software Architectures: Methods and Case Studies

[QRS 2015] ATAM Quality Attribute Library document

[Microsoft Corporatioin, 2009] Microsoft Application Architecture Guide, 2nd Edition
https://msdn.microsoft.com/en-us/enus/library/ee658094.aspx

[Universidad Politecnica de Valencia 2013] Architecture Evaluation Methods: Introduction to ATAM
http://users.dsic.upv.es/~jagonzalez/IST/files/IntroductionATAM.pdf

http://www.softwarearchitectures.com/qa.html