Carnegie Mellon
Software Engineering Institute

Architecture Tradeoff Analysis Method[SM] (ATAM[SM])

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Role of a Software Architecture

If the only criterion for software was to get the right answer, we would not need architectures.

Unstructured, monolithic systems would suffice.

But other things also matter, such as:

- modifiability
- time of development
- performance
- coordination of work teams

These issues are often addressed in the Software Architecture

Why Analyze Software Architectures?

All design involves tradeoff in system qualities

- System qualities are largely dependent on architectural decisions
- Promoting one quality often comes at the expense of another quality

A software architecture is the earliest life-cycle artifact that embodies significant design decisions: choices and tradeoffs.

- Choices are easy to make, but hard to change once the system is implemented

The ATAM

SEI has developed the Architecture Tradeoff Analysis Method (ATAM) over several years.

The purpose of ATAM is: *to assess the consequences of architectural decision alternatives in light of quality attribute requirements*.

Purpose of ATAM - 1

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

- **Discover *risks* - alternatives that might create future problems in some quality attribute**
- **Discover *non-risks* - decisions that promote qualities that help realize business/mission goals**
- **Discover *sensitivity points* - alternatives for which a slight change makes a significant difference in some quality attribute**
- **Discover *tradeoffs* - decisions affecting more than one quality attribute**

Purpose of ATAM - 2

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.

Surfaced tradeoffs can be explicitly identified and documented.

ATAM Benefits

There are a number of benefits from performing ATAM analyses:

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

The results are improved architectures.

Purpose of ATAM

The purpose of ATAM is to assess the consequences of architectural decisions in light of quality attribute requirements.

The ATAM process is a short, facilitated interaction between multiple stakeholders, leading to the identification of risks, sensitivities, and tradeoffs.

The purpose of an ATAM is NOT to provide precise analyses, the purpose IS to discover risks created by architectural decisions.

Preconditions for an ATAM

1. Clients must have a Software Architecture
   - Scope/scale must be manageable
   - ATAM *will not work* if the software architecture has not been created yet
   - ATAM team members will review architectural artifacts, and may help refine documentation
   - Architect must prepare an architecture presentation
2. Clients must prepare a business/mission goals presentation
3. ATAM will review architecture artifacts, presentations, and read ahead material to become familiar with domain

Evaluation Team

Each ATAM team consists of a leader and at least three other team members

- domain expertise is not necessary
- ATAM team members must be experienced architects
- ATAM leaders must have EXCELLENT communication and facilitation skills

The ATAM team members fill multiple roles during the course of the evaluation.

Evaluation Team Roles - 1

Moderator — *facilitates discussions, brainstorming, analysis*

Scenario scribe(s) — *writes utility tree, raw scenarios, risks, sensitivities, tradeoffs on flip-charts or whiteboards*

Proceedings scribe — *captures scribe's writing on a laptop computer, preparing the Results Presentation template*

Evaluation Team Roles - 2

Process enforcer/observer — *monitors the process steps, takes notes about the process, and how it could be improved*

Timekeeper — *informs the evaluation leader when the time allocated for a step has expired*

Questioner(s) — *raise issues that the stakeholders have not thought of; asks questions based on how quality attributes of interest relate to architectural styles*

Basic Rules for ATAM Team Members

- Keep the process moving!
- Ask questions
- Propose scenarios
- Write down exactly what stakeholders say; do not "edit" their words!

ATAM Steps

1. Present the ATAM
2. Present business drivers
3. Present architecture
4. Identify architectural approaches
5. Generate quality attribute utility tree
6. Analyze architectural approaches
7. Brainstorm and prioritize scenarios
8. Analyze architectural approaches
9. Present results

Phase I

Phase II

1. Present the ATAM

Evaluation Team presents an overview of the ATAM including:

- ATAM steps in brief
- Techniques
  - utility tree generation
  - architecture elicitation and analysis
  - scenario brainstorming/mapping
- Outputs
  - architectural approaches
  - utility tree
  - scenarios
  - risks and “non-risks”
  - sensitivity points and tradeoffs

2. Present Business Drivers

ATAM customer representative describes the system's business drivers including:

- **Business context for the system**
- **High-level functional requirements**
- **High-level quality attribute requirements**
  - **architectural drivers: quality attributes that "shape" the architecture**
  - **critical requirements: quality attributes most central to the system's success**

3. Present Architecture

Architect presents an overview of the architecture including:

- **Technical constraints such as an OS, hardware, or middle-ware prescribed for use**
- **Other systems with which the system must interact**
- **Architectural approaches/styles used to address quality attribute requirements**

Evaluation team begins probing for and capturing risks.

ATAM Steps

1. Present the ATAM
2. Present business drivers
3. Present architecture
4. Identify architectural approaches
5. Generate quality attribute utility tree
6. Analyze architectural approaches
7. Brainstorm and prioritize scenarios
8. Analyze architectural approaches
9. Present results

Phase I

Phase II

4. Identify Architectural Approaches

Start to identify places in the architecture that are key for realizing quality attribute goals.

Identify any predominant architectural approaches.

Examples:

- **client-server**
- **3-tier**
- **watchdog**
- **publish-subscribe**
- **redundant hardware**

5. Generate Quality Attribute Utility Tree

Identify, prioritize, and refine the most important quality attribute goals by building a *utility tree*.

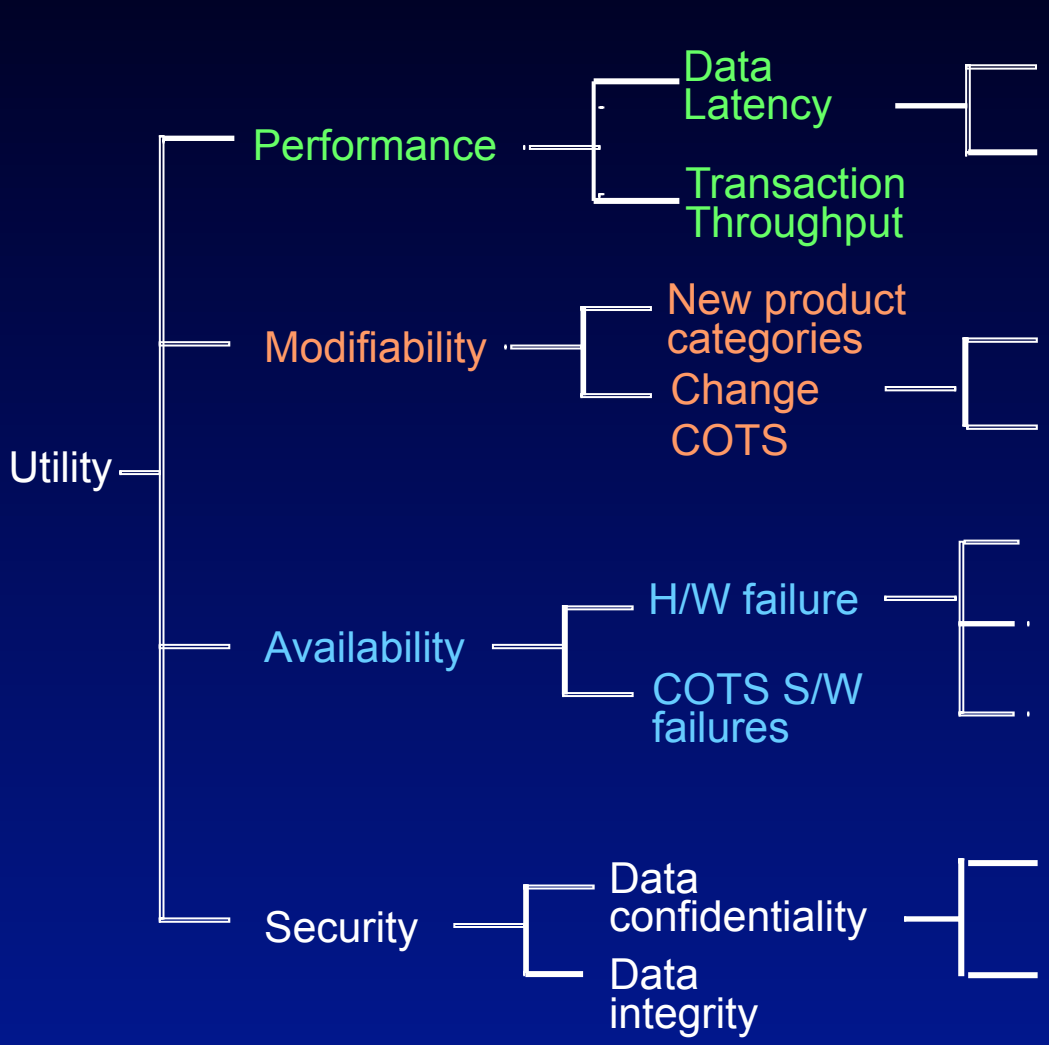
- A utility tree is a top-down vehicle for characterizing the "driving" attribute-specific requirements
- Select the most important quality goals to be the high-level nodes (typically performance, modifiability, security, and availability)
- Scenarios are the leaves of the utility tree

Output: a characterization and a prioritization of specific quality attribute requirements.

High/Medium/Low importance for the success of the system

High/Medium/Low difficulty to achieve (architect's assessment)

Utility Tree Construction -1



Reduce storage latency on customer DB to < 200 ms.

Deliver video in real time

Add CORBA middleware in < 20 person-months

Change web user interface in < 4 person-weeks

Power outage at site1 requires traffic redirected to site2 in < 3 seconds.

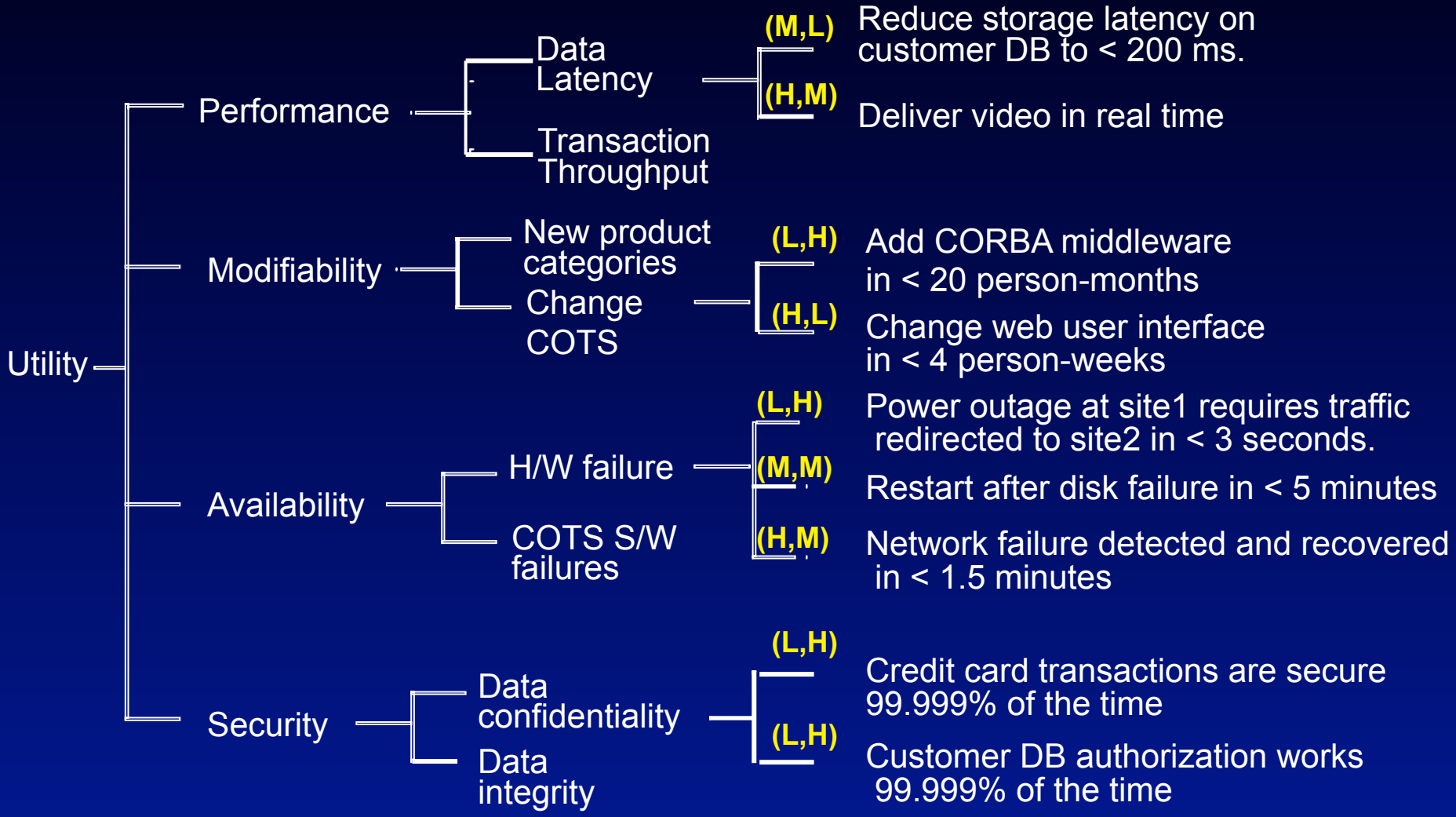
Restart after disk failure in < 5 minutes

Network failure detected and recovered in < 1.5 minutes

Credit card transactions are secure 99.999% of the time

Customer DB authorization works 99.999% of the time

Utility Tree Construction -2



Scenarios

Scenarios **are used to**

- **Represent *stakeholders'* interests**
- **Understand quality attribute requirements**

Scenarios should cover a range of

- **Anticipated uses of (use case scenarios),**
- **Anticipated changes to (growth scenarios), or**
- **Unanticipated stresses (exploratory scenarios) to the system.**

A good scenario makes clear what the stimulus is that causes it and what responses are of interest.

Example Scenarios

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.

Stimuli, Environment, Responses

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.

6. Analyze Architectural Approaches

Evaluation Team probes architectural approaches from the point of view of specific quality attributes to identify risks.

- **Identify the approaches that pertain to the highest priority quality attribute requirements**
- **Generate quality-attribute specific questions for highest priority quality attribute requirement**
- **Ask quality-attribute specific questions**
- **Identify and record risks and non-risks, sensitivity points and tradeoffs**

Quality Attribute Questions

Quality attribute questions probe styles to elicit architectural decisions which bear on quality attribute requirements.

Performance

- How are priorities assigned to processes?
- What are the message arrival rates?

Modifiability

- Are there any places where layers/facades are circumvented ?
- What components rely on detailed knowledge of message formats?

Risks and Non-Risks

Example Risks

- *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.*

Example Non-Risk

- *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.*

Sensitivities and Tradeoffs

Example Sensitivity

- *Changing the timing scheme from a harmonic framework to a non-harmonic framework would be easy, but due to implied timing dependencies, there would be far reaching impacts to other modules.*

Example Tradeoffs

- *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.*

ATAM Steps

1. Present the ATAM
2. Present business drivers
3. Present architecture
4. Identify architectural approaches
5. Generate quality attribute utility tree
6. Analyze architectural approaches
7. Brainstorm and prioritize scenarios
8. Analyze architectural approaches
9. Present results

Phase I

Phase II

7. Brainstorm and Prioritize Scenarios

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

Each stakeholder is allocated a number of votes roughly equal to 0.3 x #scenarios.

8. Analyze Architectural Approaches

Identify the architectural approaches impacted by the scenarios generated in the previous step.

This step continues the analysis started in step 6 using the new scenarios.

Continue identifying risks and non-risks.

Continue annotating architectural information.

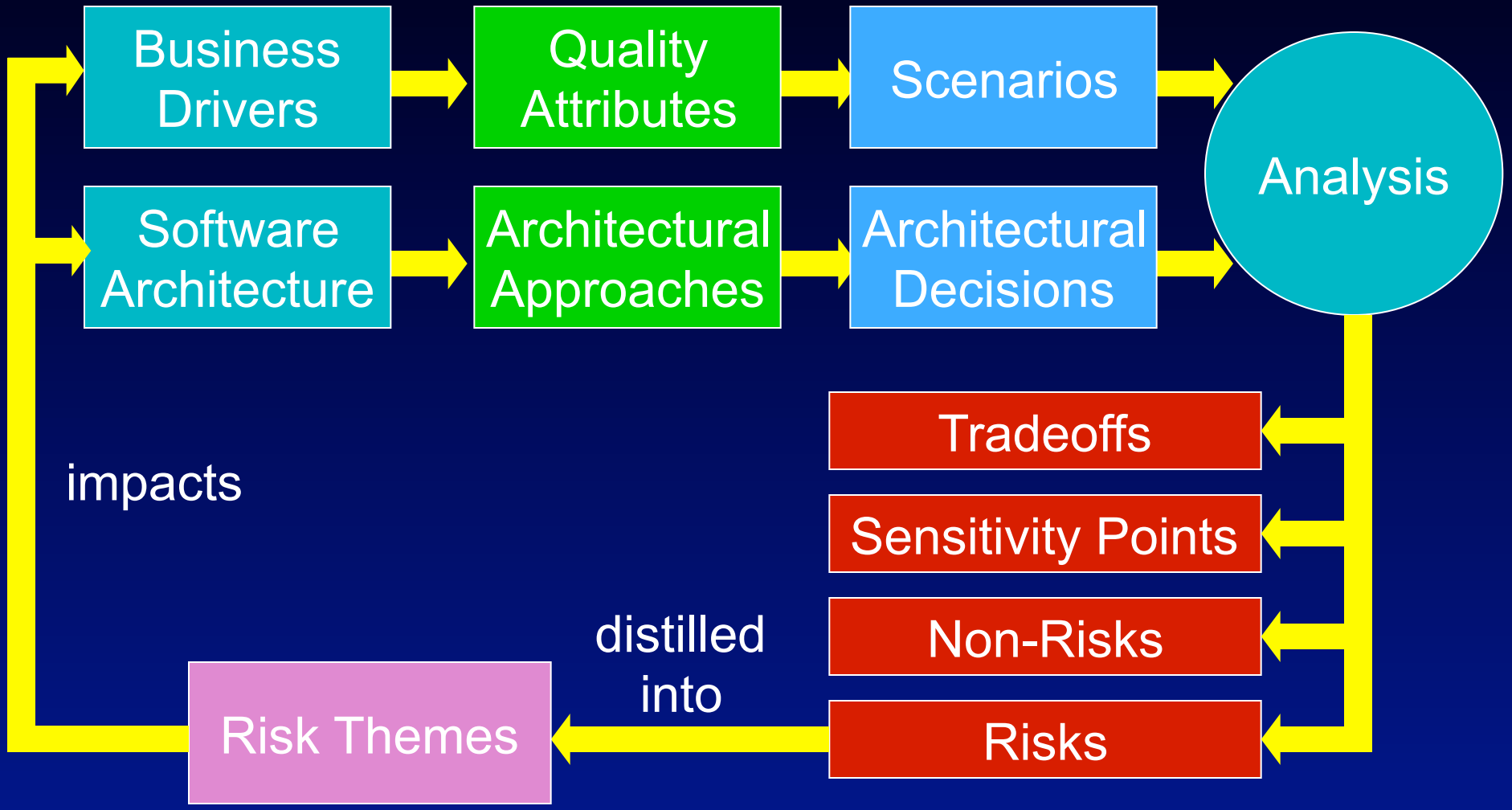
9. Present Results

Recapitulate steps of the ATAM

Present ATAM outputs

- architectural approaches
- utility tree
- scenarios
- risks and “non-risks”
- sensitivity points and tradeoffs

Conceptual Flow of ATAM

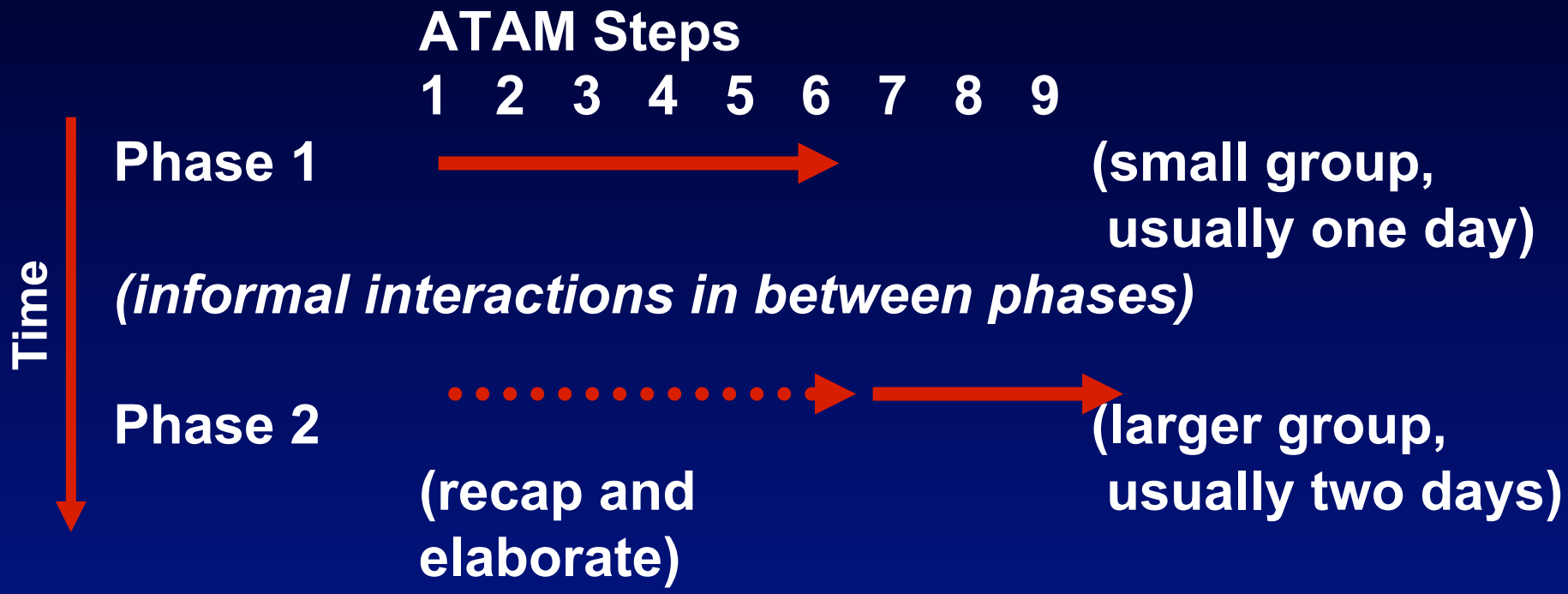


ATAM Nominal Phases - 1

ATAM evaluations are often conducted in two stages or phases:

- **During phase 1 the architect describes the quality attribute goals and how the architecture meets these goals**
- **During phase 2 we determine if a larger group of stakeholders agrees with the goals and the results**

ATAM Nominal Phases - 2



ATAM versus QAW

ATAM

- Need architecture
- Focused on:
  - business/mission goals
  - quality attributes
  - architecture decisions
- Scenario-driven
- Proven to be useful for software architectures
- Analysis done by evaluation team
- Short duration

QAW (quality attribute workshops)

- Need requirements
- Quality-attribute focused
- Scenario-driven
- Proven to be useful at system level
  - helps define software's role in overall system
- Analysis done by developers, designers ~ reviewed by evaluation team
- Iterative, extended duration

When to use ATAM

Academically, the time to use ATAM is right after the architecture has been specified when there is little or no code.

However, in practice, ATAM has been very effective in the following situations:

- Evaluating alternative candidate architectures
- Evaluating existing systems prior to committing to major upgrades
- Deciding between upgrade or replace

JNTF ATAM Case Study

Time permitting, this case study illustrates a typical ATAM for a contractor/government organization on a very large simulation system…

JNTF ATAM Case Study

A case study on a government-contractor team is developing the Wargame 2000 system at the Joint National Test Facility (JNTF), Colorado.

This case study presents the contextual background about the software architecture, the organization, the system being evaluated, and the ATAM results.

This ATAM is an SEI TR: CMU/SEI-2001-TN-022
http://www.sei.cmu.edu/pub/documents/01.reports/pdf/01tn022.pdf

JNTF ATAM Case Study

The Joint National Integration Center supports the evolutionary development and incremental fielding of an overarching ballistic missile defense system for the USA.

To do this, the JNIC performs interoperability tests; develops models and simulations; hosts and supports missile defense-related wargames; and provides missile defense exercise support, system-level engineering support, and related analyses.

JNTF ATAM Case Study

JNTF ATAM evaluation background:

- **Conducted November 2000**
- **Wargame 2000 system was relatively far along in the development cycle**
- **Primary goal of the ATAM was to obtain a measure of confidence in the system's software architecture**
  - **Still early enough to redirect development, if necessary.**

JNTF Mission Drivers

Wargame 2000 puts human operators in a realistic simulated combat command and control environment.

This drives the need to have a system with the resolution and fidelity to allow mission analysts to answer the "Big Five Questions":

1. **What did the operator know?**
2. **When did the operator know it?**
3. **What did the operator do?**
4. **When did the operator do it?**
5. **What effect did their actions have on the battle?**

JNTF Mission Drivers

What did the operator know?

- need realistic displays and complete display contents.

When did the operator know it?

- need end-to-end timing, accurate within a second.

What did the operator do?

- drives the need for a realistic set of controls, rules of engagement, decision aids, command structure coordination.

JNTF Mission Drivers

When did the operator do it?

- drives the need for deterministic and reliable response to operator actions

What effect did the operators actions have on the battle?

- drives the need for detailed data storage, retrieval, and analysis

Driving System Qualities

Performance – has multiple facets:

- Margin: The term *margin* means how much faster than real time the system can run in a worst case scenario. The more margin, the more features that can be offered to the customer in the future
- Firm real-time: The system must meet statistical deadlines
- Turn-around time: The Wargame 2000 system must undergo preparation prior to a simulation. While this is not a traditional interpretation of performance, it certainly is an important driver for Wargame 2000.

Driving System Qualities

Modifiability: *The ability to change the system to incorporate new scenarios, entities, and entity behavior; this includes changes in breadth (new entities) and depth (entity detail).*

Integrability: *The system must be able to easily connect new and dissimilar "parts" over time.*

Flexibility: *The system must be flexible in terms of changing direction by incorporating new requirements and mapping new functionality.*

Interoperability: *Wargame 2000 must be able to connect to and communicate with other simulations. Must be compliant with the High Level Architecture Standard (HLA).*

Driving System Qualities

Scalability: *The system must be able to instantiate models up to the limits of memory and processor capacity.*

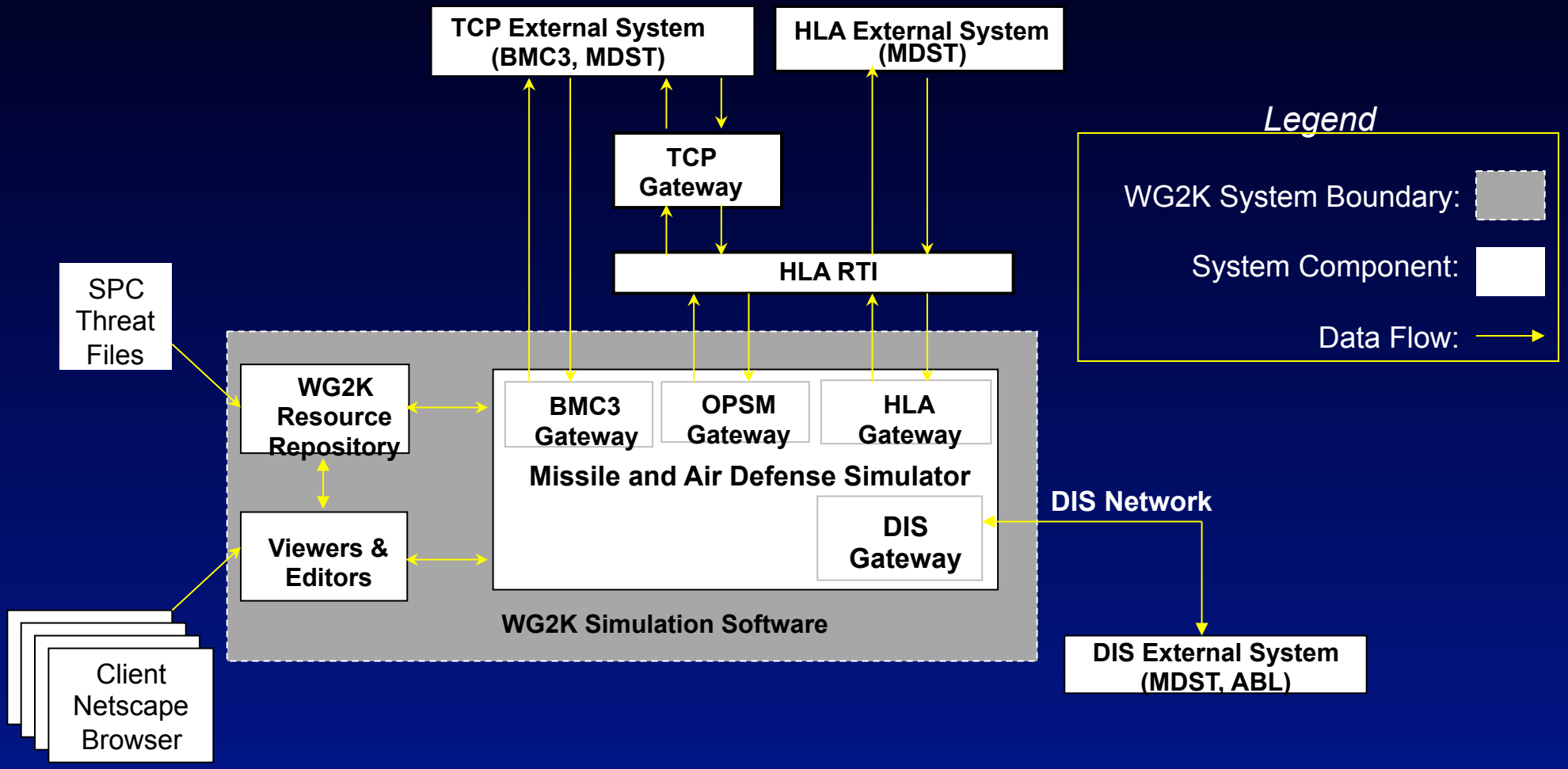
Reliability: *The cost to prepare for and conduct an exercise is very high. "On game day, its got to work."*

Maintainability: *Good system documentation is necessary for long-term supportability.*

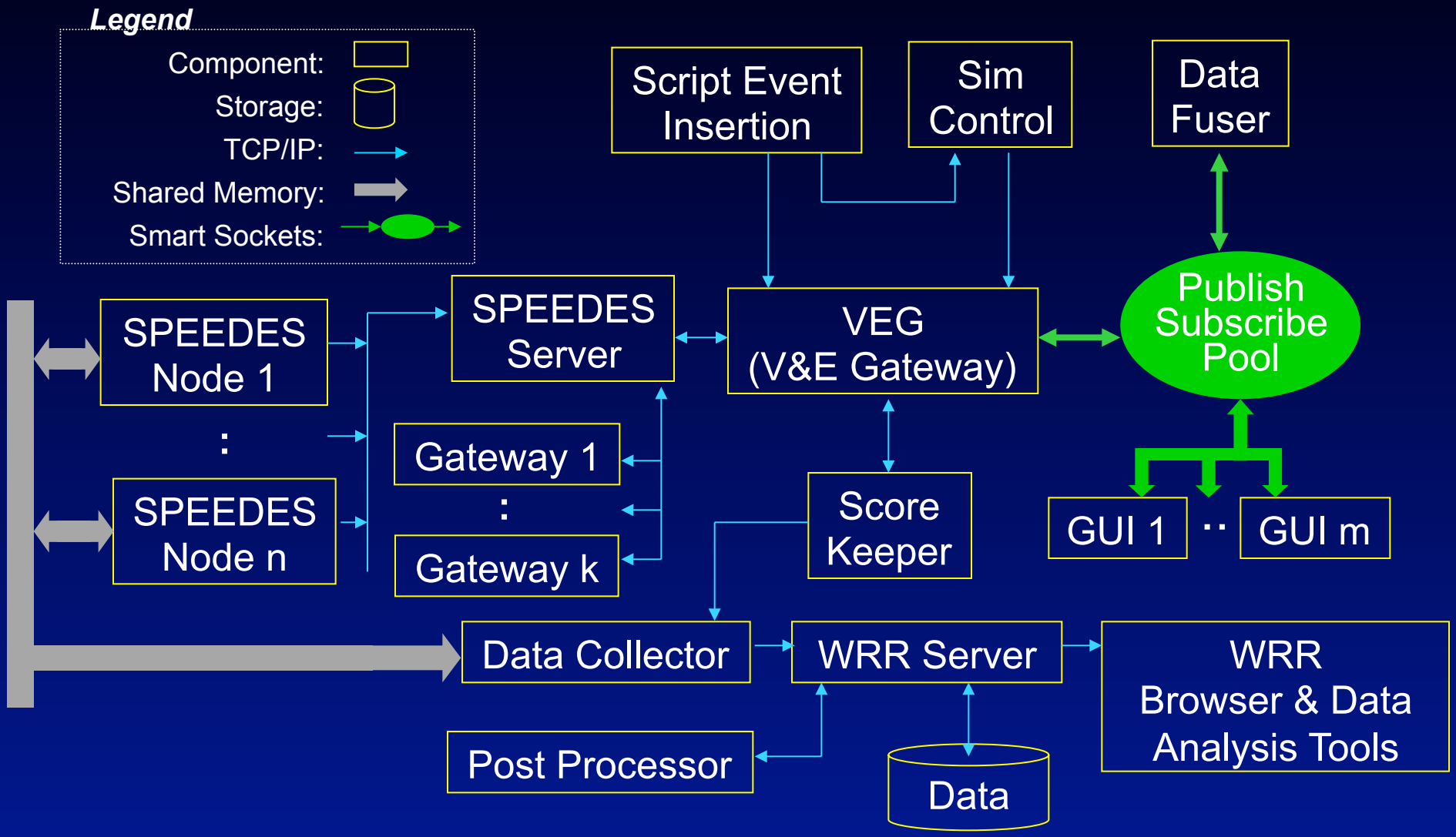
Reuse: *Code reuse itself has not helped at all. Design and knowledge reuse has proven helpful and should be maximized as much as possible.*

Fidelity: *Ability to accurately model the details of the objects that give the entities in the simulation some level of realism. This includes modeling the "fog of war," the uncertainty factor that needs to be part of a realistic simulation.*

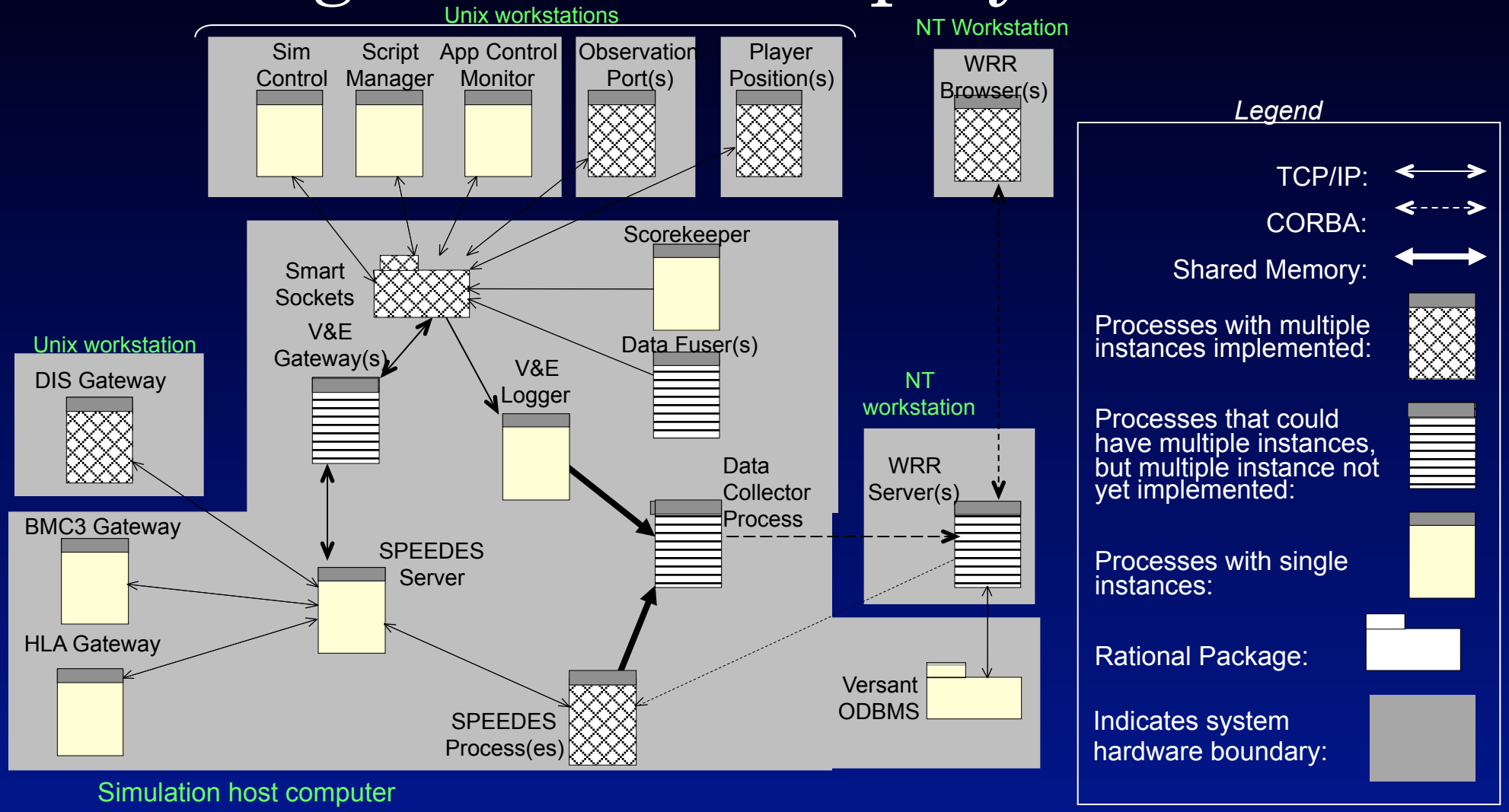
Wargame 2000 System Context



MADSIM System Architecture



Wargame 2000 Deployment View



Architectural Approaches (Styles)

Parallel Discrete Event Simulation was the overall strategy to generate events for the simulation.

Client-Server:

- Used to decouple the event generator from the rest of the system. This offered flexibility for upgrading to a more capable event generator.
- Used between the user interface (UI) and the rest of the system, offering more flexibility in changing/upgrading the UI.
- there was also distributed access to the data repository.

A publish/subscribe:

- Used throughout the system as an approach to decouple various components within the MADSIM subsystem.

Architectural Approaches (Styles)

Event-based message passing

- used as an approach to indicate that various simulation events had occurred

Shared memory

- Used to achieve high levels of performance

Wrapping/Facades

- mitigate dependence on some commercial-off-the-shelf (COTS) products

Object Request Broker (ORB)

- further decouple and mediate client-to-server interfaces

High Priority Scenarios

There were 8 high priority scenarios; some sample high priority scenarios:

- *Provide the ability to create parameter files (configuration files) allowing users to change the characteristics of a sensor or other simulation entity (such as radar aperture or loop gain of a radar) using one graphical interface.*
- *Provide capable to analyze system performance during periods of heavy data throughput (objects, tracks), and loads to the graphical user interface (GUI).*
- *Provide virtual gaming that includes online, offsite, and multiple games running simultaneously and allows participants to play from their own operational command centers.*

Example Analysis of Scenarios

Parameter files were created to allow generic objects in the simulation to be instantiated into specific objects during system initialization.

- Non-risk: Decision permitted maximum flexibility in configuring simulations – *thus promoting modifiability*

Decision to use text-based parameter files to configure the system.

- Non-risk: use of text files promotes fast system initialization time - *thus promoting performance*
- Risk: there is no capability to configure parameter files easily and consistently – thus impacted business goal to meet "exercise turnaround time"
- Implicit tradeoff was made between initialization speed and configuration complexity

Example Analysis of Scenarios

In order to facilitate interoperability with other simulations, compliance with the High Level Architecture (HLA) was mandated.

HLA compliance in Wargame 2000 is achieved via an HLA gateways to interface with other systems that are HLA compliant.

- Risk: The system architect expressed concerns about the HLA real-time interface and indicated that it would not be able to meet the user's performance expectations.

Sensitivity points

Performance is sensitive to careful adherence to architectural constraints and implementation guidelines.

Performance was an issue of great concern for the system architect.

- Goal was to have enough capacity and reserve capacity for future requirements

The ATAM exercise showed that the performance was impacted due to violation of architectural constraints

- Attributed to weak architectural documentation
  - Initializing simulated sensors all at the same time during system start-up
  - Modifiability sacrificed when architectural connectors by-passed

Sensitivity points

Increasing reliability comes at a cost of performance.

Approaches included: added displays, and “heart-beat monitoring”

While easy to implement, they required committing resources, particularly in network usage, thereby affecting performance.

- The impact of requesting more reliability via monitors was not obvious all stakeholders until the ramifications were illuminated through the ATAM exercise.
- This is a case where the ATAM exercise broadened communication among diverse stakeholders.