SATURN 2018

14th Software Engineering Institute Architecture Technology User Network Conference

Boot Camp: Architecture Evaluation

Ipek Ozkaya
ozkaya@sei.cmu.edu

John Klein
jklein@sei.cmu.edu

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM18-0601

Boot Camp Objectives

This session will familiarize participants with

- why we evaluate architectures
- the benefits of evaluating architectures
- the cost of evaluating architectures
- when to evaluate architectures
- the SEI Architecture Tradeoff Analysis Method® (ATAM®)

® Architecture Tradeoff Analysis Method and ATAM are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

Check Your Understanding: Critique A Design

Consider a company that builds hardware-based field systems for controlling building functions such as heating, ventilation, air conditioning, access, and safety.

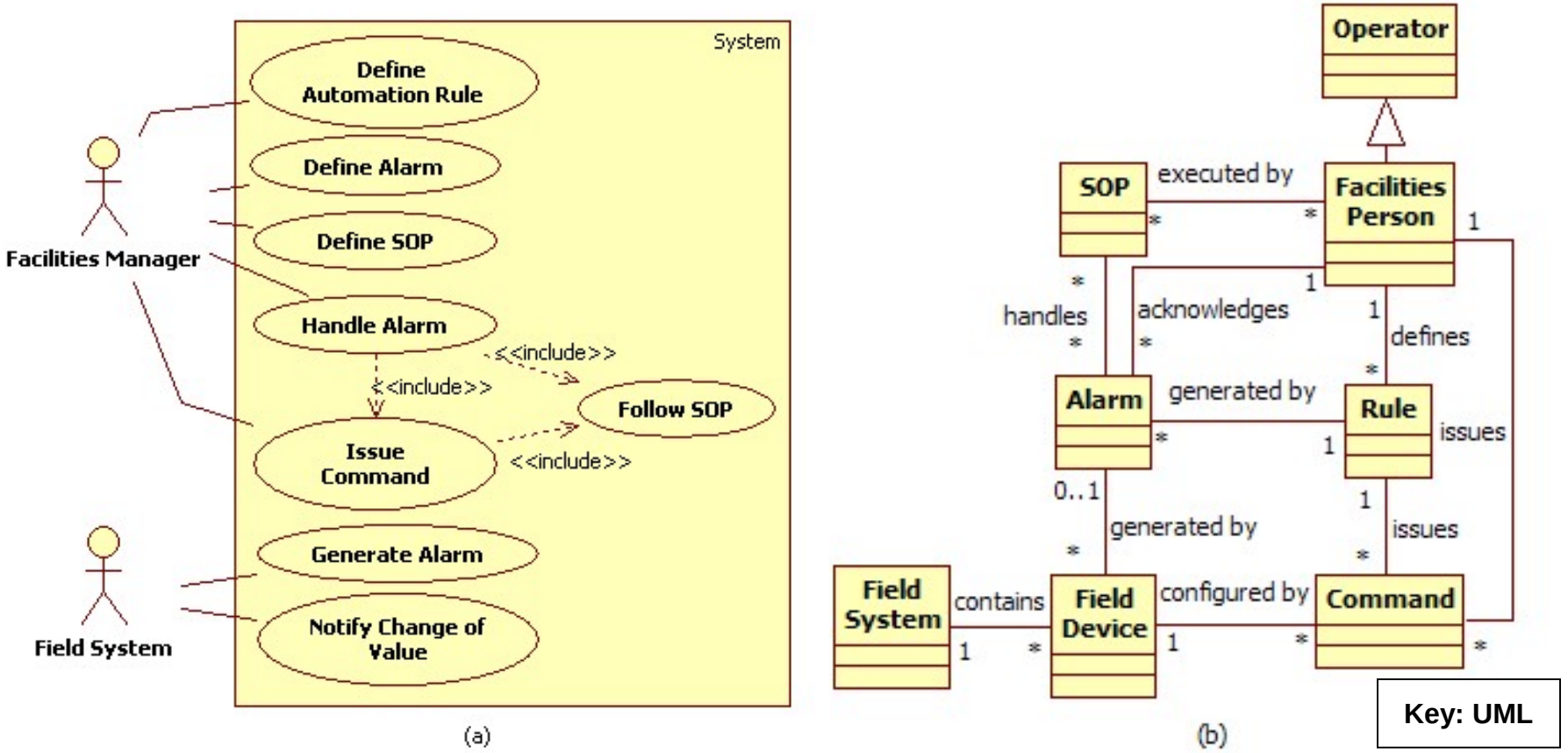
The company wants to develop a software-based system for facilities managers. The system would perform these functions:

- manage a network of hardware-based field systems
- issue commands to configure the field systems
- define rules based on property values that trigger commands
- for life-critical situations, trigger alarms notifying appropriate users

The company wants to offer this product in new markets, and expand its sales by allowing value-added resellers to sell the system under their own brands, supporting field systems from manufacturers of their choice.

Check Your Understanding: Design - 1

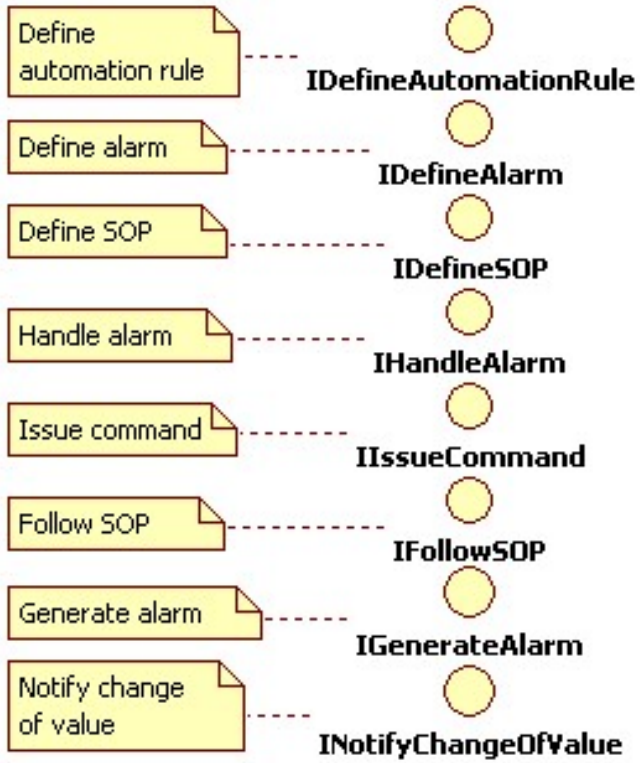
Object-oriented analysis and design artifacts showing
(a) use cases (b) business domain model[1]



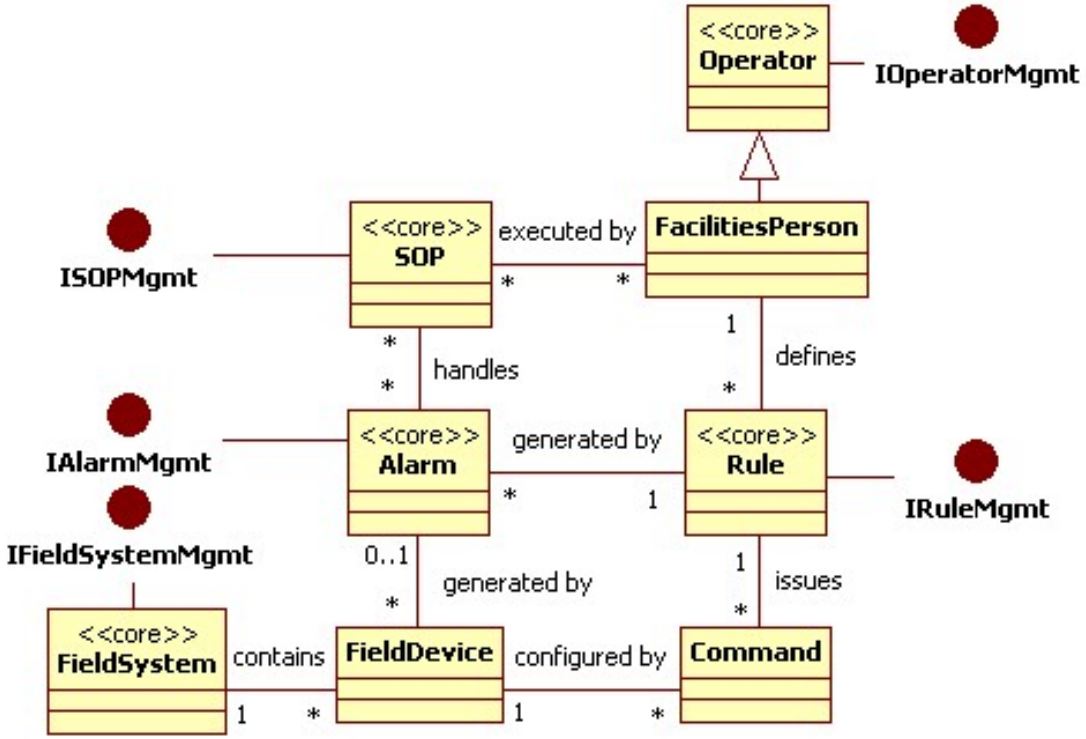
[1] Ipek Ozkaya, Len Bass, Raghvinder Sangwan and Robert Nord. "Making Practical Use of Quality Attribute Information." IEEE Software special issue on Software Quality Requirements, March/April 2008.

Check Your Understanding: Design - 2

Object-oriented analysis and design artifacts showing
(c) system interfaces (d) business interfaces



(c)

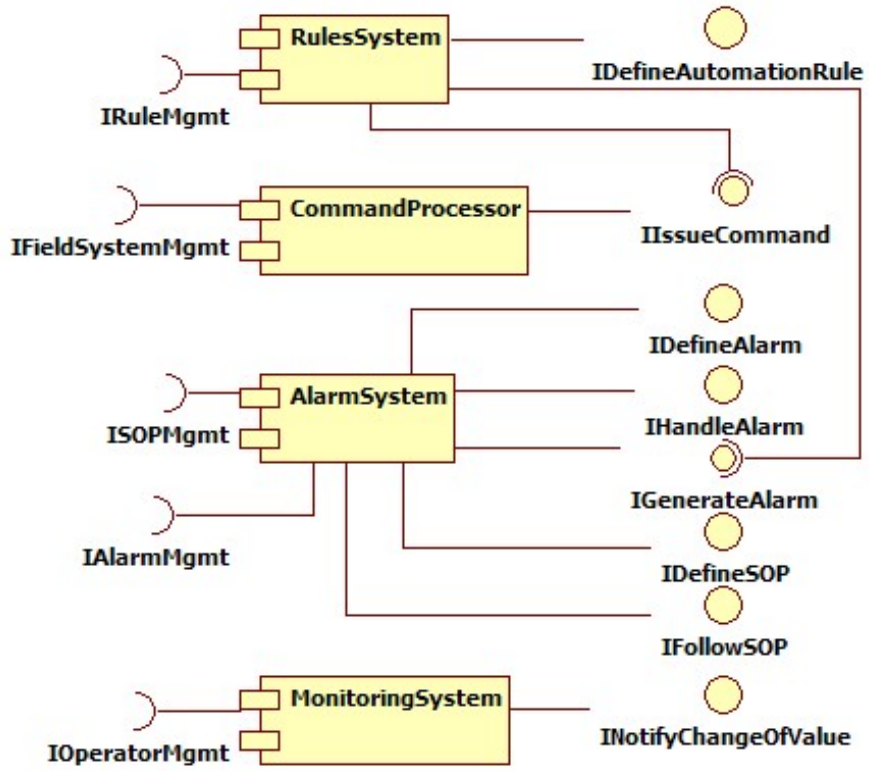


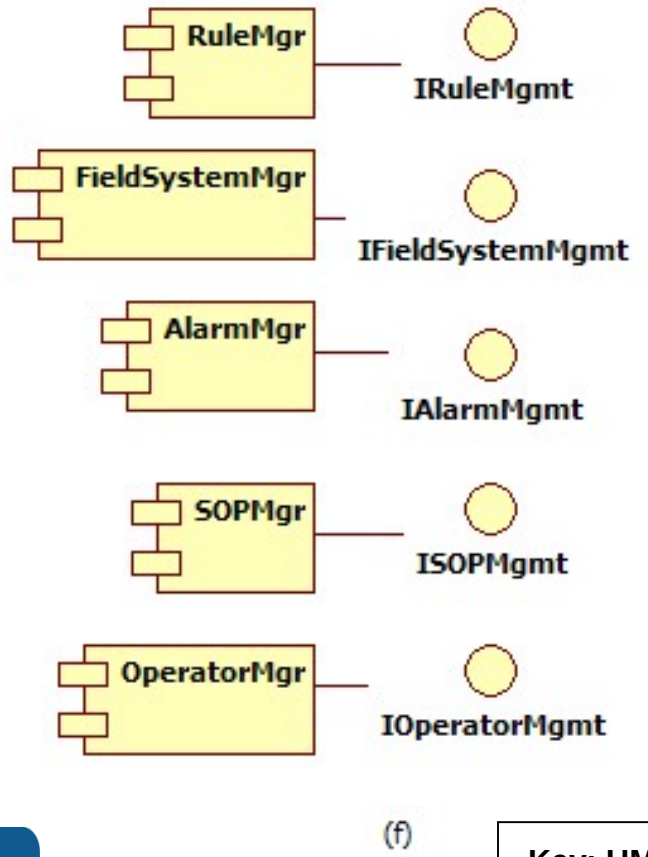
(d)

Key: UML

Check Your Understanding: Design - 3

Object-oriented analysis and design artifacts showing
(e) system components (f) business components

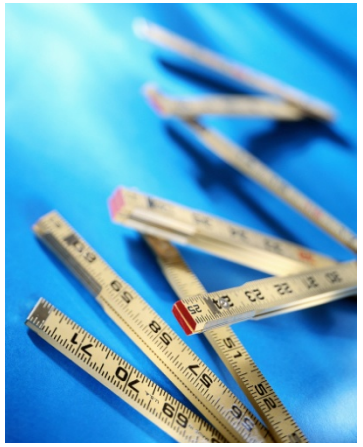




Key: UML

What do you think of this design?

Things to Establish for Architecture Evaluation

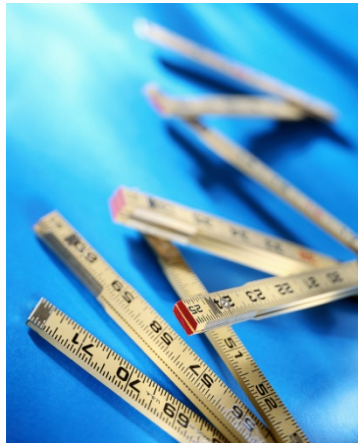


An objective measure

An understanding of the architecture

The analysis

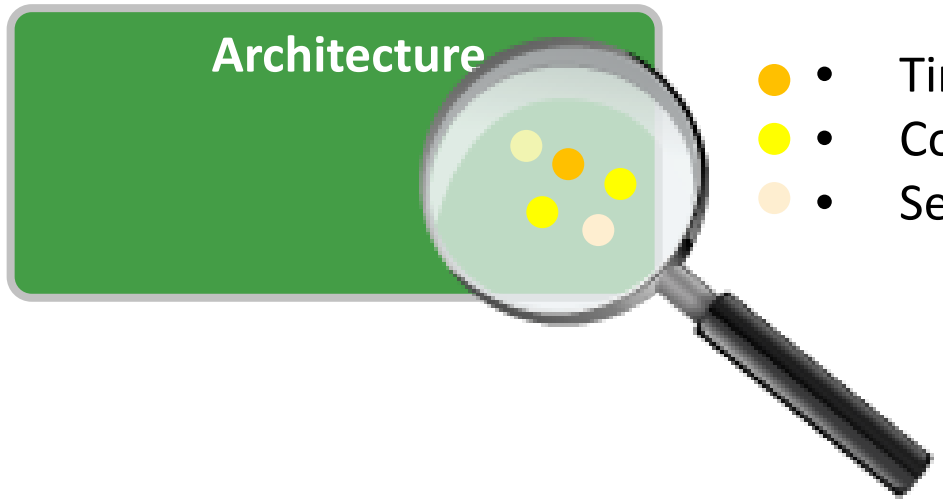
An Objective Measure



Building the Yardstick

Without a yardstick to measure, every architecture is good or bad.

Will the designed system solution have the properties to make the organization successful?



- Timing
- Cost of change
- Sensitive data

System properties of interest here are the quality attribute properties of the functions the system has.

Stakeholders

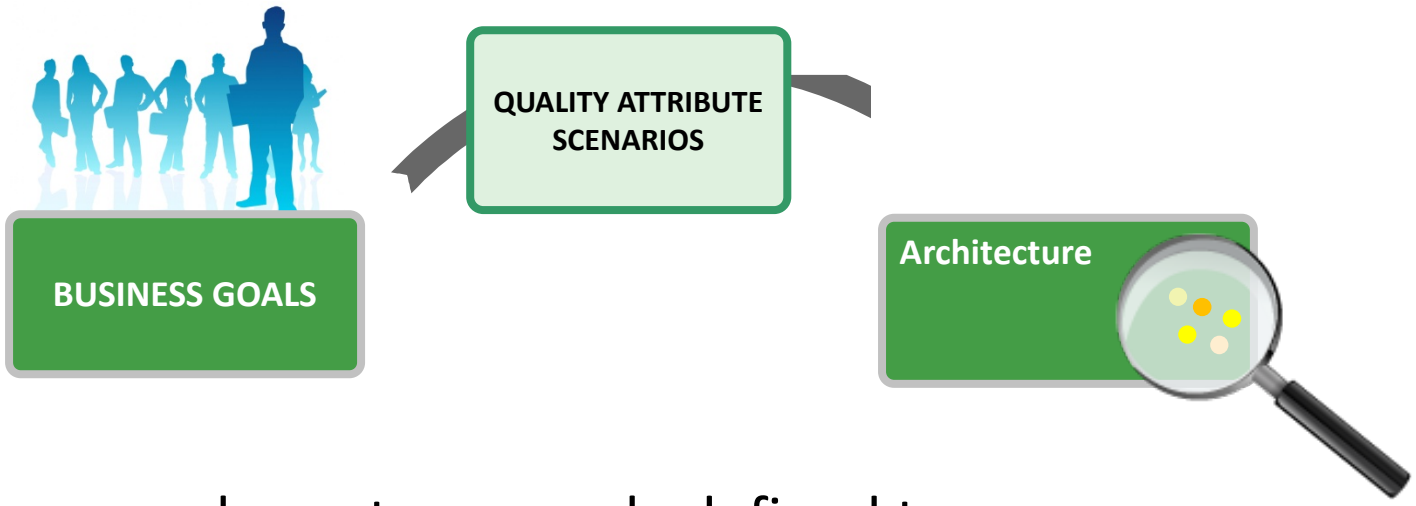
BUSINESS GOALS

Only the stakeholders of an organization can specify what the expected important quality attributes are.

- We need to have a system that can easily be adapted to new market conditions.
- Our systems cannot have any defects in the field
- We guarantee 24/7 availability

These statements (business goals) describe how the organization plan to be successful with their business.

Quality Attribute Scenarios

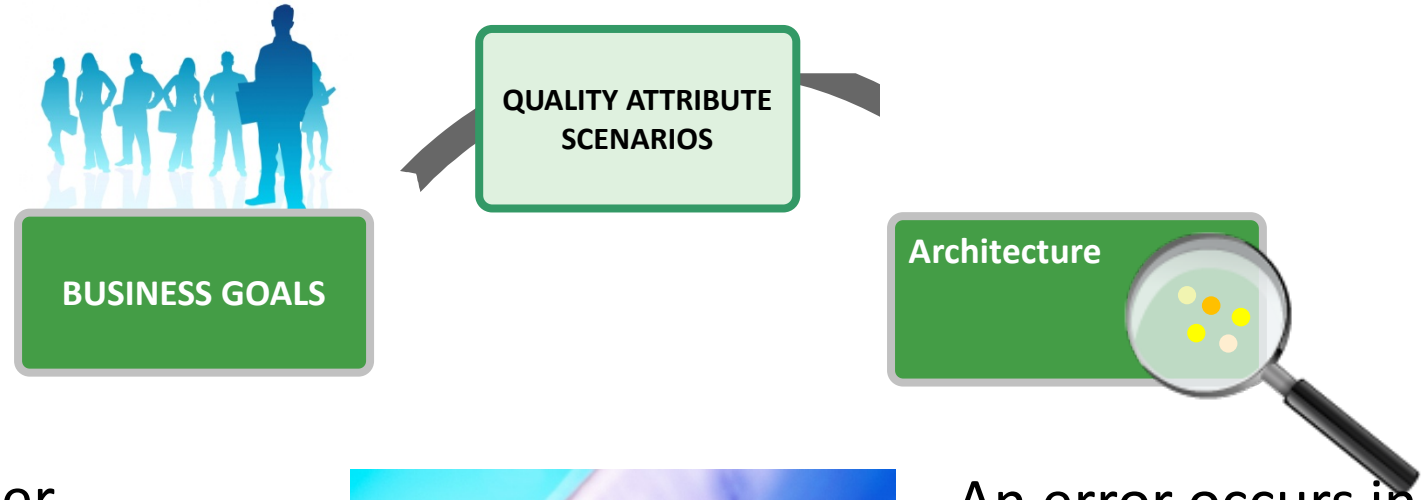


In most cases business goals are too vaguely defined to actually be able to measure their achievement.

Quality attribute scenarios bridge business goals and architecture quality attribute properties.

They describe what quality attribute properties the system is expected to possess to run a successful business.

The Yardstick



A customer requires a new auction algorithm. A developer implements and integrates that function within one day of effort.

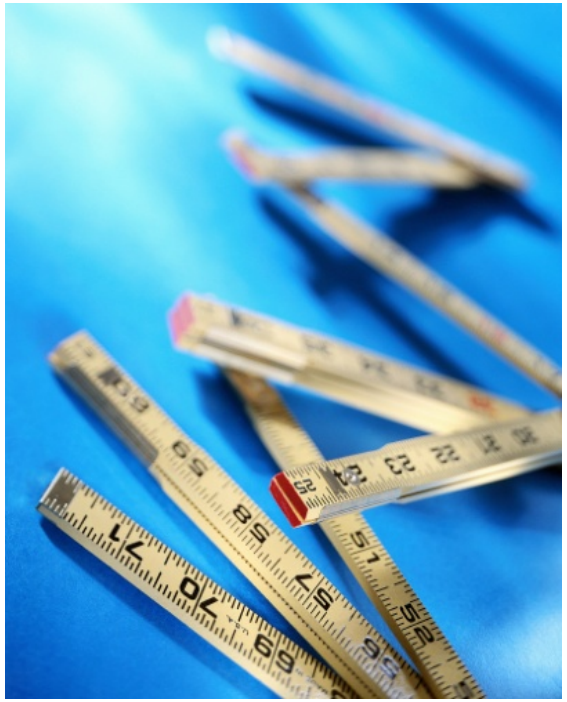
An error occurs in a fielded system. The system recovers from the error without manual intervention within 5 seconds.

The Yardstick – Summary

Principle 1: Important quality attribute properties of the architecture need to be evaluated.

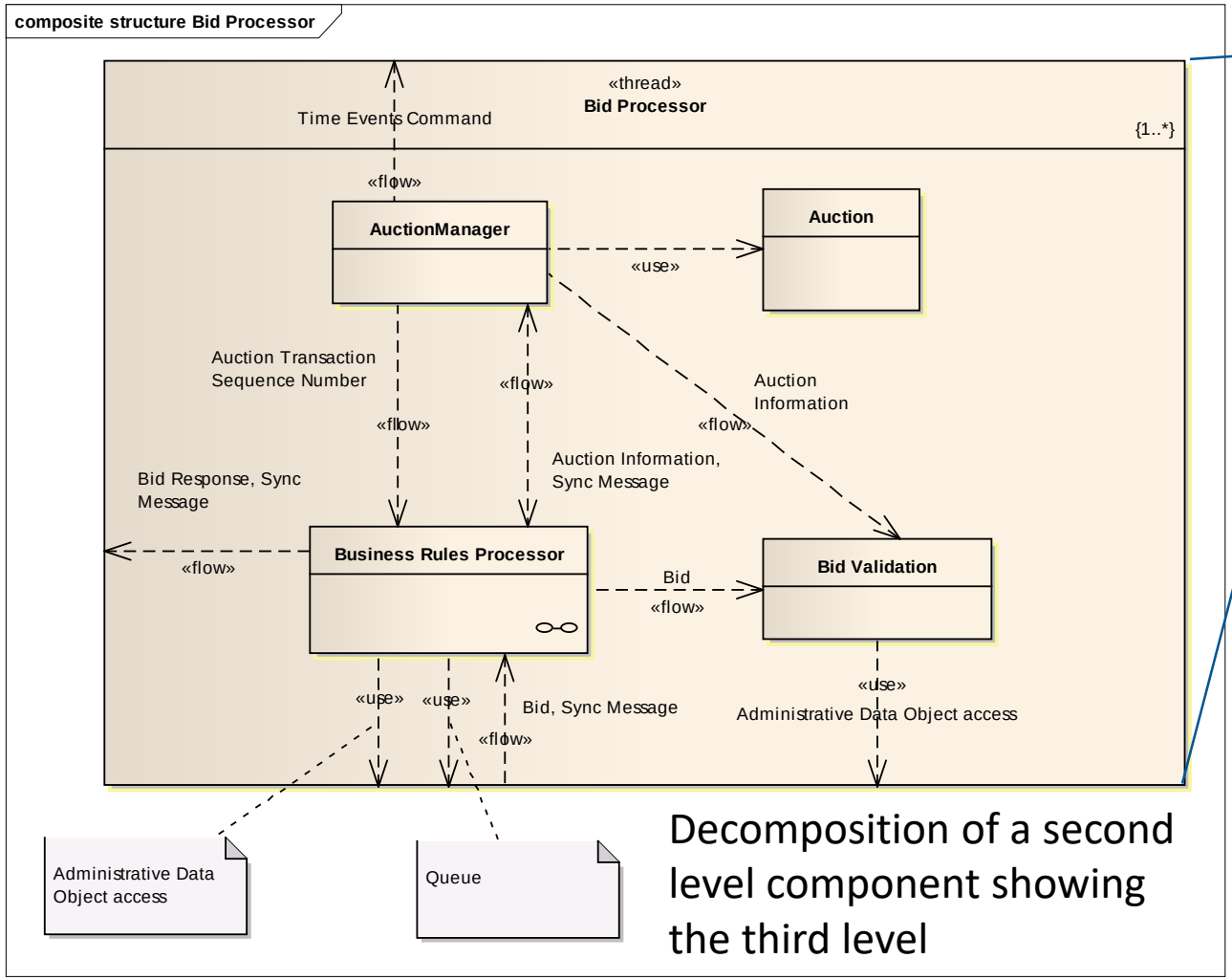
Principle 2: What is important is derived from the business goals.

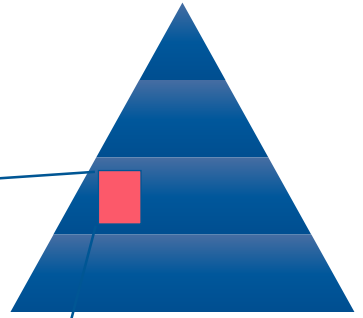
Principle 3: Quality attribute scenarios translate business goals into requires quality attribute properties.



An Understanding of the Architecture

Example Component Diagram





A typical piece of architecture documentation

What are the properties here?

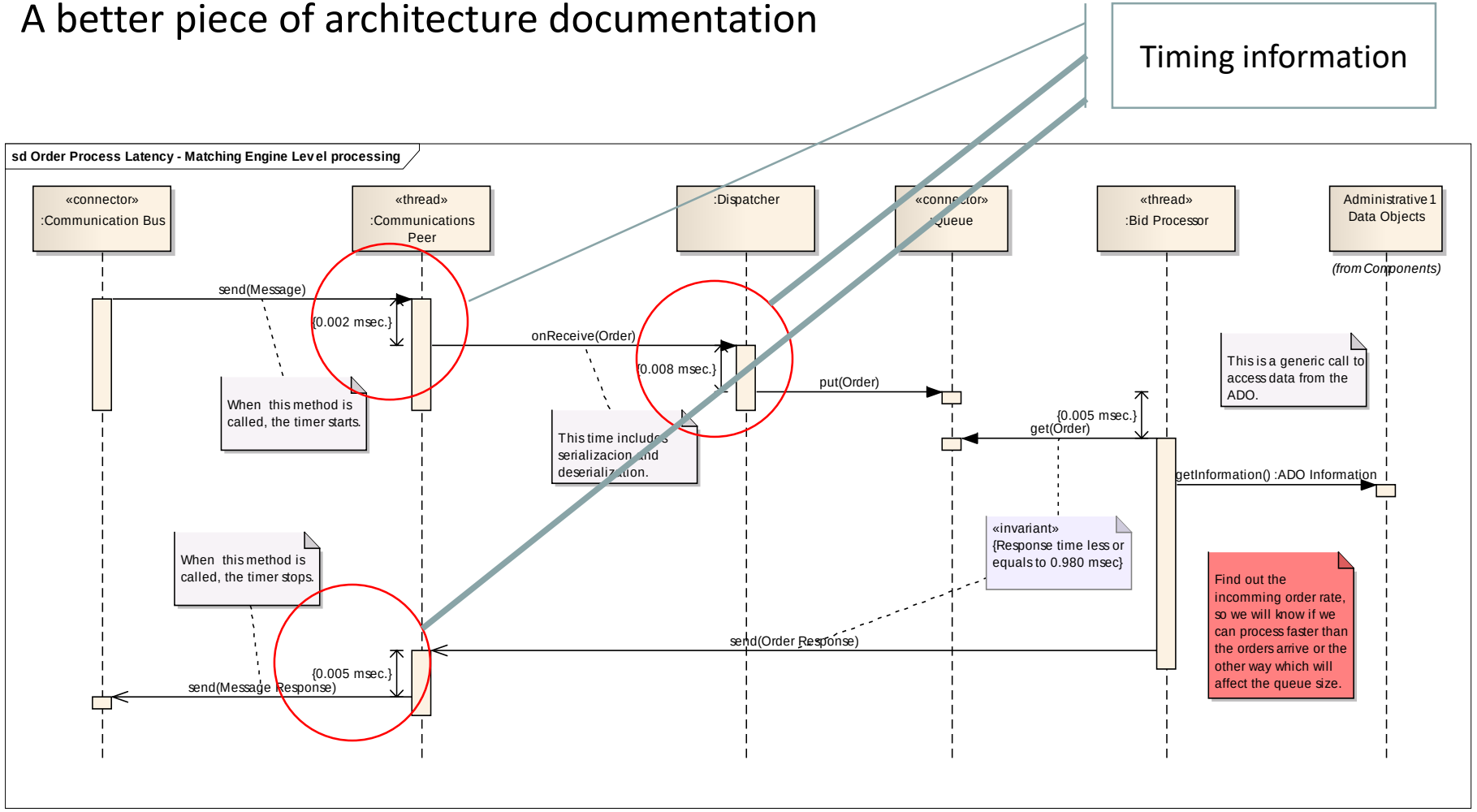
Where are they?

What can we say about:

- Performance?
- Modifiability?
- Security?

Sequence Diagram with Timing

A better piece of architecture documentation



Architecture Approaches

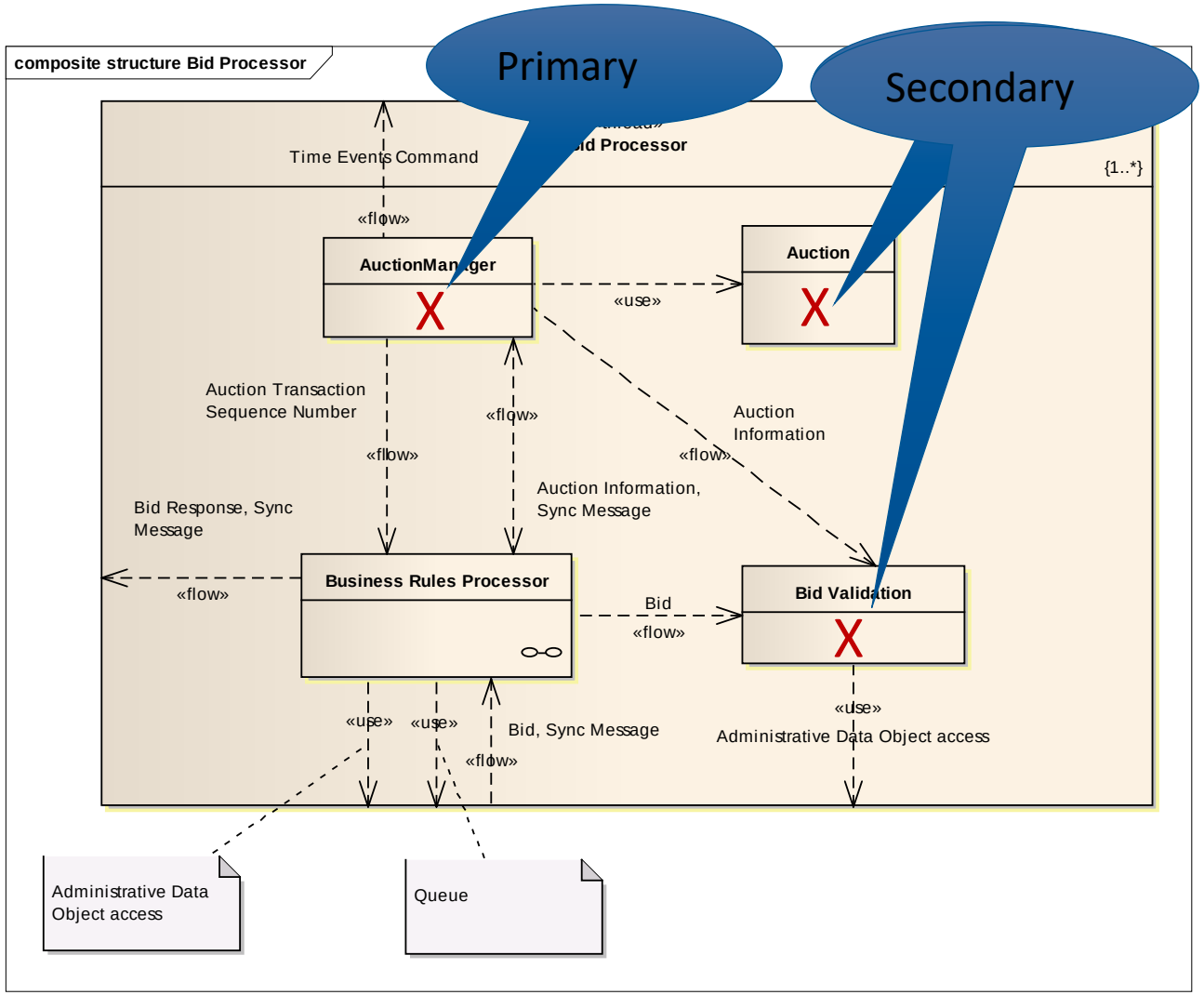
Narrow down the problem by focusing on a quality attribute scenario:

> A customer requires a new auction algorithm. A developer implements and integrates that function within one day of effort.

First question to ask:

> What are the components involved in this scenario

Example Component Diagram



Architecture Approaches

Second question to ask:

What are concepts put in place to make this change easy?

- Localized everything that needs to be changed in one component
- Data structures have a generic interface that allows to hide internal changes
- Component also has a versioned interface to allow backward compatibility

These are architecture approaches to support this extensibility scenario.

Architect vs. Documentation

An architecture documentation that can be used for evaluation needs to show where those concepts are and what their properties are:

- Adding a new algorithm means specializing the generic class “AuctionManager”. A typical algorithm can be implemented in four hours.
- New data structures can be added, but existing data structures cannot be changed without any impact on existing functions. Very seldom new structures are required. If so, they can be added within one hour.
- Every component working with the AuctionManager requests an interface of a specific version. Creating a new version of an interface will take about one day.

If this information is not in the architecture document then the answers have to come from the architect.

Side Effects

Every architecture approach used has also negative impact on other quality attributes:

- Putting everything that needs to change in one place may introduce unnecessary dependencies to other components. Bad for security and other types of changes
- Data structures with generic interfaces may impose a performance penalty
- Versioned interfaces increase complexity, which is more difficult to test and the change for system crashes increases

The architect needs to be aware of these issues, needs to put mitigations into place, and document them.

Understanding the Architecture – Summary

Principle 4: Identify relevant components by using scenarios.

Principle 5: Identify architecture approaches with their quality attribute properties

Principle 6: Identify the side effects of those architecture approaches.

The Analysis

Matchmaking

The architecture evaluation process has to answer the questions:

- Do the quality attribute properties of the identified components support the given quality attribute scenario sufficiently?
- Is the negative impact of the chosen architecture approaches on other quality attribute scenarios acceptable?

If the answers are documented, then the examining the documentation is sufficient for the evaluation.

If not, the architect has to provide the answers in an interview.

The evidence provided must be sufficient to convince quality attribute experts and the stakeholders.

Risks

No system design is without any risks.

Successful organizations understand and handle the risks.

An architectural risk is when

- The system properties do not completely satisfy the given quality attribute scenarios.
- The approaches used for one scenario negatively impact other scenarios.

If an important scenario cannot be supported, then one or more business goals are impacted.

Analysis – Summary

Principle 7: The architect provides evidence that the system's quality attribute properties will indeed fulfill the scenarios.

Principle 8: Mismatches between architecture properties and scenarios become risks to the business goals.

The ATAM

The SEI developed the Architecture Tradeoff Analysis Method (ATAM) and has applied it to architectures for systems of wide-ranging sizes and domains.

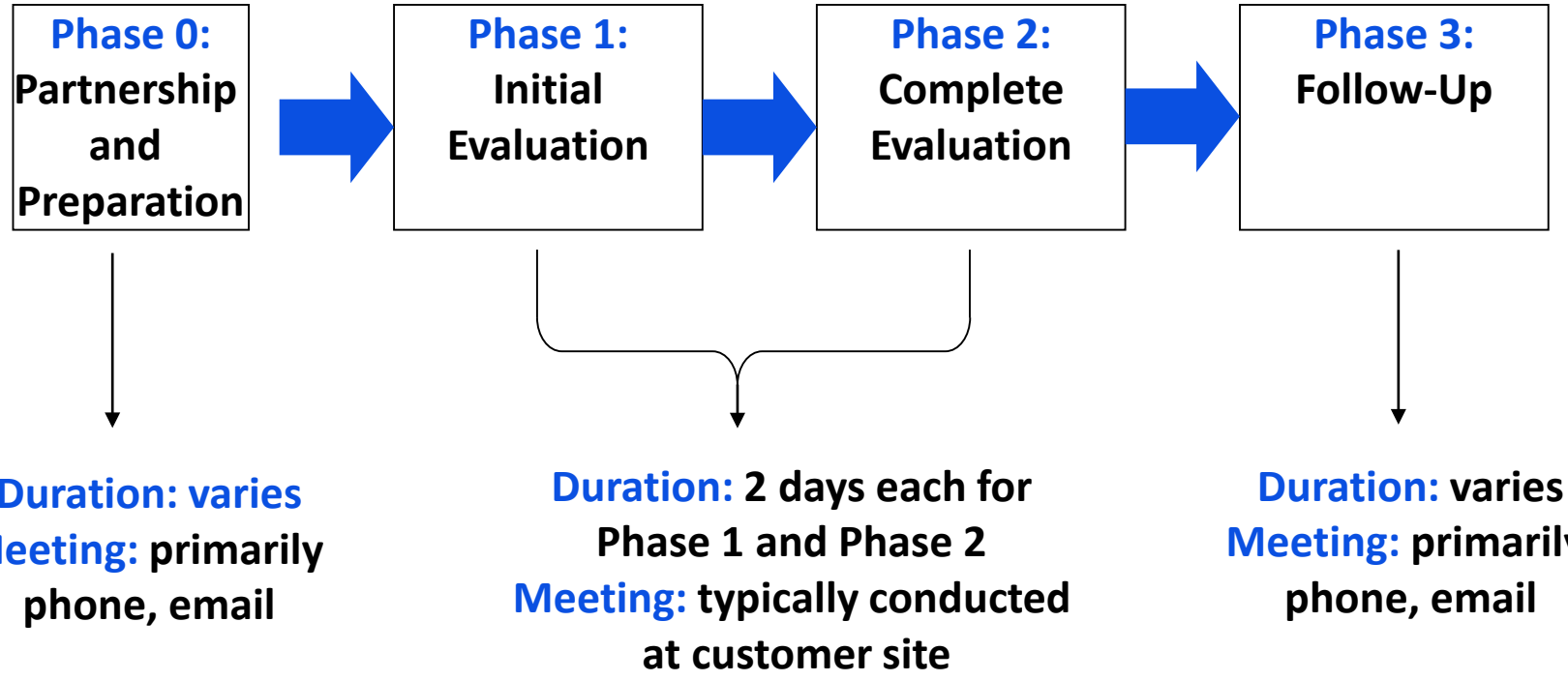
The purpose of the ATAM is to assess the consequences of architectural decisions in light of quality attribute requirements and business goals.

The ATAM brings together three groups in an evaluation:

1. a trained evaluation team
2. an architecture's "decision makers" (architect, senior designers, project managers, customers)
3. representatives of the architecture's stakeholders

ATAM Phases

ATAM evaluations are conducted in four phases.



ATAM Phase 1

Phase 1 involves a small group of predominantly technically oriented stakeholders.

Phase 1 is

- architecture-centric
- focused on eliciting detailed architectural information and analyzing it
- a top-down analysis

ATAM Phase 1 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.

Phase 1

7. Brainstorm and prioritize scenarios.
8. Analyze architectural approaches.
9. Present results.

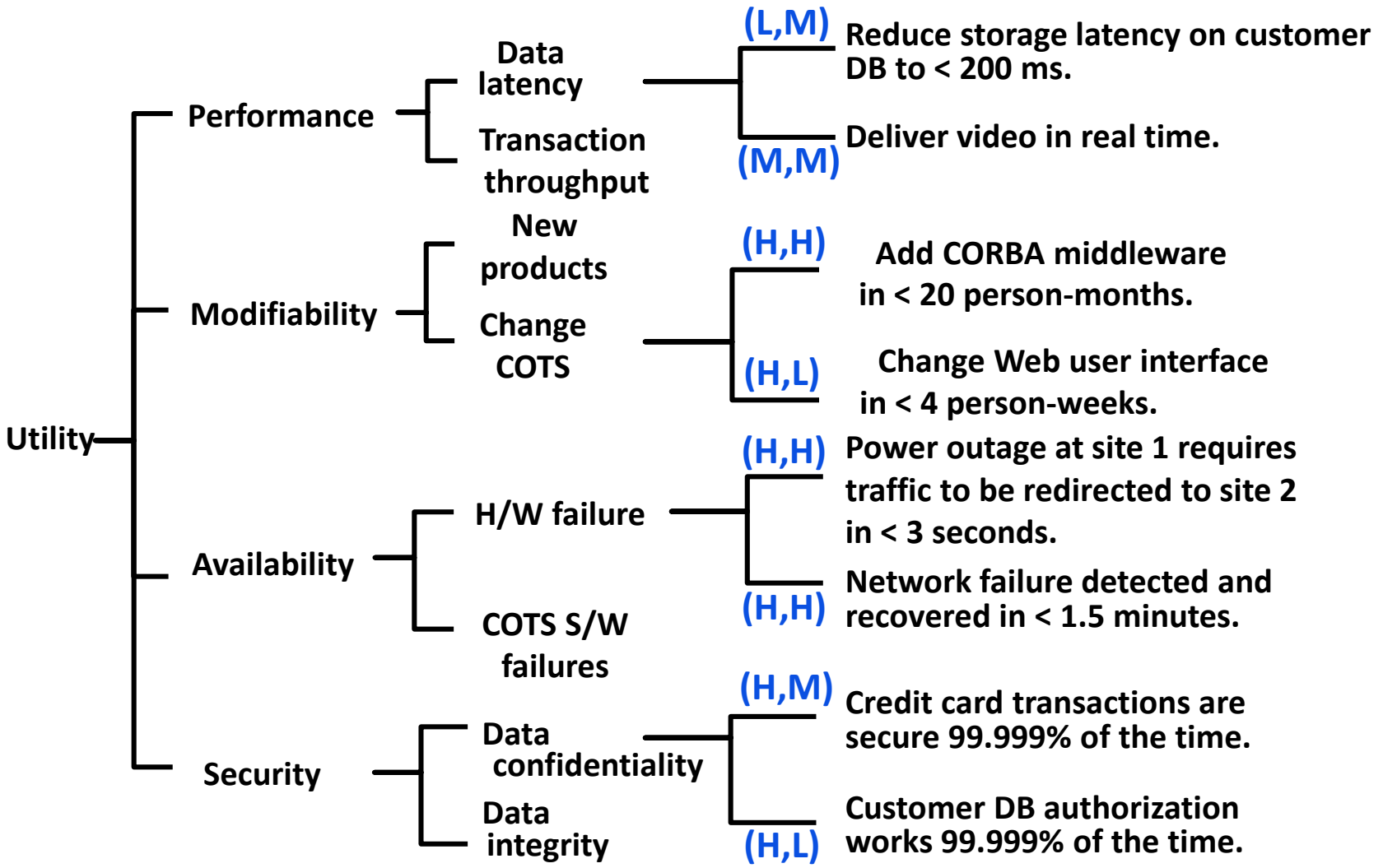
What Are Quality Attribute Utility Trees?

You can 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.
- The highest level nodes are typically quality attributes such as performance, modifiability, security, availability, and so forth.
- Scenarios are the leaves of the utility tree.

The utility tree is a characterization and a prioritization of specific quality attribute requirements.

Example of Quality Attribute Utility Tree



L = Low, M = Medium, H = High

How Scenarios Are Used – 1

For design purposes, we use six-part scenarios as described earlier:

1. **source –** an entity that generates a stimulus
2. **stimulus –** a condition that affects the system
3. **artifact(s) –** the part of the system that was stimulated by the stimulus
4. **environment –** the condition under which the stimulus occurred
5. **response –** the activity that results because of the stimulus
6. **response measure –** the measure by which the system's response will be evaluated

How Scenarios Are Used – 2

Scenarios are used to

- represent stakeholders' interests
- understand quality attribute requirements

Scenarios should cover a range of

- anticipated uses of the system (use case scenarios)
- anticipated changes to the system (growth scenarios)
- unanticipated stresses on the system (exploratory scenarios)

Scenarios are linked to business goals, for traceability.

A good scenario clearly states the stimulus and the responses of interest.

Examples of Scenarios

Use case scenario

- A remote user requests a database report via the Web during a peak period and receives it within 5 seconds.

Growth scenario

- During maintenance, add an additional data server within 1 person-week.

Exploratory scenario

- Half of the servers go down during normal operation without affecting the overall system availability.

Scenarios should be as specific as possible.

Stimuli, Environment, Responses

Use case scenario

- The remote user requests a database report via the Web during a peak period and receives it within 5 seconds.

Growth scenario

- During maintenance, add an additional new data server within 1 person-week.

Exploratory scenario

- Half of the servers go down during normal operation without affecting the overall system availability.

Scenario Analysis Outputs

As each scenario is analyzed against the architecture, the evaluation team identifies risks.

A risk is a potentially problematic architectural decision.

> *"Rules for writing business logic modules in the second tier of your three-tier architecture are not articulated clearly. This could result in the replication of functionality, thereby compromising the modifiability of the third tier."*

ATAM Phase 2

Phase 2 involves a larger group of stakeholders.

Phase 2 is

- stakeholder-centric
- focused on eliciting diverse stakeholders' points of view and on verifying the results of Phase 1
- bottom-up analysis

ATAM Phase 2 Steps

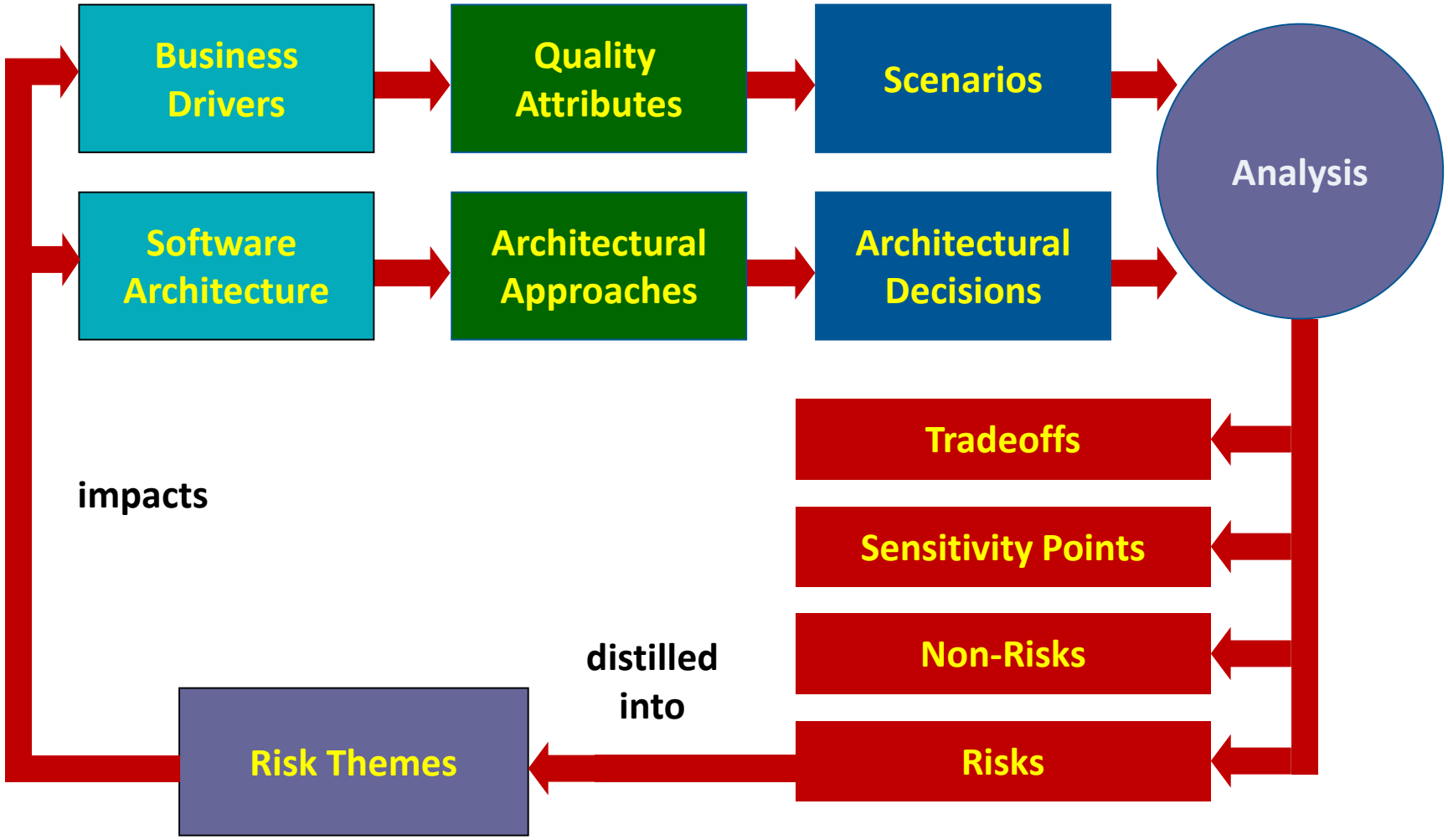
1. Present the ATAM.
2. Present business drivers.
3. Present architecture.
4. Identify architectural approaches.
5. Generate quality attribute utility tree.
6. Analyze architectural approaches.
7. Brainstorm and prioritize scenarios.
8. Analyze architectural approaches.
9. Present results.

Recap Phase 1

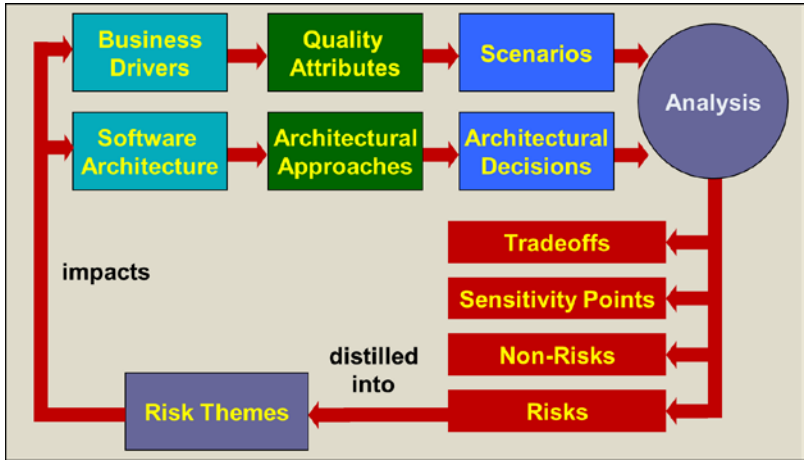
Do this

Phase 2

Conceptual Flow of the ATAM



Principles Apply to ATAM



Principle 1: Important quality attribute properties of the architecture need to be evaluated.

Principle 2: What is important is derived from the business goals.

Principle 3: Quality attribute scenarios translate business goals into requires quality attribute properties.

Principle 4: Identify relevant components by using scenarios.

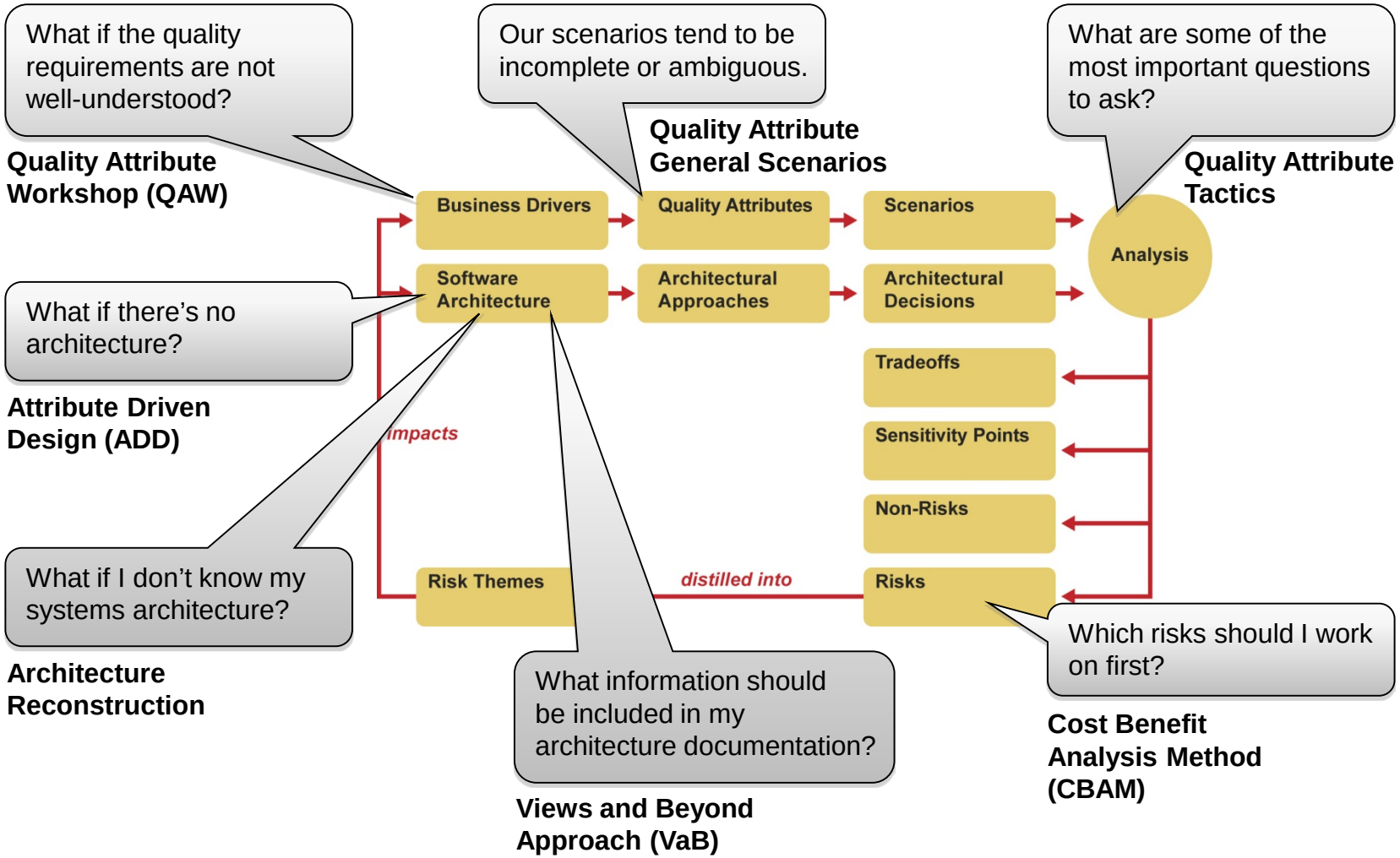
Principle 5: Identify architecture approaches with their quality attribute properties

Principle 6: Identify the side effects of those architecture approaches.

Principle 7: The architect provides evidence that the system's quality attribute properties will indeed fulfill the scenarios.

Principle 8: Mismatches between architecture properties and scenarios become risks to the business goals.

ATAM Can be Used in Conjunction with Other Methods and Techniques



Summary

Architecture evaluation

- why – to understand and analyze design tradeoffs
- when – throughout the lifecycle
- benefits – overall cost reduction, better understanding and prioritization of requirements and quality goals, early detection of problems, improved architectures

Evaluations are carried out via questioning and measuring techniques.

The ATAM is the most widely used method for evaluating architectures with respect to multiple quality attributes.

More Information About Architecture Evaluation

1. Clements, P.; Kazman, R.; & Klein, M. *Evaluating Software Architectures: Methods and Case Studies*. Addison-Wesley, 2002.

2. Architecture Tradeoff Analysis Method (ATAM) Case Studies. Software Engineering Institute, Carnegie Mellon University , 2000-2003. http://www.sei.cmu.edu/architecture/start/publications/atam.cfm

3. Architecture Tradeoff Analysis Method. http://www.sei.cmu.edu/architecture/tools/evaluate/atam.cfm