Software Architecture Thoughts for the System Security Design

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

James Ivers
April 17, 2007

Role of 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, like

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

Quality attributes such as these are largely dependent on architectural decisions.

- All design involves tradeoffs among quality attributes.
- The earlier we reason about tradeoffs, the better.

Key Topics in Creating a Software Architecture

- Scoping the problem
- Defining/refining the architecture
- Documenting the architecture
- Evaluating the architecture

Scoping the Problem

What is being defined in the architecture?

- New features/assets
- Integration between new features/assets and existing systems
- Recommendations for existing features/assets

What constraints are we under?

- Business (e.g., deadlines, cost, or regulatory standards)
- Technical (e.g., existing assets or interfaces)

What are the driving quality attributes?

- Security, modifiability, reliability, performance, usability, etc.
- How do we manage the trade-offs among qualities

How will the architecture be used?

- Basis for implementation
- Detailed analyses
- Contract between component suppliers and acquirers

Key Topics in Creating a Software Architecture

Scoping the problem

➡ Defining/refining the architecture

Documenting the architecture

Evaluating the architecture

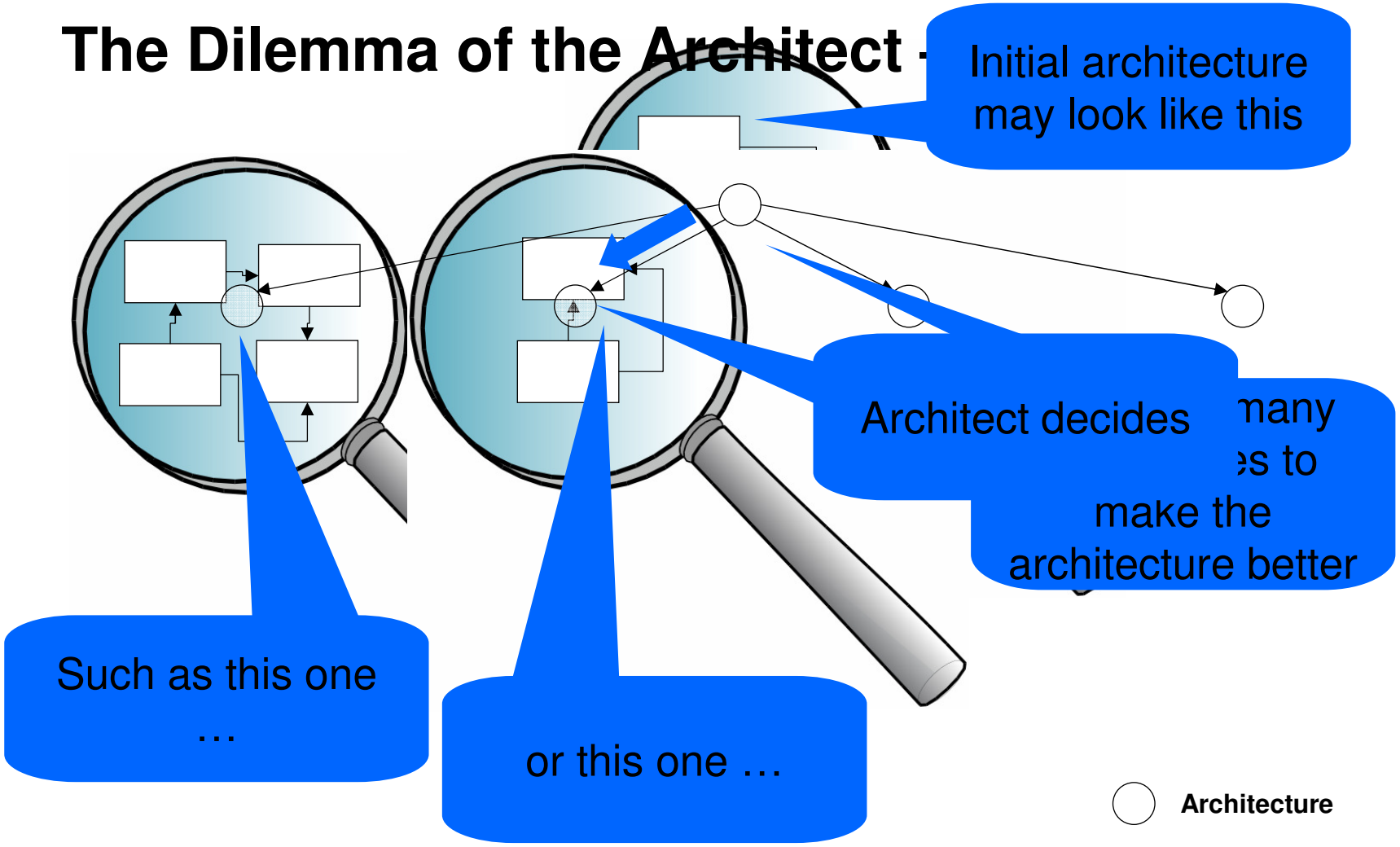
The Architecture Design Process

An architecture design follows (should, really!) this process:

1. Create a measurable specification of quality attribute requirements that need to be supported by the architecture
2. Evaluate if the current architecture you have fulfills those requirements
3. If not, make some changes to the architecture to improve and repeat step 2
4. If yes, Lucky you! You are done.

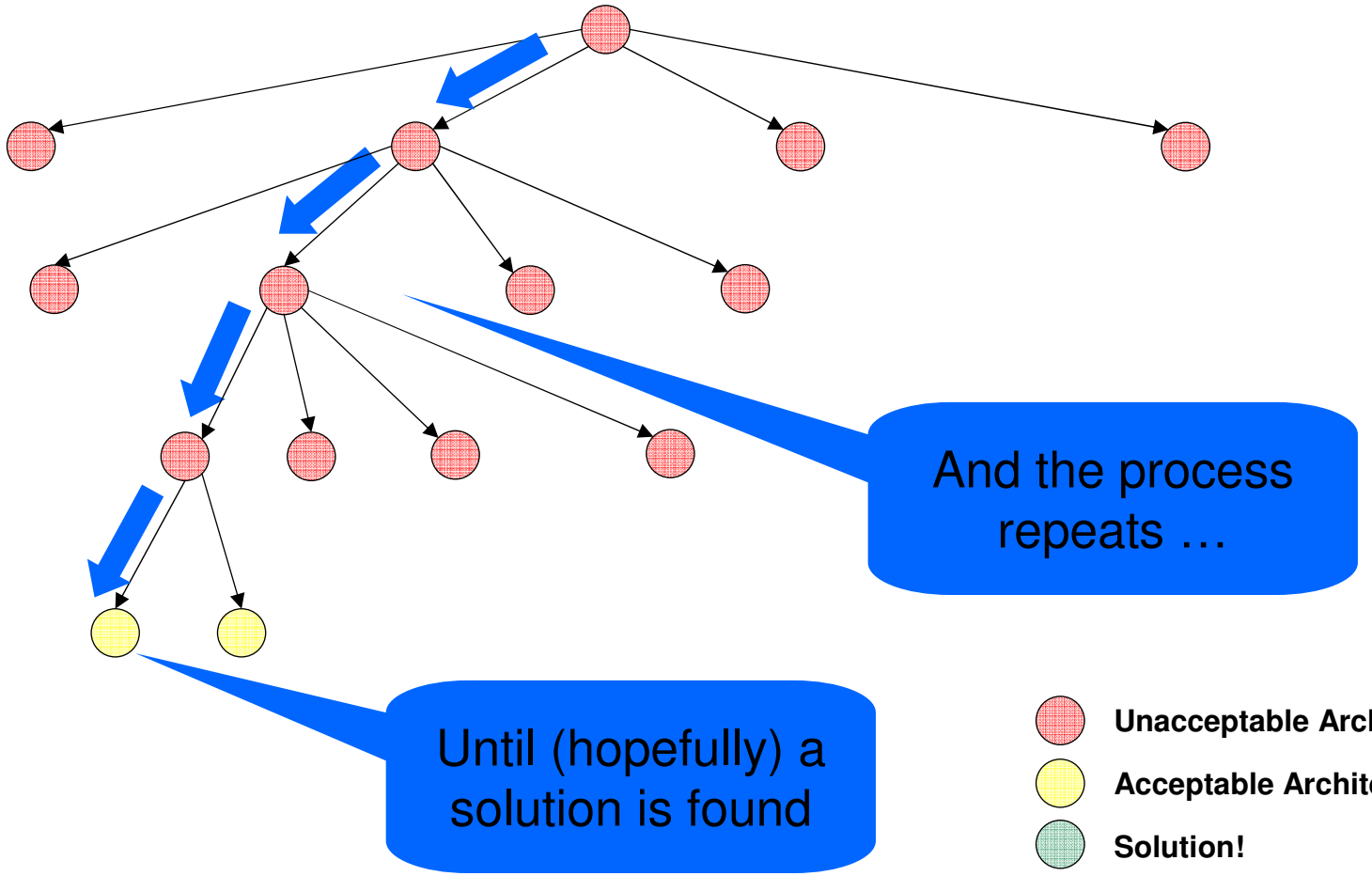
As simple as this may sound, it creates a huge problem …

The Dilemma of the Architect -



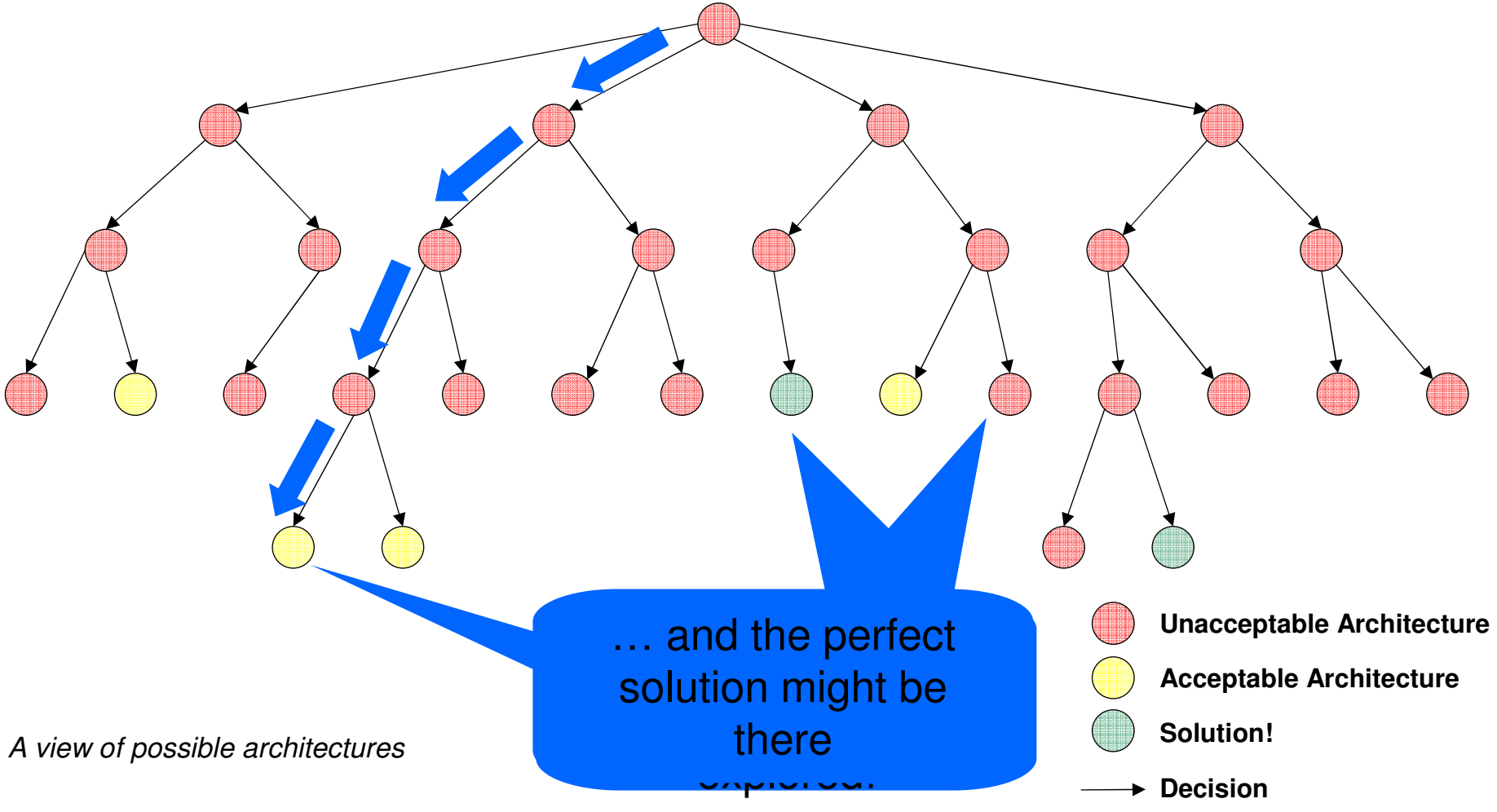
A view of possible architectures

The Dilemma of the Architect – 2



A view of possible architectures

The Dilemma of the Architect – 3

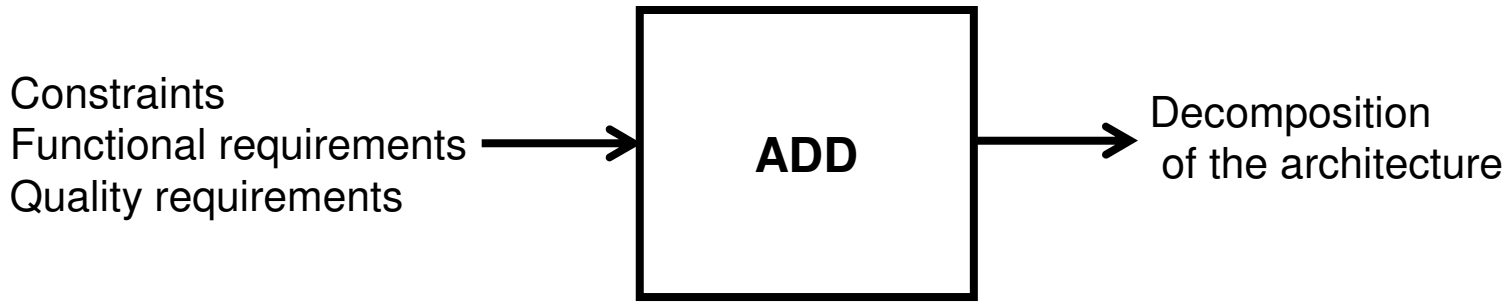


A view of possible architectures

Attribute-Driven Design (ADD) Method

The ADD method is an approach to defining software architectures by basing the design process on the architecture's quality attribute requirements.

It follows a recursive decomposition process where,
at each stage in the decomposition, tactics and architectural patterns are chosen to satisfy a set of quality attribute scenarios.



Steps of the ADD Method

1. Choose the element to decompose.
2. Refine the element according to these steps:
   a. Choose the architectural significant requirements.
   b. Choose an architectural pattern that satisfies the architectural significant requirements.
   c. Instantiate elements and allocate functionality from the use cases using multiple views.
   d. Define interfaces of the child elements.
   e. Verify and refine use cases and quality scenarios and make them constraints for the child elements.
3. Repeat these steps for the next element.

Remember that early decisions constrain later decisions. Make those with the biggest impact early.

Key Topics in Creating a Software Architecture

Scoping the problem

Defining/refining the architecture

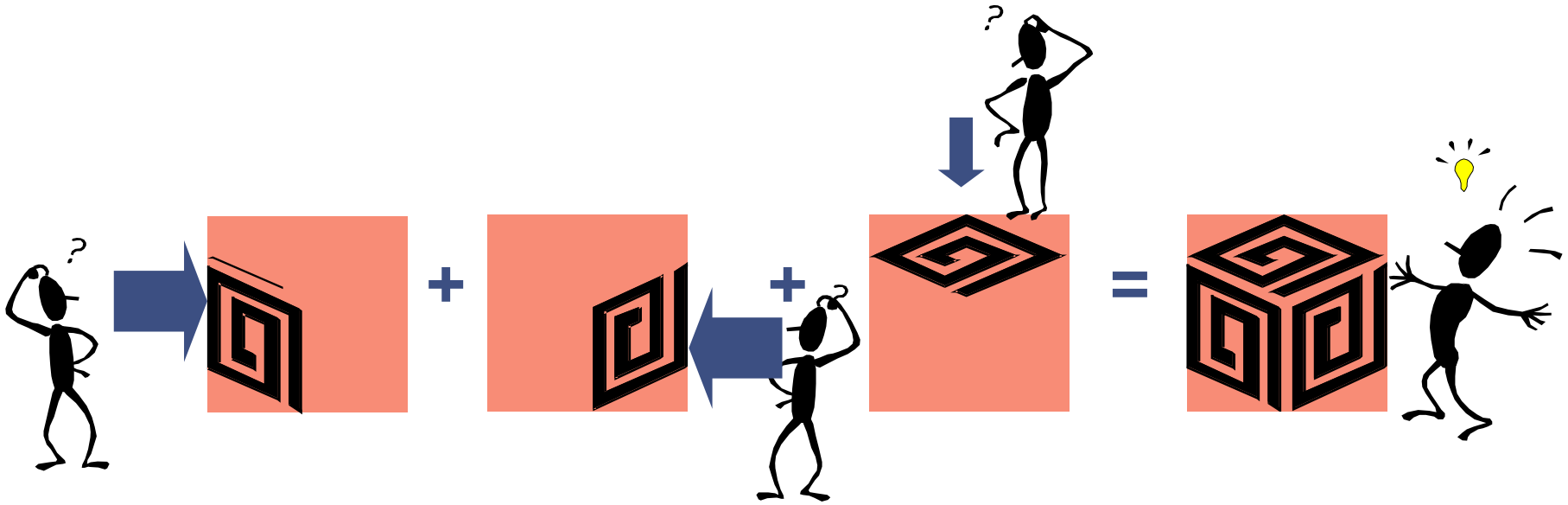

Documenting the architecture

Evaluating the architecture

View-Based Documentation

All modern approaches to software architecture creation and documentation are based on views. A general principle for documenting a software architecture is

> *Documenting a software architecture is a matter of documenting the relevant views and then adding information that applies to more than one view.*



Views

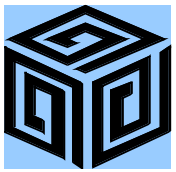
An architecture is a multidimensional construct, too involved to be seen all at once.

Systems are composed of many structures that show

- modules, their composition/decomposition and mapping to code units
- processes and how they synchronize
- programs and how they call or send data to each other
- how software is deployed on hardware
- how teams cooperate to build the system
- how components and connectors work at runtime
- …

Views are representations of structures. We use them to manage complexity by separating concerns.

What Is the "Right" Set of Views?



Unlike approaches that prescribe a fixed set of views, we take a more general approach:

Choose the best views for each situation.

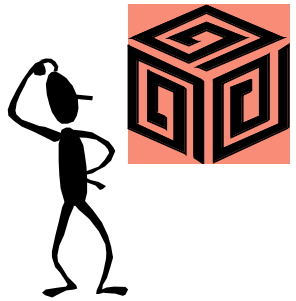
Which views are "right" depends on

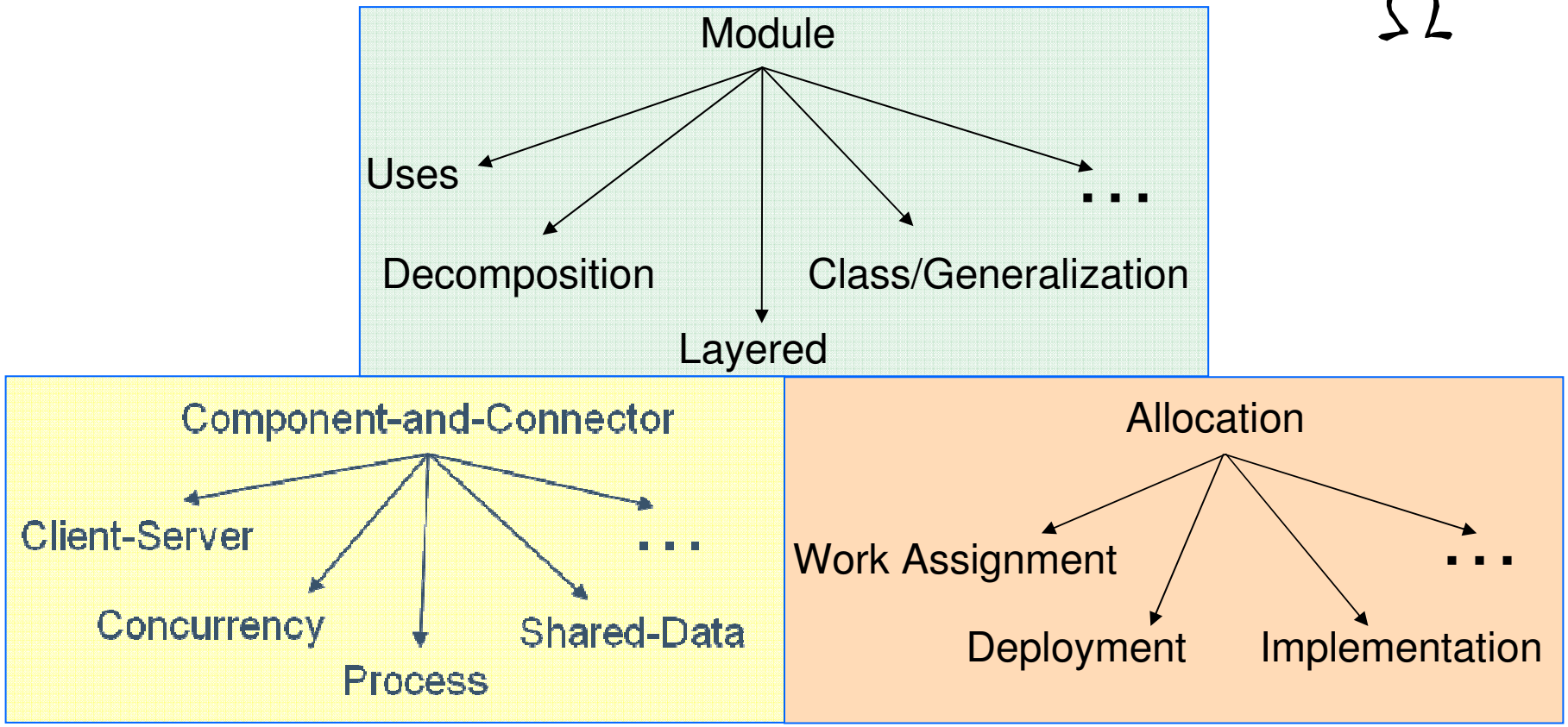
1. the structures that are inherent in the software
2. who the stakeholders are and how they will use the documentation

How do stakeholders use documentation?

- education—introducing people to the project
- communication—especially among stakeholders
  - architect to developers
  - architect to (current or future) architect
- analysis—assuring quality attributes

But Which Views to Consider?





Producing Documentation

Documenting individual views

- Unambiguous notations
- Enough information to support purpose
- Rationale!

Mapping between views

- Reconciling different perspectives to avoid inconsistencies
- Many analyses require information found in different views

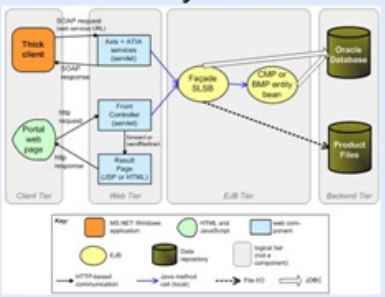
Standards compliance

- IEEE 1471, ISO/IEC 42010:2007

Etc.

Template for a View

Section 1. Primary Presentation

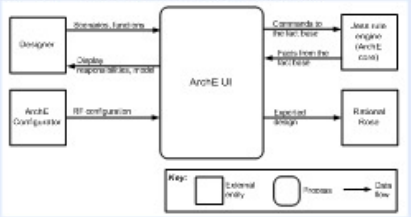


OR

Textual version of the primary presentation

Section 2. Element Catalog
- Section 2.A Elements
- Section 2.B Relations
- Section 2.C Element Interfaces
- Section 2.D Element Behavior

Section 3. Context Diagram



Section 4. Variability Guide

Section 5. Architecture Background
- Section 5.1 Design Rationale
- Section 5.2 Analysis Results
- Section 5.3 Assumptions

Section 6. Other Information

Key Topics in Creating a Software Architecture

Scoping the problem

Defining/refining the architecture

Documenting the architecture

➡ Evaluating the architecture

Why Evaluate an Architecture?

Because so much is riding on it!

- An unsuitable architecture can precipitate disaster.
- Architecture determines the structure of the project.

Because we can!

- Repeatable, structured methods offer a low-cost risk mitigation capability that can be employed early in the development life cycle.
- Making sure an architecture is the right one simply makes good sense.

Architecture evaluation should be a standard part of every architecture-based development methodology.

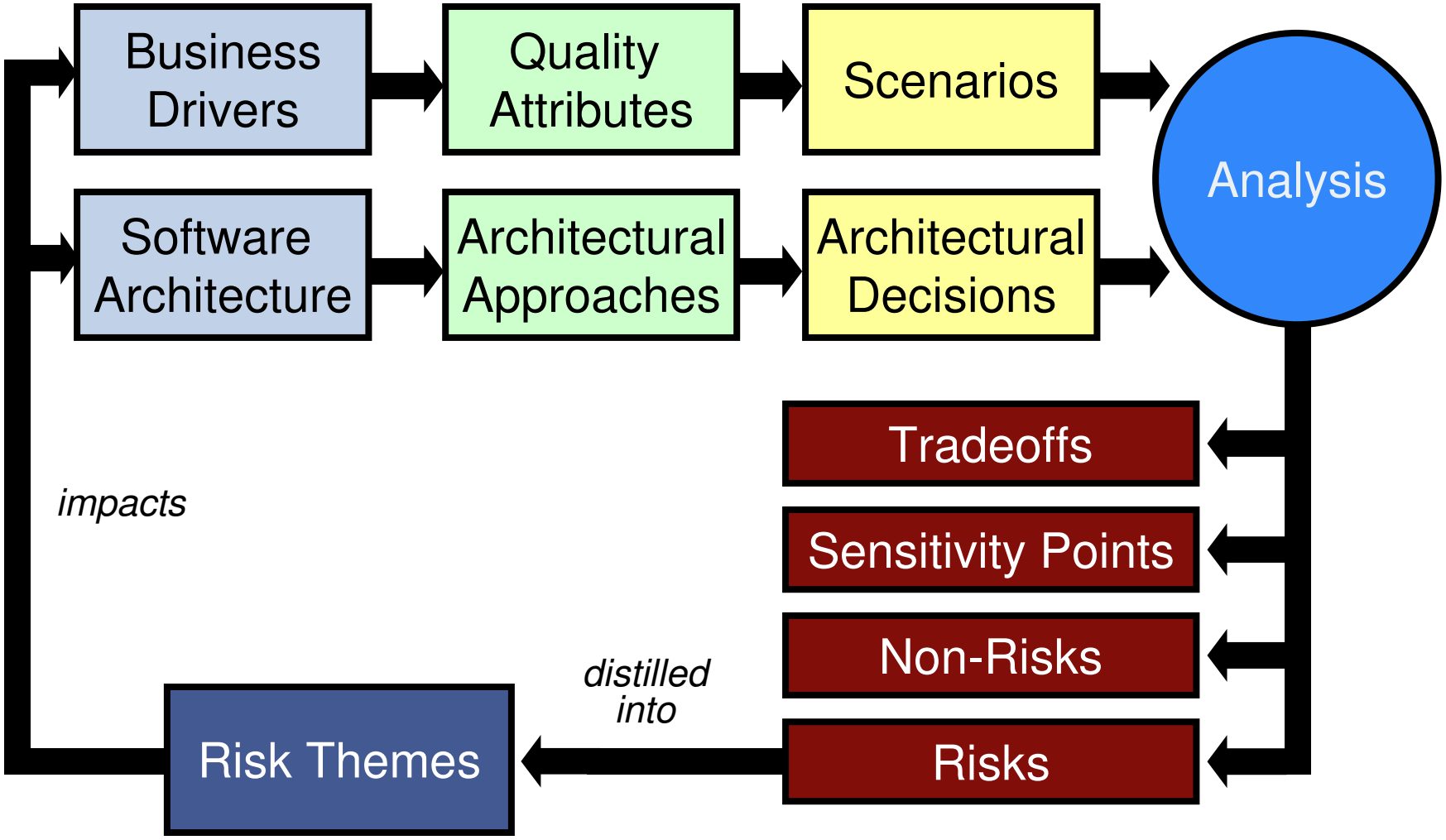
Evaluation Techniques

There are a variety of techniques for performing architecture evaluations, each having a different cost and providing different information.

These techniques fall into two broad categories:

1. questioning techniques
   - are applied to evaluate an architecture for any given reason
2. measuring techniques
   - are applied to answer questions about specific quality attributes

Conceptual Flow of the ATAM®



Typical Output from Evaluations

Set of ranked issues, risks, risk themes, or problem areas that

- have supporting data
- are contained in a formal report
- are used as feedback to the project

Set of scenarios, questions, or checklists for future use

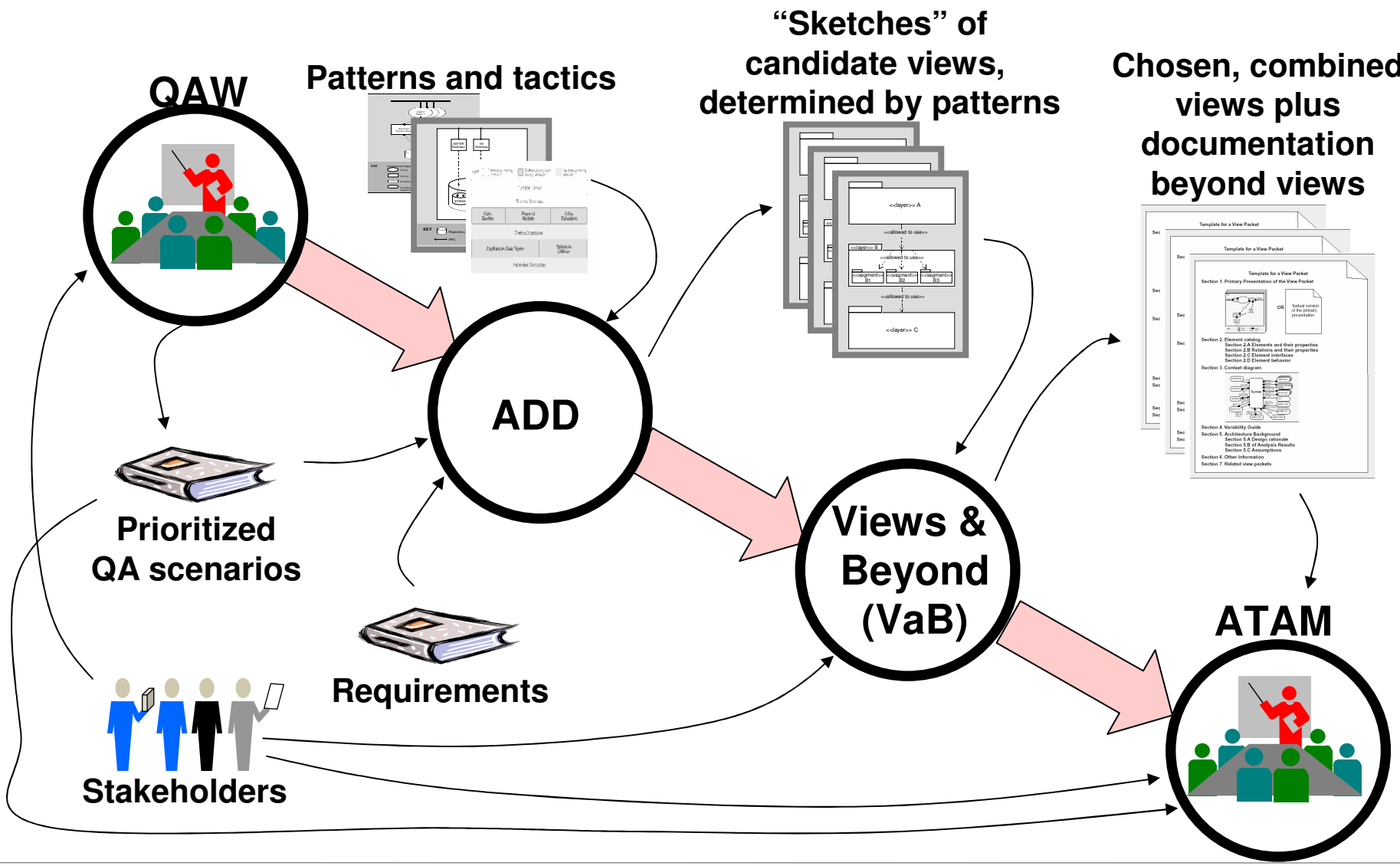
Identification of potentially reusable components

Enhanced system documentation

Estimation of the evaluation's costs and benefits of the evaluation

Improvements to the evaluation technique or process

SEI Software Architecture Methods & Techniques



For More Information

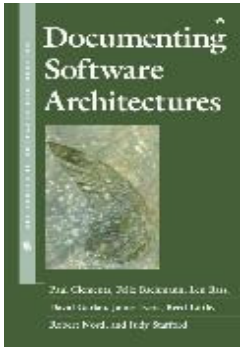
James Ivers
Email: jivers@sei.cmu.edu

World Wide Web:

http://www.sei.cmu.edu/architecture

- Technical reports
- Case studies
- Tools & templates

Software Architecture in Practice, 2nd Edition



Documenting Software Architectures: Views and Beyond

Evaluating Software Architectures: Methods and Case Studies