Trade-off Analysis

Software Architecture
3rd Year – Semester 1
Lecture 15

What is a Trade-off?

- A trade-off is a situation that involves losing one quality or aspect of something in return for gaining another quality or aspect
- How much must I give up to get a little more of what I want most?

What is ATAM

- ATAM (Architecture Trade-Off Analysis Method) is an architecture evaluation method developed in the Software Engineering Institute at the end of the 90s
- The objective of Architecture Trade-off Analysis (ATAM) is to provide a justifiable way to understand a Software Architecture's fitness with respect to multiple competing Quality Attributes

Why ATAM?

- The purpose of ATAM is to assess the consequences of architectural design decisions in the light of quality attributes
- ATAM helps in foreseeing how an attribute of interest can be affected by an architectural design decision
- The Quality Attributes of interest are clarified by analyzing the stakeholder's scenarios in terms of stimuli and responses
- Once the scenarios had been defined, the next step is to define which architectural approaches can affect to those quality attributes

Quality Attributes and ATAM

- Quality Attributes can interact or conflict – improving one often comes at the price of worsening one or more of the others, thus it is necessary to trade off among multiple Software Quality Attributes at the time the Software Architecture of a system is specified, and before the system is developed
- ATAM helps above process

Stakeholder Expectations

- Performance
- Availability
- Usability
- Security

End User's view

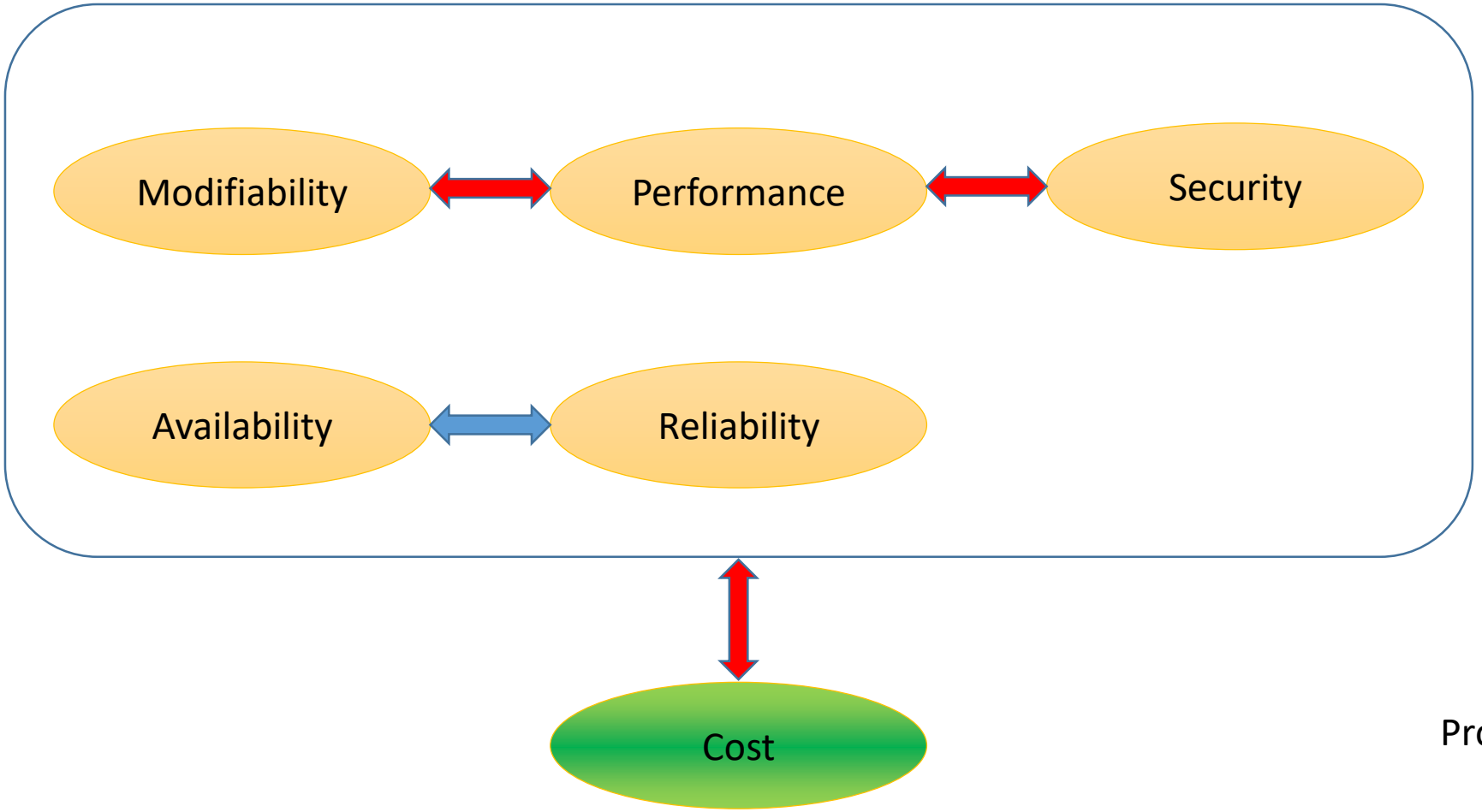
- Time To Market
- Cost and Benefits
- Projected life time
- Targeted Market
- Integration with Legacy System
- Roll back Schedule

Business Community view

- Maintainability
- Portability
- Reusability
- Testability

Developer's view

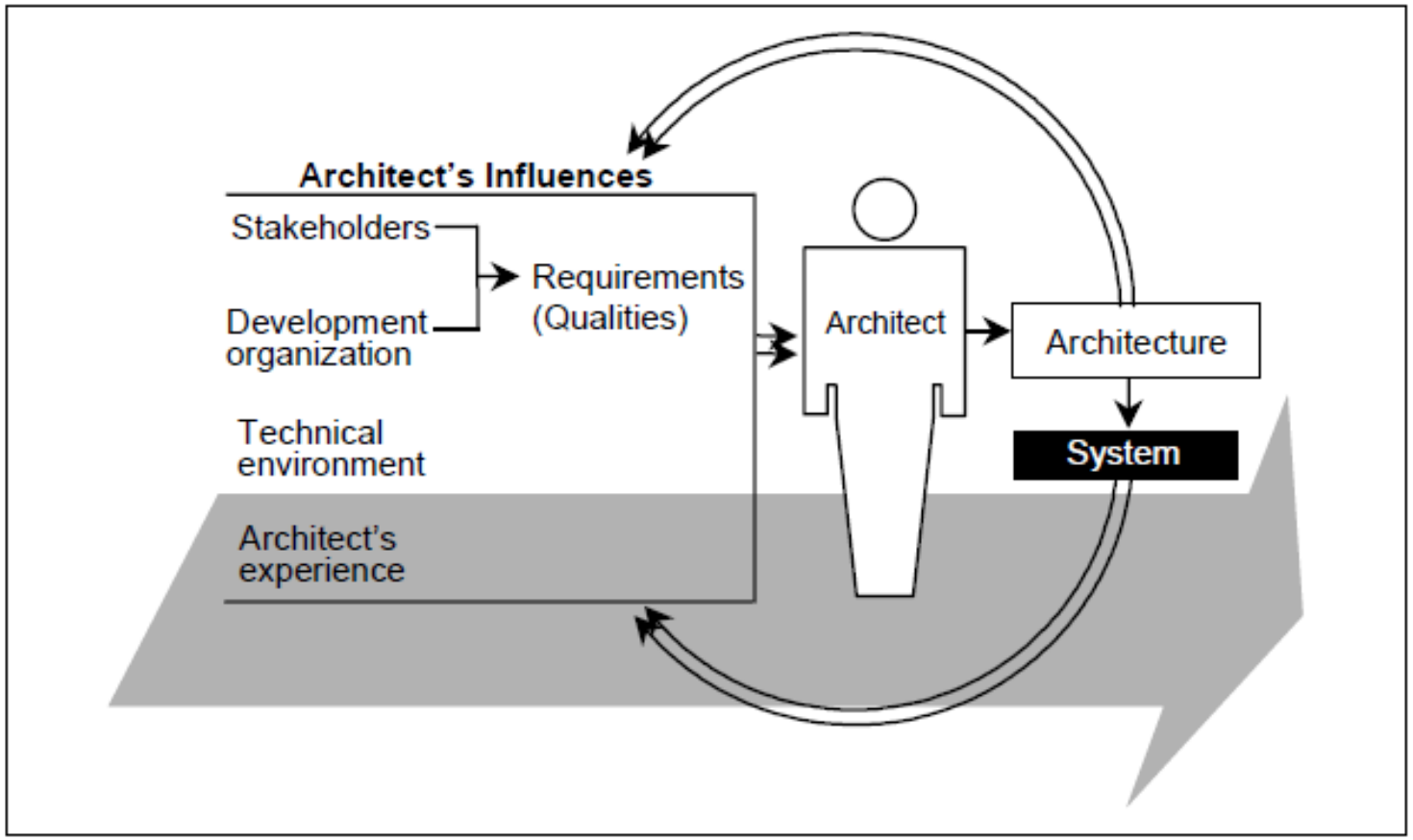
Attribute Impact



Project Management Triangle

Architect's Role & Influence

- Envision
- Realize
- Influence



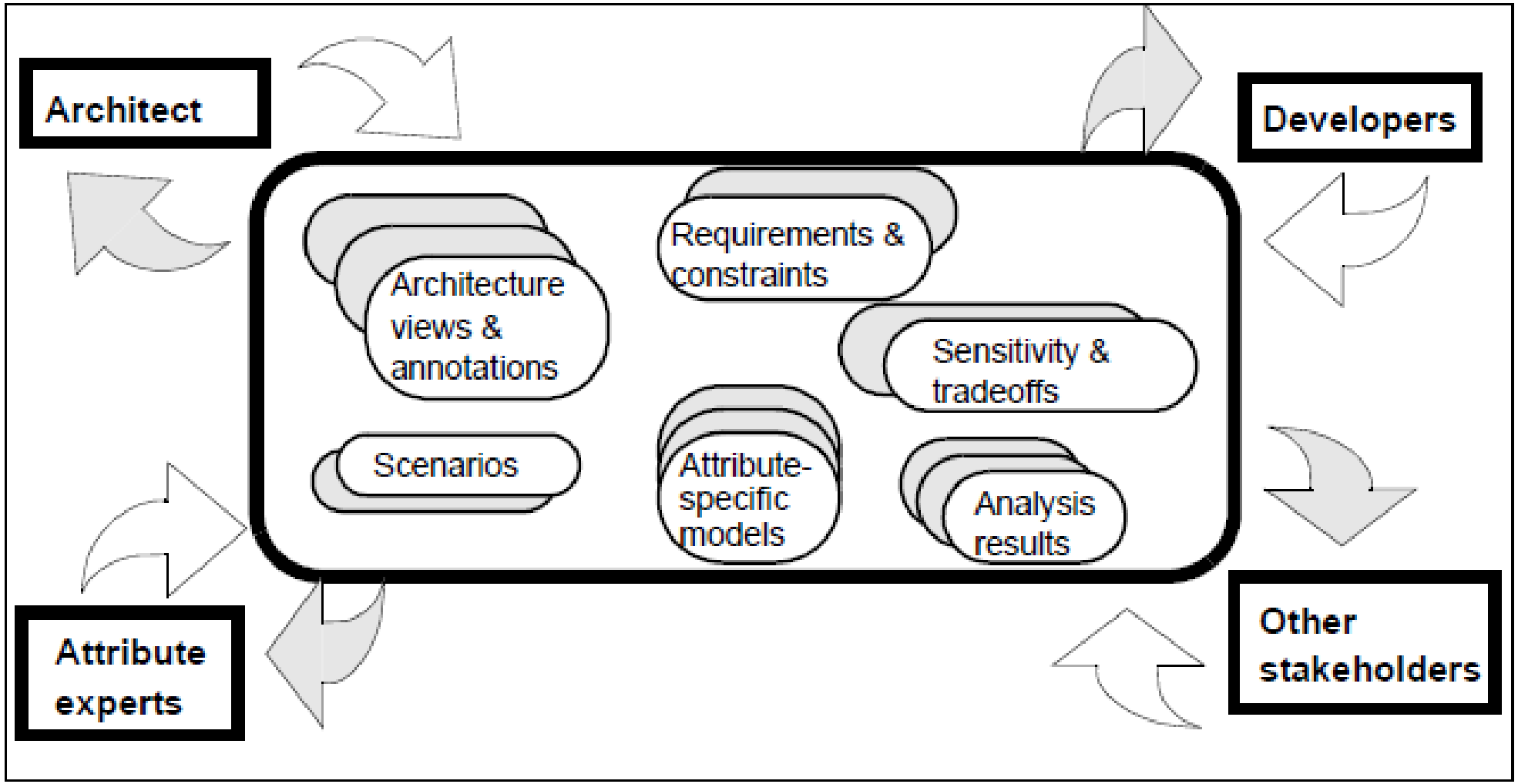
The Architecture Business Cycle

- Software architecture is a result of Technical, Business, and Social influences. Its existence in turn affects the technical, business, and social environments that subsequently influence future architectures. We call this cycle of influences, from the environment to the architecture and back to the environment, the Architecture Business Cycle
  - How organizational goals influence requirements and development strategy.
  - How requirements lead to an architecture.
  - How architectures are analyzed.
  - How architectures yield systems that suggest new organizational capabilities and requirements

Architect's Roles & Responsibilities

- Abstracts the complexity of a system into a manageable model that describes the essence of a system by exposing important details and significant constraints
- Sets quantifiable objectives that encapsulate quality attributes of a system
- Maintains control over the architecture lifecycle parallel to the project's software development lifecycle
- Stays on course in line with the long term vision
- Makes critical decisions that define a specific direction for a system in terms of implementation, operations, and maintenance
- Creates and distributes tailored views of software architectures to appropriate stakeholders at appropriate intervals
- Works closely with executives to explain the benefits and justify the investment in software architecture of a solution
- Acts as an agent of change in organizations where process maturity is not sufficient for creating and maintaining architecture centric development
- Inspires, mentors, and encourages colleagues to apply intelligently customized industry's best practices

Supporting roles for ATAM



Challenges

Most complex software systems are required to be modifiable and have good performance. They may also need to be secure, interoperable, portable, and reliable.

For any particular system:

- What precisely do these quality attributes such as modifiability, security, performance, and reliability mean?
- Can a system be analyzed to determine these desired qualities?
- How soon can such an analysis occur?
- How do you know if a software architecture for a system is suitable without having to build the system first?

Flow of ATAM


Business Drivers
Quality Attributes
Scenarios
Analysis
Architectural Plan
Architectural Approaches
Architectural Decisions
Tradeoffs
Impacts
Sensitivity Points
Non-Risks
Distilled into
Risk Themes
Risks

Participants of ATAM

- The Evaluation Team
  - A group external to the project. The team might be consultants or other members of the organization.
  - Typically 3–5 members, each member is assigned to a number of specific roles.
- Project Decision Makers
  - Empowered to speak for the development of project and have the authority to mandate changes to it.
  - Most importantly, the architect must be present. 2 or more members (e.g. Project Manager, Customer,…)
- Architecture Stakeholders
  - With a vested interest in the architecture performing correctly.
  - These are the stakeholder affected by the architecture, usually 12–15 members. (e.g. Modifiability – Developers, Maintainability & Security – System Administrators, etc…)

Phases of ATAM

- Phase 0: Partnership and Preparation
  - Assemble the Evaluation Team, leadership and key project decision makers
  - Prepares Architecture Description & Scenarios
- Phase 1: Evaluation – by Evaluation Team
  - Evaluation ground rules are presented
  - Business drivers (including quality attribute goals) are presented
  - The architecture itself is presented
  - Architectural approaches are identified
  - Quality attribute goals are prioritized and refined into specific scenarios
  - Architectural approaches are analyzed to determine how they satisfy the refined quality goals
- Phase 2: Evaluation – includes Stakeholders
- Phase 3: Follow-up
  - Addresses the issues raised
  - Deliver Final Report

Phase 1 - Steps

- Step 1 - Present the ATAM
  - Evaluation leader explain the process that everyone will be following and set the context and expectations
- Step 2 - Present Business Drivers
  - Decision Makers (Project Manager or Customer) presents a system overview from a business perspective
    - The system's most important functions
    - Any relevant technical, managerial, economic or political constraints
    - The business goals and context as they relate to the project
    - The major stakeholders
    - The architectural drivers (major quality attribute goals that shape the architecture)
- Step 3 - Present the Architecture
  - Architect(s) presents the architecture covering technical constraints such as operating system, hardware, or middleware prescribed for use, and other systems with which the system must interact
- Step 4 - Identify Architectural Approaches
  - Architecture Approaches or Patterns are presented at a High-Level to cover the essence of the Architecture
- Step 5 - Generate Quality Attribute Utility Tree
  - Present the quality attribute goals in detail
  - Expressed as Concrete Quality Attribute Scenarios & assign priorities
- Step 6 - Analyze Architectural Approaches
  - Determine how the approaches satisfy the refined quality goals
  - Architectural risks, sensitivity points, and tradeoff points are identified

Phase 2 - Steps

- Step 7 - Brainstorm and Prioritize Scenarios
  - A larger set of scenarios are stimulated with the entire group of stakeholders
  - Set of scenarios are prioritized via a voting process
- Step 8 - Analyze Architectural Approaches
  - Same as Step #6 but High Priority Scenarios are taken for analysis with the larger group of stakeholders
  - Document additional architectural approaches, risks, sensitivity points, and tradeoff points
- Step 9 - Present Results
  - Presents the findings to the assembled stakeholders

Outputs of ATAM

- A concise presentation of the architecture.
- Articulation of the business goals.
- A set of quality requirements in the form of quality scenarios.
- Mapping of architectural decisions to quality requirements. This specifies how the architecture supports each of the quality scenarios.
- A set of identified sensitivity and tradeoff points. Architecture decisions that affect qualities are identified along with how they cause tradeoff (e.g. performance vs. reliability).
- A set of risks and non-risks. This will be the basis of a risk mitigation plan.
- A set of risk themes. The systemic weakness that explain the risks.

ATAM - Benefits

- Identified risks early in the life cycle
- Increased communication among stakeholders
- Clarified quality attribute requirements
- Improved architecture documentation
- Documented basis for architectural decisions

ATAM – Example #1

- Introduction
  - The Case Study software architecture we are going to evaluate has a component called “game space”
  - This “game space” services a game to 2 different game engines: one is a commercial off-the-shelf game engine (Torque1) and the second is a bespoke simulation engine
  - Using a “game space” allows adding a new game engine without the need to modify the game to suit the new engine

Case Study Reference: Using ATAM to Evaluate a Game-based Architecture

http://www.cs.rug.nl/~paris/ACE2006/papers/BinSubaih.pdf

ATAM – Example #1

- Game based architecture
  - Developing games using game engines
- Problems
  - Game is developed in the game engine format – game is available for only that engine.
  - Have to modify the game when running on different game engine.

- Examines the suitability of employing ATAM to evaluate a game-based architecture for evaluating portability between game engines

Applying ATAM – Phase 1 | Step #1

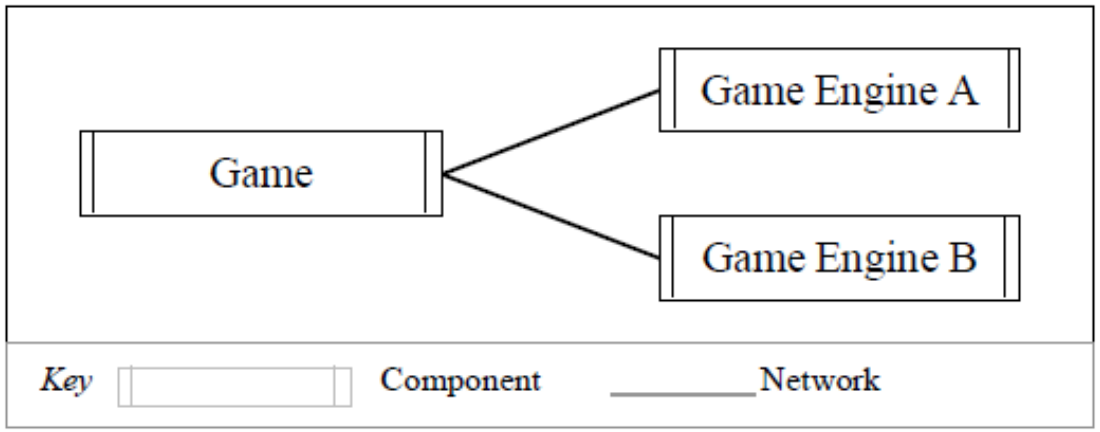
- Present ATAM
  - The first step starts by presenting the ATAM process to the stakeholders.
  - Explain the Process to the Evaluation Team
  - The evaluation leader describes the evaluation method to the assembled participants, tries to set their expectations, and answers questions they may have.

Applying ATAM – Phase 1 | Step #2

- Present Business Drivers
  - Business goals of the architecture
    - Break the current tightly coupled practices employed when developing games using game engines
    - Make the architecture flexible to accommodate different games
  - Describe
    - The system's most important functions
    - Any relevant technical, managerial, economic, or political constraints
    - The business goals and context as they relate to the project
    - The major stakeholders
    - The architectural drivers (the major quality attribute goals)

Applying ATAM – Phase 1 | Step #2

- Present Business Drivers
  - Quality Attributes
    - Portability - aims to run the same game on multiple game engines without modifying the game itself.



The portability attribute succeeds if the same game can be serviced to multiple game engines without any modification to the game

Applying ATAM – Phase 1 | Step #2

- Present Business Drivers
  - Quality Attributes
    - Modifiability - measured by the amount of changes required to the different components of the architecture and the less changes needed the better the modifiability is.
    - Performance - The stimuli can be user interactions or messages arriving over the network or method calls from other components or programs. The acceptable time to response for a stimulus is less than one second and the acceptable display rate at the game engine is no less than 20 frames per second

Applying ATAM – Phase 1 | Step #3

- Present the Architecture
  - Driving architectural requirements, measurable quantities associated with these, standards/models/approaches for meeting these
  - Important architectural information
    - Context diagram
    - Module or layer view
    - Component and connector view
    - Deployment view

Applying ATAM – Phase 1 | Step #3

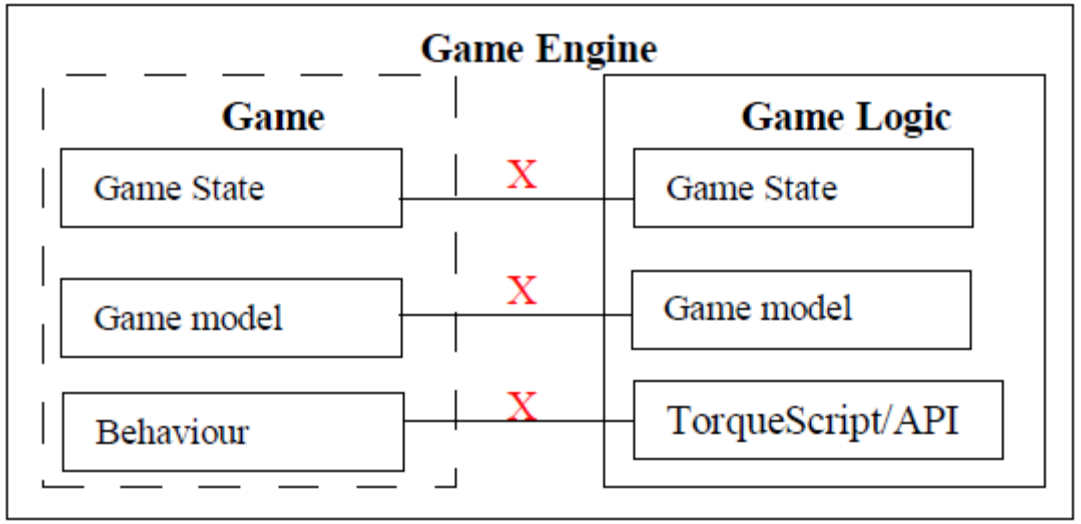
- Present the Architecture
  - Architectural approaches, patterns, tactics employed, what quality attributes they address and how they address those attributes
  - Use of COTS and their integration
  - Most important use case scenarios
  - Most important change scenarios
  - Issues/risk for meeting the diving requirements

Applying ATAM – Phase 1 | Step #4

- Identify Architectural Approaches
  - Catalog the evident patterns and approaches
    - Based on step 3
    - Serves as the basis for later analysis

Applying ATAM – Phase 1 | Step #4

- Identify Architectural Approaches
  - Architectural decisions to support portability
    - Have the game state living outside the game engine's game state and find a way to communicate between the two states
    - Model view controller (MVC) pattern
    - Asynchronous messaging



Applying ATAM – Phase 1 | Step #4

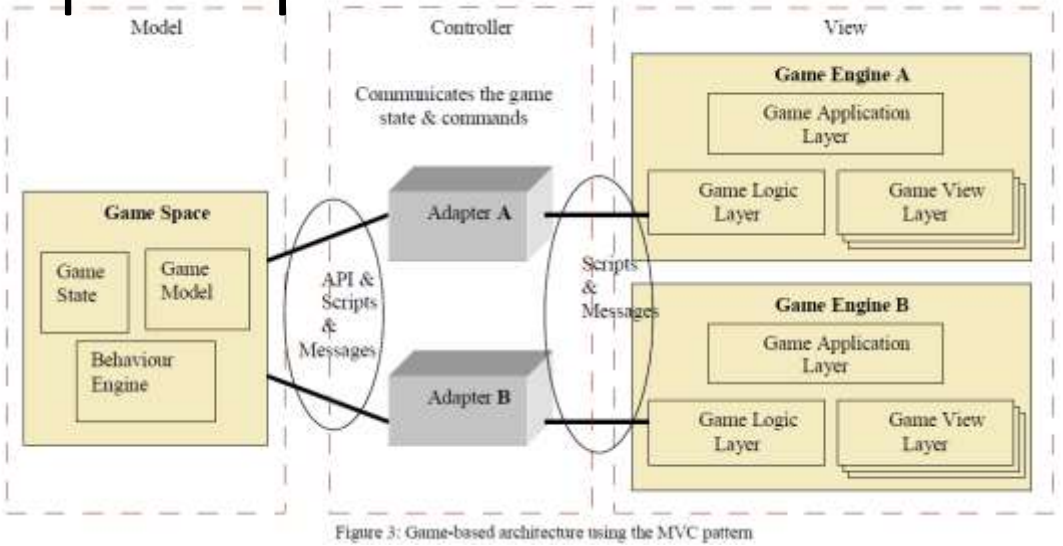


Figure 3: Game-based architecture using the MVC pattern

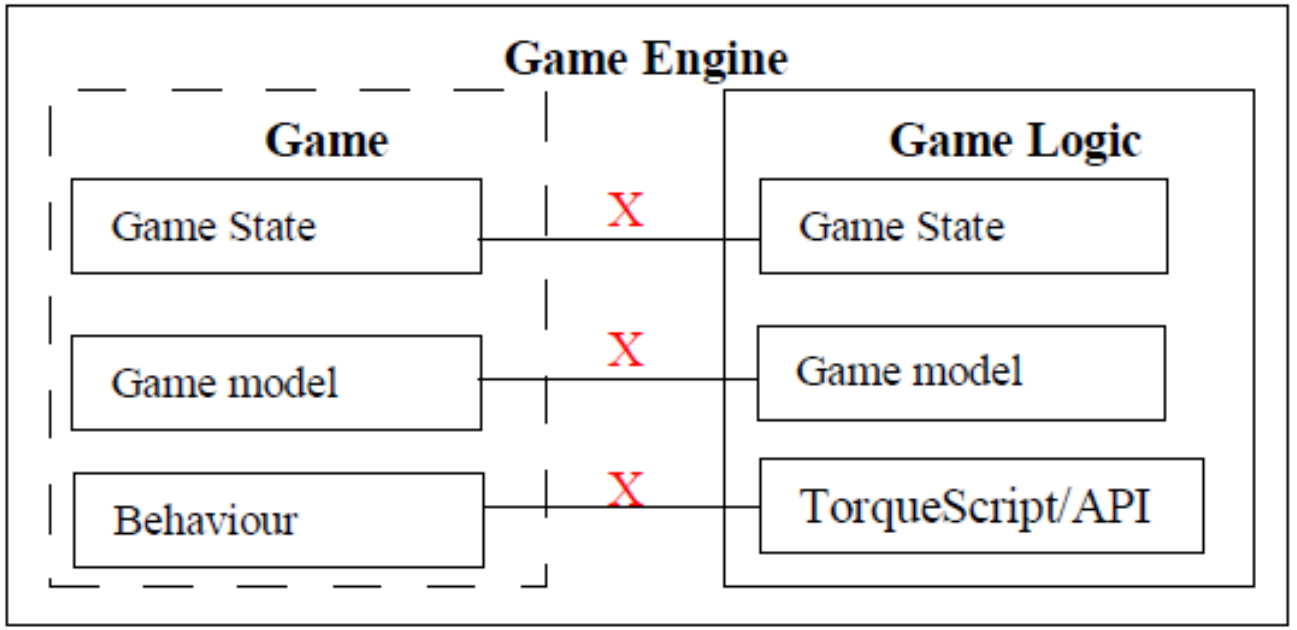
- MVC Architecture
  - Breaks dependency links – Portability
  - Separates core from view – Modifiability
    - Anticipate expected changes
    - Semantic coherence
    - Prevent ripple effects
  - The introduction of layers by using MVC to promote portability and modifiability also adds processing overhead on the architecture as information needs to be passed and sometimes translated between the layers.
  - Placing adapters
    - in its own application on its own machine – network overhead
    - in its own application on either the game space machine or the game engine machine - extra overhead to communicate across applications
    - placing the adapter in the same application as the game space - least overhead expense

Applying ATAM – Phase 1 | Step #4

- Asynchronous Messaging
  - Synchronization of the game states
  - Messages from behaviour engine to be carried out to the game engine
  - To compensate for the network overhead the architecture uses an asynchronous communication mechanism to reduce the impact on the display rate.

Applying ATAM – Phase 1 | Step #4

- Architectural Decisions
  - To support portability: not to have a game model that is only accessible through the game engine's interface but have a game model that can be accessed outside the game engine
  - Ontologies
  - Mid-game scripting



Applying ATAM – Phase 1 | Step #4

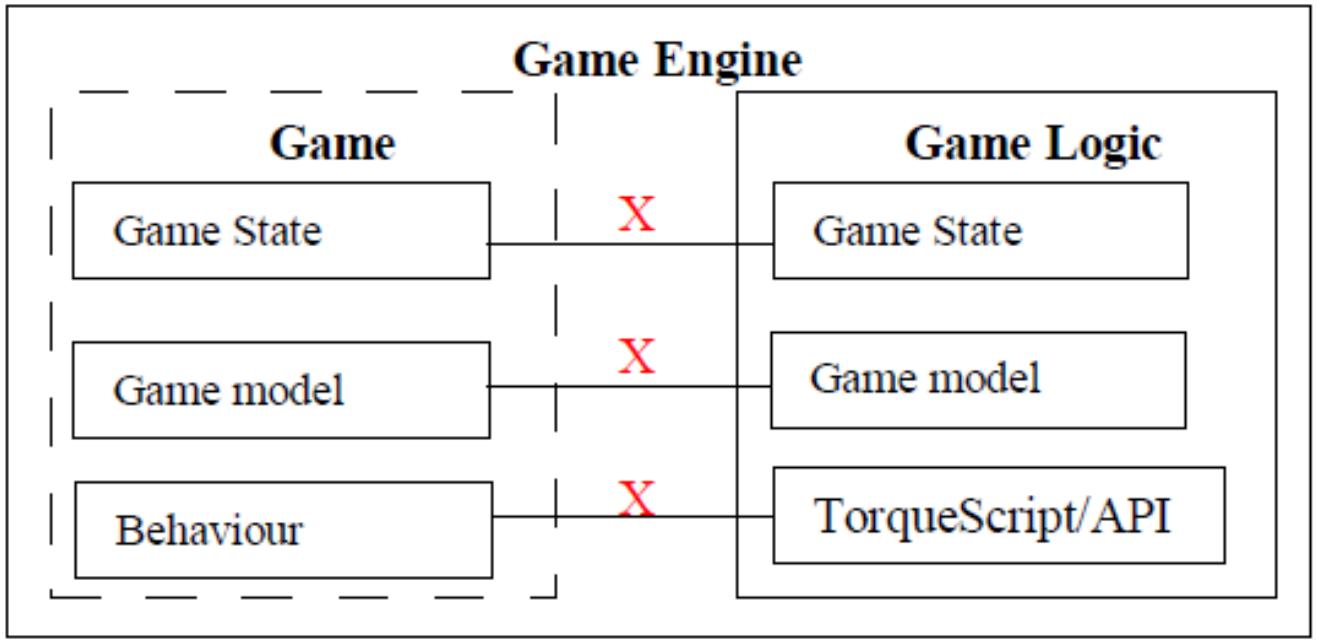
- Ontologies
  - Ontologies allow creating concepts which have properties, and relations. We use these to create the classes of our domain. The concept becomes the class and its properties are the properties of the class and finally the relationships describe things like inheritance and association.
  - Used to specify the game knowledge representation independently from the game engine's game model.
  - Allows building classes on-the-fly

Applying ATAM – Phase 1 | Step #4

- Scripting
  - Enables manipulating the game state and setting the behaviour without recompiling the game space system.
  - Can modify game engine without affecting the game engine code
  - Mid-game scripting is used instead of pre-compiled scripting since using the latter would require some of the game logic to be put in the game engine's specific format.
  - For better modifiability mid-game scripting was used despite the fact that mid-game scripting runs at much slower speed than the precompiled code

Applying ATAM – Phase 1 | Step #4

- Architectural Decisions
  - To support Portability: final flag should be raised when the game engine requires the behavior to be specified in the game engine's own language or format.
  - Objects mapping table
  - API



Applying ATAM – Phase 1 | Step #4

- API and Mapping
  - API - to interact with the game space through code or through scripting
  - Objects mapping table – link the corresponding objects on both sides

Applying ATAM – Phase 1 | Step #5

- Generate Quality Attribute Utility Tree
  - Utility Tree
    - Present the quality attribute goals in detail
  - Quality attribute goals
    - Identified, prioritized, refined
    - Expressed as scenarios
  - Utility is an expression of the overall goodness of the system
    - Quality attributes form the second level being components of utility
  - Prioritize Scenarios
    - Depending on how important they are
    - Depending on how difficult it will be for the architecture to satisfy a scenario

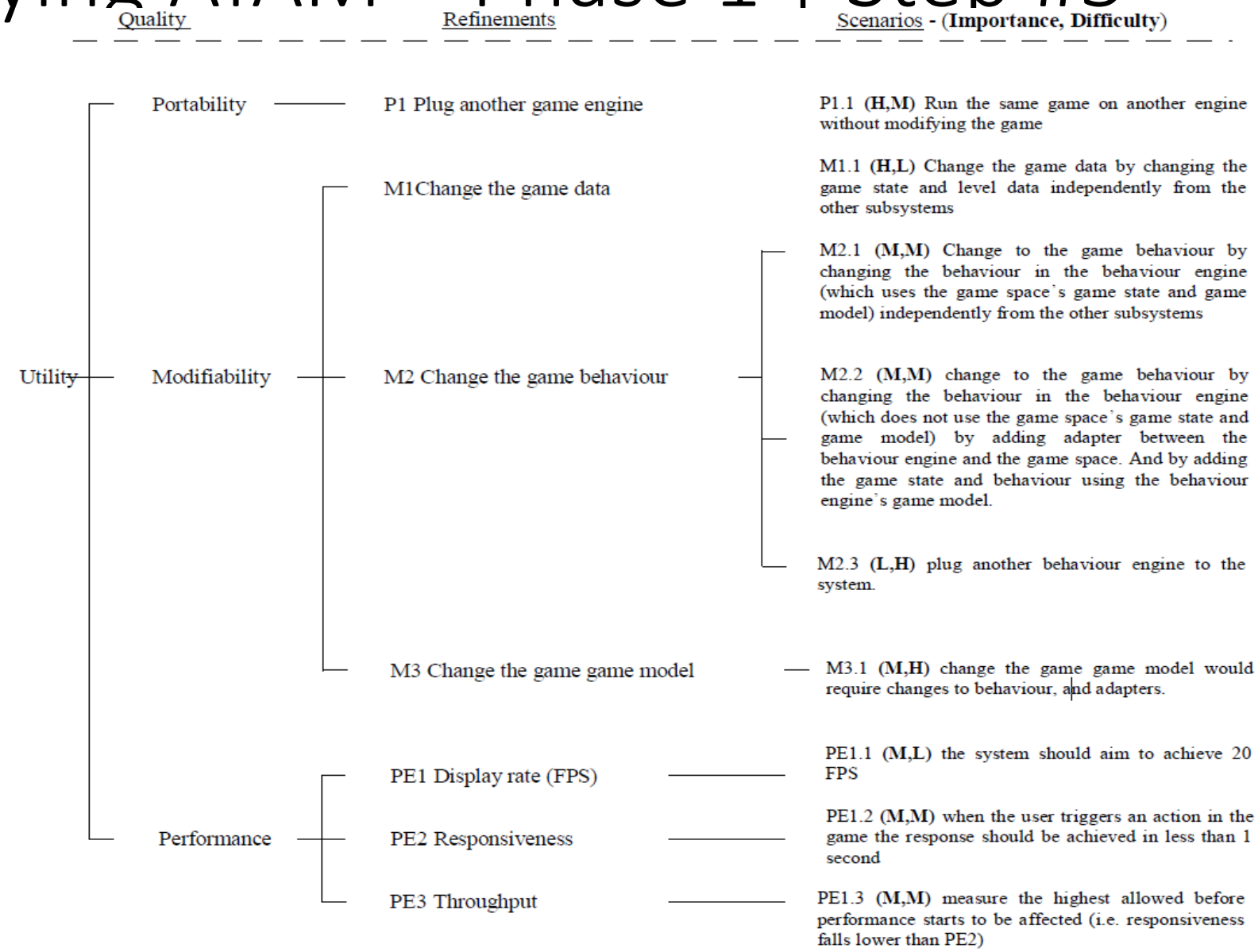
Applying ATAM – Phase 1 | Step #5

- Generate Utility Tree
  - Identify, prioritize, and refine the most important quality attribute goals by building a utility tree.
  - A utility tree is a top-down vehicle for characterizing the "driving" attribute-specific requirements
  - Select the most important quality goals to be the high-level nodes (typically performance, modifiability, security, and availability)
  - Scenarios are the leaves of the utility tree
    - Output: a characterization and a prioritization of specific quality attribute requirements.
  - Scenarios are used to represent stakeholders' interests
    - Understand quality attribute requirements
    - A good scenario makes clear what the stimulus is that causes it and what responses are of interest.

Applying ATAM – Phase 1 | Step #5

- Quality Attribute Utility Tree Template
  - Quality Attribute Level
  - Refinements Level
    - The aim of the refinement is to decompose the quality attribute further if possible
  - Scenario Level
    - Holds the scenarios in the Concrete form
    - Holds Scenario Rankings
      - Importance/Priority – High/Medium/Low or 1 – 10 rating
      - Difficulty Factor – High/Medium/Low or 1 – 10 rating

Applying ATAM – Phase 1 | Step #5



Applying ATAM – Phase 1 | Step #6

- Analyze Architectural Approaches
  - Examine the highest ranked scenarios
  - The goal is for the evaluation team to be convinced that the approach is appropriate for meeting the attribute-specific requirements
  - Scenario walkthroughs
  - Identify and record a set of sensitivity points and tradeoff points, risks and non-risks
    - Sensitivity and tradeoff points are candidate risks

Applying ATAM – Phase 1 | Step #6

- Concrete Quality Attribute Scenarios & assign priorities: Portability

Analysing Scenario	P1.1			
Scenario	plug another game engine to the architecture			
Attributes	Portability			
Stimulus	running the same game on another game engine			
Environment				
Response	should be achieved by adding an adapter to connect the game space and the newly added game engine			
Architecture Decision	**Sensitivity**	**Tradeoff**	**Risk**	**Nonrisk**
AD1 MVC		T1	R1,R2	N1
AD2 Messaging	S1,S2	T2	R3	N2
AD3 Mid-game scripting		T3	R4	N3
AD4 Ontologies		T4		
AD6 COTS behaviour engine	S1,S2		R2	N4

Applying ATAM – Phase 1 | Step #6

Sensitivities:

- S1: Concern over network latency
- S2: Concern over messages load
- S3: In the event that a single unique identifier cannot be set this architecture decision becomes very sensitive to any modification as they have to be added manually to the table
- S4: Using ontologies allows for better game model scalability but it makes the architecture very sensitive to change as the change propagates to behaviour and adapters.

Tradeoffs:

- T1: Portability (+) and Modifiability (+) vs. Performance (-) – separating the architecture into layers adds an overhead for exchanging information between layers which affects the performance.
- T2: Portability (+) vs. Performance (-)
- T3: Portability (+) and Modifiability (+) vs. Performance (-) – mid-game scripting allows better portability and modifiability but runs at 10x slower than pre-compiled code.
- T4: Portability (+) and Modifiability (+) vs. Performance (-) – using ontologies allows for having a game model that is independent from the game engine's game model however it has adverse effect on performance.
- T5: Modifiability (-) vs. Performance (+)

Risks:

- R1: The risk is caused by the tight coupling between the controller and the model which is a known liability of using this pattern.
- R2: Data integrity
- R3: The risk is introduced by the tight coupling as a consequence of using pre-determined messages for the messages arriving form the game engine which means changes to them would require deployment of new system every time.
- R4: The risk raised is a consequence the game engine or the game space which might not have fully exposed their functionality through the scripting.
- R5: If no unique id can be set this means the mapping table should be done manually.

Nonrisks:

- N1: The nonrisk exists because of the decision taken to remove the other liability of using MVC – the removal of the direct link between the view and the controller described earlier.
- N2: The nonrisk risk exists because of using asynchronous mechanism as it avoids negative impact on the frame rate that could be caused by the synchronous mechanism
- N3: The nonrisk is the use of API approach which should stay compatible.
- N4: assume that the behaviour engine requires the game state to be replicated to its own working memory
- N5: A one-to-one mapping for objects across both game states is established by having a unique identifier and if that is not possible the objects mapping table handles that.
- N6: Should stay compatible

Applying ATAM – Phase 1 | Step #6

- Sensitivities:
  - S1: Concern over network latency
  - S2: Concern over messages load
  - S3: In the event that a single unique identifier cannot be set this architecture decision becomes very sensitive to any modification as they have to be added manually to the table
  - S4: Using ontologies allows for better game model scalability but it makes the architecture very sensitive to change as the change propagates to behaviour and adapters.

Applying ATAM – Phase 1 | Step #6

- Tradeoffs:
  - T1: Portability (+) and Modifiability (+) vs. Performance (-) – separating the architecture into layers adds an overhead for exchanging information between layers which affects the performance.
  - T2: Portability (+) vs. Performance (-)
  - T3: Portability (+) and Modifiability (+) vs. Performance (-) – mid-game scripting allows better portability and modifiability but runs at 10x slower than pre-compiled code.
  - T4: Portability (+) and Modifiability (+) vs. Performance (-) – using ontologies allows for having a game model that is independent from the game engine's game model however it has adverse effect on performance.
  - T5: Modifiability (-) vs. Performance (+)

Applying ATAM – Phase 1 | Step #6

- Risks:
  - R1: The risk is caused by the tight coupling between the controller and the model which is a known liability of using this pattern.
  - R2: Data integrity
  - R3: The risk is introduced by the tight coupling as a consequence of using pre-determined messages for the messages arriving form the game engine which means changes to them would require deployment of new system every time.
  - R4: The risk raised is a consequence the game engine or the game space which might not have fully exposed their functionality through the scripting.
  - R5: If no unique id can be set this means the mapping table should be done manually.

Applying ATAM – Phase 1 | Step #6

- Non-risks:
  - N1: The non-risk exists because of the decision taken to remove the other liability of using MVC – the removal of the direct link between the view and the controller described earlier.
  - N2: The non-risk risk exists because of using asynchronous mechanism as it avoids negative impact on the frame rate that could be caused by the synchronous mechanism
  - N3: The non-risk is the use of API approach which should stay compatible.
  - N4: assume that the behaviour engine requires the game state to be replicated to its own working memory
  - N5: A one-to-one mapping for objects across both game states is established by having a unique identifier and if that is not possible the objects mapping table handles that.
  - N6: Should stay compatible

Applying ATAM – Phase 1 → Phase 2

- After Phase 1, there is a detailed set of quality requirements with the architectural decision tradeoffs that affect those requirements.
- In Phase 2,
  - The stakeholders help to prioritize the existing scenarios and propose new ones.
  - The architectural tradeoffs are again analyzed, now in consultation with the stakeholders.
- The results of the review are presented to all.
- During Phase II, in addition to brainstorming on quality scenarios, the stakeholders provide the in-depth knowledge of whether the proposed architectural tactics will actually meet their expectations.

Applying ATAM – Phase 1 → Phase 2

- Scenario Examples
  - Use case scenario
    - Remote user requests a database report via the Web during peak period and receives it within 5 seconds.
  - Growth scenario
    - Add a new data server to reduce latency in scenario 1 to 2.5 seconds within 1 person-week.
  - Exploratory scenario
    - Half of the servers go down during normal operation without affecting overall system availability.
  - Scenarios should be as specific as possible

Applying ATAM – Phase 1 → Phase 2

Type	Scenario
Use case	- Run two games from the same domain - Run two games from two different domains
Growth	- Interoperate between two game engines - Run two behaviour engines (internal and external)
Exploratory	- Increase the load by increasing the number of NPCs the user can interact with - Increase the load by simulating multiple number of simultaneous players - Run the game space on the same machine as the game engine - Run the behaviour engine and the game space on the same machine as the game engine

Applying ATAM – Phase 1 → Phase 2

- Sensitivity & Tradeoffs
  - Stakeholders will join form this step.
  - Sensitivity:
    - A property of a component that is critical to success of system.
    - The number of simultaneous database clients will affect the number of transaction a database can process per second. This assignment is a sensitivity point for the performance
    - Keeping a backup database affects reliability.
    - Power of encryption (Security) sensitive to number of bits of the key
  - Tradeoff point:
    - A property that affects more than one attribute or sensitivity point.
    - In order to achieve the required level of performance in the discrete event generation component, assembly language had to be used thereby reducing the portability of this component.
    - Keeping the backup database affects performance also so it's a trade-off between reliability and performance.

Applying ATAM – Phase 1 → Phase 2

- Risks & Non-Risks
  - Risks
    - The decision to keep backup is a risk if the performance cost is excessive
    - Rules for writing business logic modules in the second tier of your 3-tier style are not clearly articulated. This could result in replication of functionality thereby compromising modifiability of the third tier.
  - Non-Risks
    - The decision to keep backup is a non-risk if the performance cost is not excessive
    - Assuming message arrival rates of once per second, a processing time of less than 30 ms, and the existence of one higher priority process, a 1 second soft deadline seems reasonable Performance.

Applying ATAM – Phase 2 | Step #7

- Brainstorm & Prioritize Scenarios
  - Utility tree shows architects view on the quality attributes
  - Stakeholders generate scenarios using a facilitated brainstorming process.
  - Scenarios at the leaves of the utility tree serve as examples to facilitate the step.
  - The new scenarios are added to the utility tree
  - Here the focus is on the other stakeholders view on the quality attributes and scenarios based on these
    - Which are the mot meaningful and important scenarios for users etc.

Applying ATAM – Phase 2 | Step #8

- Analyze Architectural Approaches
  - Highest ranked scenarios from Step #7 are presented to the architect
  - Explain how relevant architectural decisions contribute to realizing each one

Applying ATAM – Phase 2 | Step #9

- Present Results (Output)
  - The architectural approaches documented
  - The set of scenarios and their prioritization from the brainstorming
  - The utility tree
  - The risks discovered
  - The non-risks documented
  - The sensitivity points and tradeoff points found

Applying ATAM – Phase 2

- Benefits and Observations
  - Technical participants in ATAM are typically amazed at how many risks can be found in a short time.
  - Managers appreciate the opportunity to see precisely how technical issues threaten achievement of their business goals.

References

- https://www.sei.cmu.edu/architecture/tools/evaluate/atam.cfm
- http://etutorials.org/Programming/Software+architecture+in+practice,+second+edition/Part+Three+Analyzing+Architectures/Chapter+11.+The+ATAM+A+Comprehensive+Method+for+Architecture+Evaluation/
- Using ATAM to Evaluate a Game-based Architecture - http://www.cs.rug.nl/~paris/ACE2006/papers/BinSubaih.pdf

Exercise (offline)

- Refer a few existing software systems (e.g. your Group Case Study) and identify what are the important quality attributes for that system
  - Identify about 2-3 Quality Attributes
  - Write Concrete Quality Attribute Scenarios
  - Generate Quality Attribute Utility Tree
  - Identify Sensitivity points, Trade-offs, Risks and Non-Risks