Software Architecture Technology Initiative

Mark Klein

Third Annual SATURN Workshop

May 2007

Presentation Outline

Getting (Re)acquainted

Transition

Current Work and Challenges

Product Line Systems Program

Our Mission:

To effect widespread product line practice, architecture-centric development and evolution, and predictable software construction throughout the global software community.

Product Line System initiatives:

- ***Software Architecture Technology (SAT) Initiative***
- Product Line Practice Initiative
- Predictable Assembly from Certifiable Components Initiative

Focus: Software Architecture

The quality and longevity of a software system is largely determined by its architecture.

Too many experiences point to inadequate software architecture education and practices and the lack of any real software architecture evaluation early in the life cycle.

Without an explicit course of action focused on software architecture, these experiences are being and will be repeated.

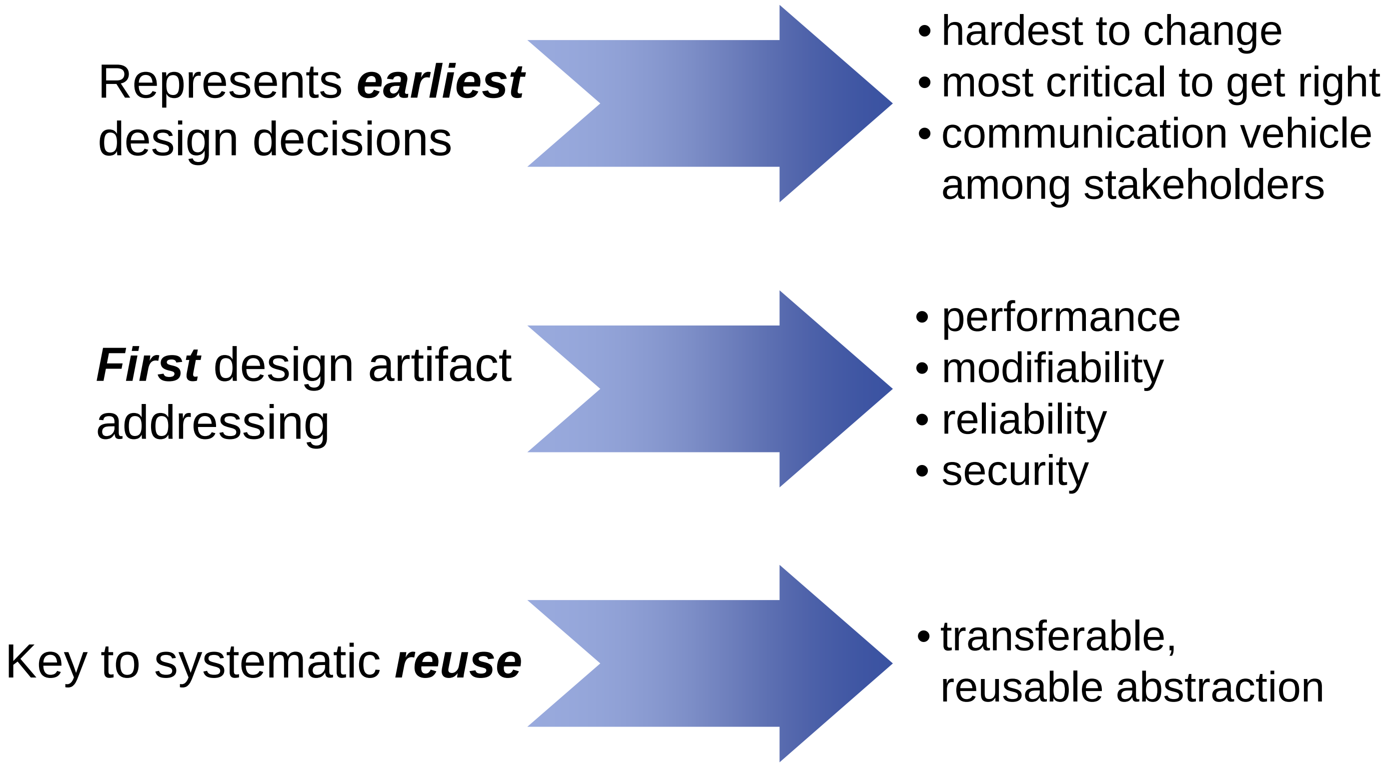
The cost of inaction is too great.

What Is a Software Architecture?

“The **software architecture** of a program or computing system is the structure or **structures of the system,** which comprise the software elements, the **externally visible properties** of those elements, and the **relationships among** them.”

Bass, L.; Clements; P. & Kazman, R. *Software Architecture in Practice, Second Edition.* Boston, MA: Addison-Wesley, 2003.

Why Is Software Architecture Important?



Represents ***earliest*** design decisions

- hardest to change
- most critical to get right
- communication vehicle among stakeholders

First design artifact addressing

- performance
- modifiability
- reliability
- security

Key to systematic ***reuse***

- transferable, reusable abstraction

The **right architecture** paves the way for system **success.**
The **wrong architecture** usually spells some form of **disaster.**

SEI Software Architecture Technology (SAT) Initiative’s Focus

Ensure that business and mission goals are predictably achieved by using effective software architecture practices throughout the development lifecycle.

“Axioms” Guiding Our Work

- Software architecture is the bridge between business and mission goals and a software-intensive system.
- Quality attribute requirements drive software architecture design.
- Software architecture drives software development throughout the life cycle.

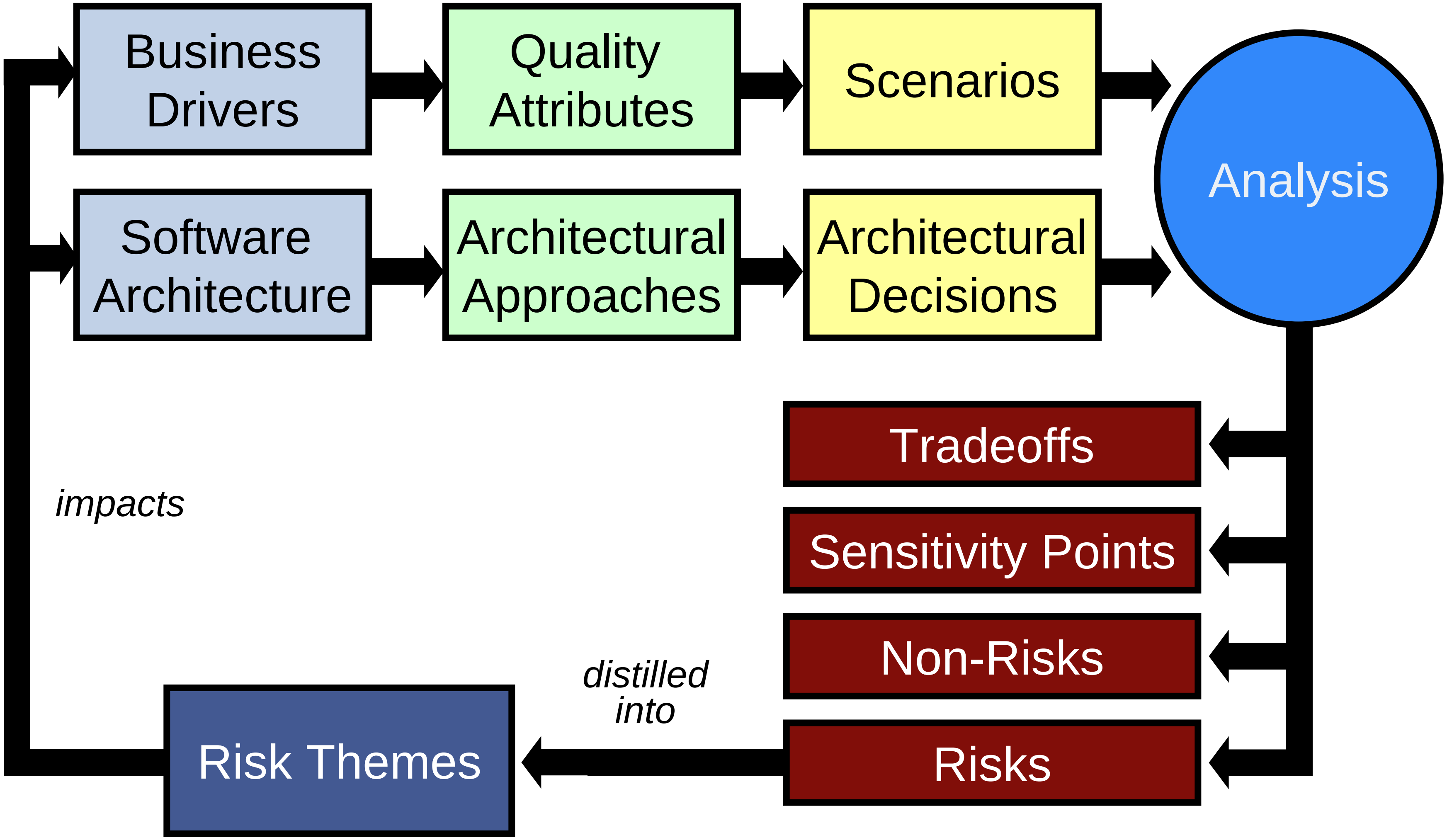
Earliest work focused on the second axiom leading to the Architecture Tradeoff Analysis Method® (ATAM ®)

SEI’s Architecture Tradeoff Analysis Method® (ATAM®)

The ATAM is an architecture evaluation method that focuses on multiple quality attributes

- illuminates points in the architecture where quality attribute tradeoffs occur
- generates a context for ongoing quantitative analysis
- utilizes an architecture’s vested stakeholders as authorities on the quality attribute goals

Conceptual Flow of the ATAM®



Architecture-Centric Development Activities

Architecture-centric activities include the following:

- creating the business case for the system
- understanding the requirements
- creating and/or selecting the architecture
- documenting and communicating the architecture
- analyzing or evaluating the architecture
- implementing the system based on the architecture
- ensuring that the implementation conforms to the architecture

ATAM® Led to the Development of Other Methods and Techniques

What if the quality requirements are not well-understood?

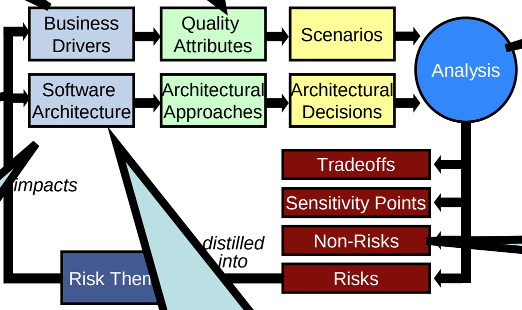
Quality Attribute Workshop (QAW)

Our scenarios tend to be incomplete or ambiguous.

Quality Attribute General Scenarios

What are some of the most important questions to ask?

Quality Attribute Tactics



What if there's no architecture?

Attribute Driven Design (ADD)

Which risks should I work on first?

Cost Benefit Analysis Method (CBAM)

What if I don't know my system's architecture?

Architecture Reconstruction using ARMIN

What information should be included in my architecture documentation?

Views and Beyond Approach (VaB)

Characteristics of SEI Methods

QAW

ADD

Views and Beyond

ATAM

CBAM

ARMIN

- are explicitly focused on quality attributes
- directly link to business and mission goals
- explicitly involve system stakeholders
- are grounded in state-of-the-art quality attribute models and reasoning frameworks
- are documented for practitioner consumption
- are applicable to DoD challenges and DoD systems

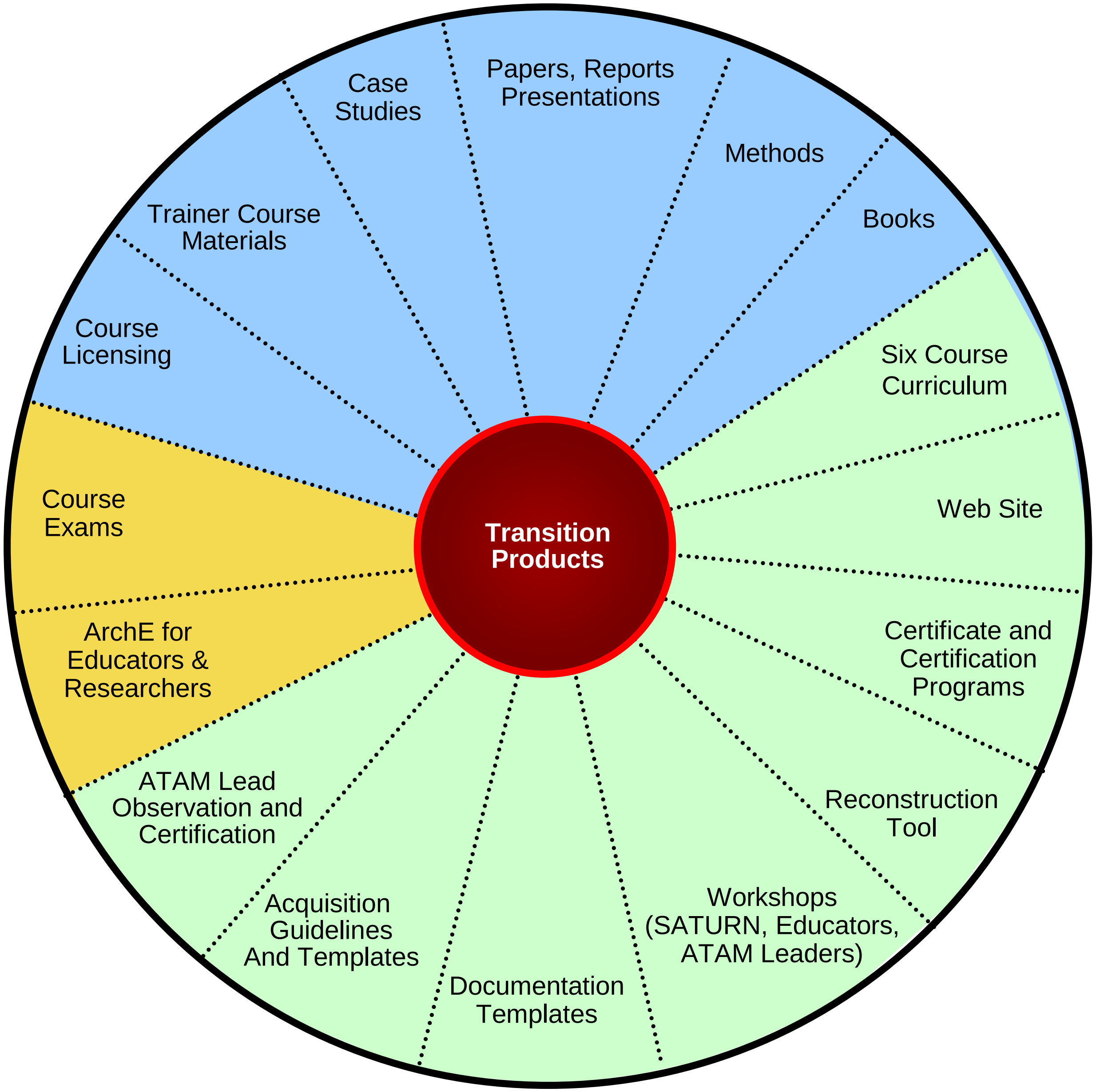
Presentation Outline

Getting (Re)acquainted

Transition

Current Work and Challenges

SAT: Transition


Transition Products
Case Studies
Papers, Reports Presentations
Methods
Books
Trainer Course Materials
Course Licensing
Six Course Curriculum
Web Site
Course Exams
ArchE for Educators & Researchers
Certificate and Certification Programs
ATAM Lead Observation and Certification
Reconstruction Tool
Acquisition Guidelines And Templates
Workshops (SATURN, Educators, ATAM Leaders)
Documentation Templates

KEY:

- 🟦 Ongoing
- 🟩 In Sustainment
- 🟨 Just Begun

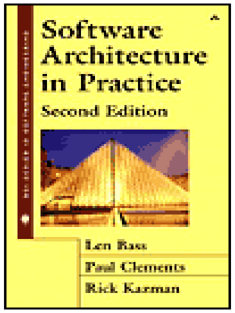
Certificate Program Course Matrix

	Three Certificate Programs		
Requirements	**Software Architecture Professional**	**ATAM® Evaluator**	**ATAM® Lead Evaluator**
Software Architecture: Principles and Practice	✓	✓	✓
Documenting Software Architectures	✓		✓
Software Architecture Design and Analysis	✓		✓
Software Product Lines	✓		
ATAM ® Evaluator Training		✓	✓
ATAM ® Leader Training			✓
ATAM ® Observation			✓

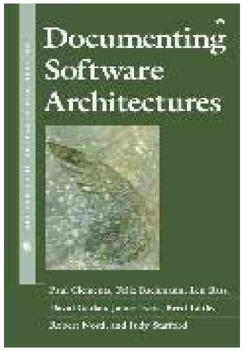
Architecture Tradeoff Analysis Method ® (ATAM ®)

Associated Texts

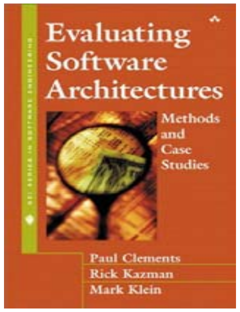
Software Architecture in Practice, 2nd Edition



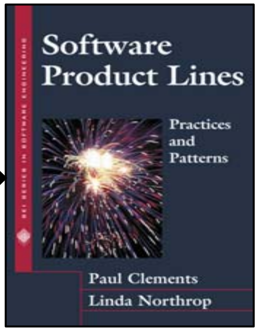
Documenting Software Architectures: Views and Beyond


Documenting Software Architectures
Paul Clements, Felix Bachmann, Len Bass,
David Garlan, James Ivers, Reed Little,
Robert Nord, and Judy Stafford

Evaluating Software Architectures: Methods and Case Studies



Software Product Lines: Practices and Patterns



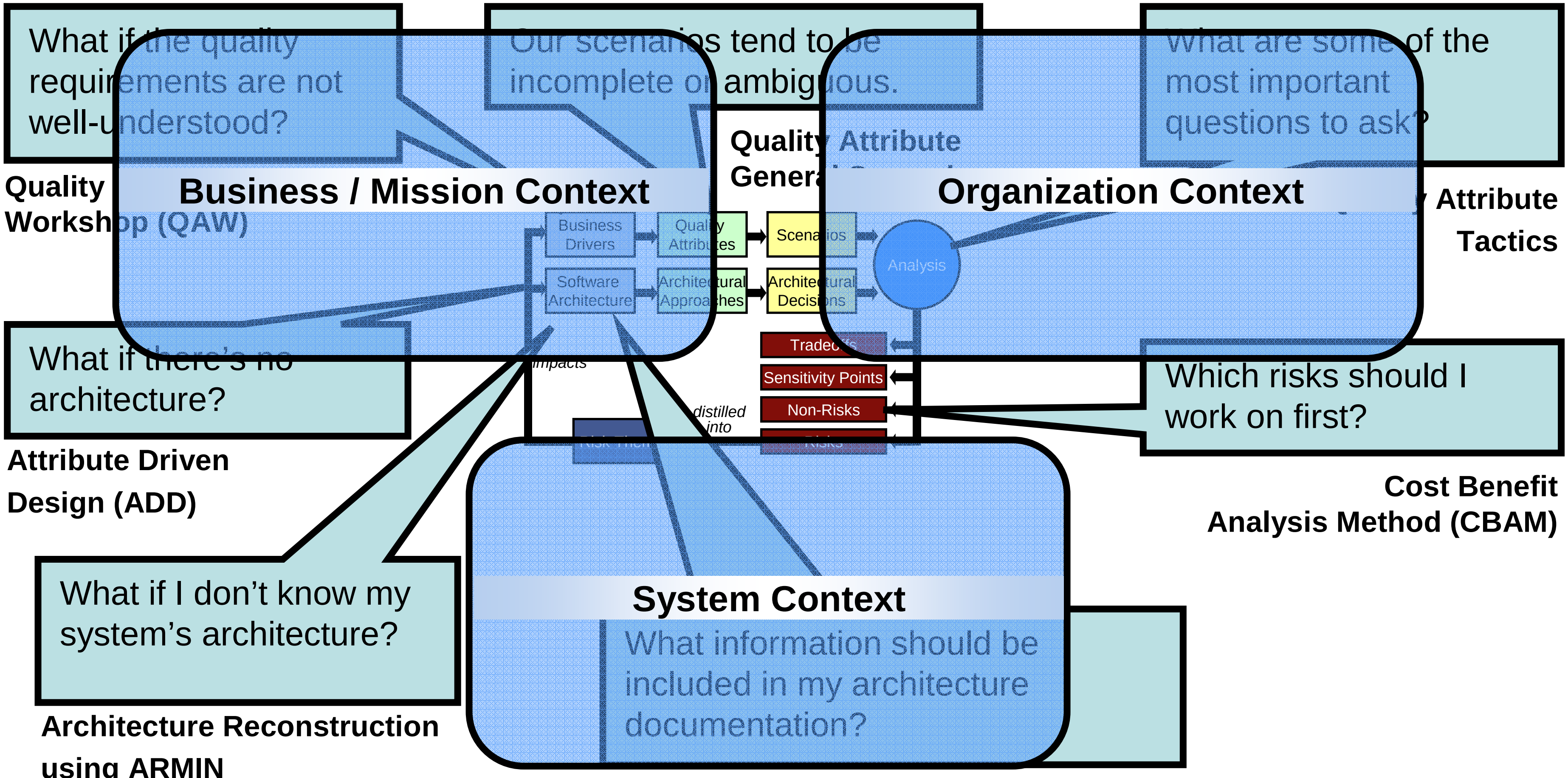
Presentation Outline

Getting (Re)acquainted

Transition

Current Work and Challenges

ATAM® Led to the Development of Other Methods and Techniques



What if the quality requirements are not well-understood?

Quality Attribute Workshop (QAW)

Our scenarios tend to be incomplete or ambiguous.

Quality Attribute General Scenarios

What are some of the most important questions to ask?

Quality Attribute Tactics

What if there's no architecture?

Attribute Driven Design (ADD)

Which risks should I work on first?

Cost Benefit Analysis Method (CBAM)

What if I don't know my system's architecture?

Architecture Reconstruction using ARMIN

What information should be included in my architecture documentation?

Views and Beyond Approach (VaB)

Architecture Evolution - 1

Problem

- The architecture of a software intensive system must continually evolve to ensure consistency between the system and its **mission and business goals**
  - "Tactical evolution" focuses on change over a short time horizon to ensure system consistency with current business and mission goals.
  - "Strategic evolution" focuses on change over a long time horizon with an emphasis on handling uncertainty in future business and mission goals.

Architecture Evolution - 2

Approach

- Leverage generality and composability of SEI architecture-centric practices to create need-specific methods to support evolution
  - architecture fact finding
  - architecture improvement
  - comparing architectures
  - enhanced cost-benefit analysis
- Link quality attribute tactics with patterns
- Use economic models, such as real options, in tandem with quality attribute models

Architecture Competence

Problem

- To date, we have focused on the "technical aspects" of software architecture, not the people and organizational aspects.
- To facilitate organizational adoption and improvement of architecture-centric software engineering practices organizations need help in measuring and improving the architecture of their individuals and teams

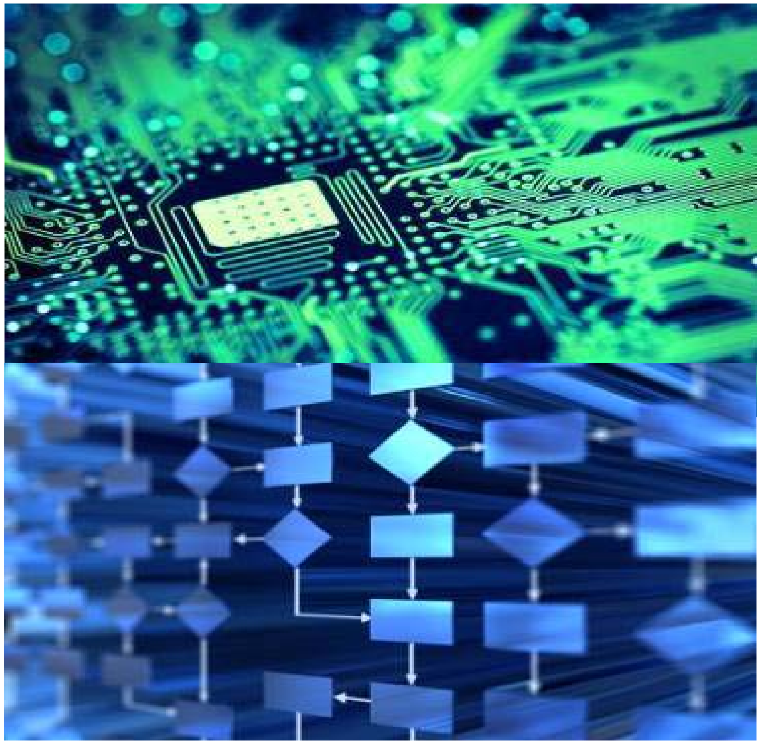
Approach

- Exploit relevant models
  - Organizational coordination mechanisms
  - Human performance model
  - Organization learning
- The work also involves
  - Exploring the relationship between business goals and quality attributes
  - Surveying community about best practices for architects and organizations
  - Crafting pilot assessment instruments
  - Pursuing case studies in competence improvement

System ATAM and SoS Architecture Evaluation

Problem

- Severe integration and runtime problems arise due to inconsistencies in how quality attributes are addressed in **system and software architectures.**
- This is further exacerbated in a System of Systems (SoS) context where major system and software elements are developed concurrently.



Approach

- Make minor enhancements to the ATAM for use on system architectures.
- Develop a method to perform a "first pass" identification of inconsistencies between constituent systems of SoSs by using mission threads augmented with quality attribute concerns.

A Sampling of New Challenges

Providing automated support for architecture design and evolution while accounting for trade-offs.

- *Our Research: Developing an architecture design assistant, which provides quality attribute, architecture design and trade-off assistance.*

Managing uncertainty in future business and mission goals

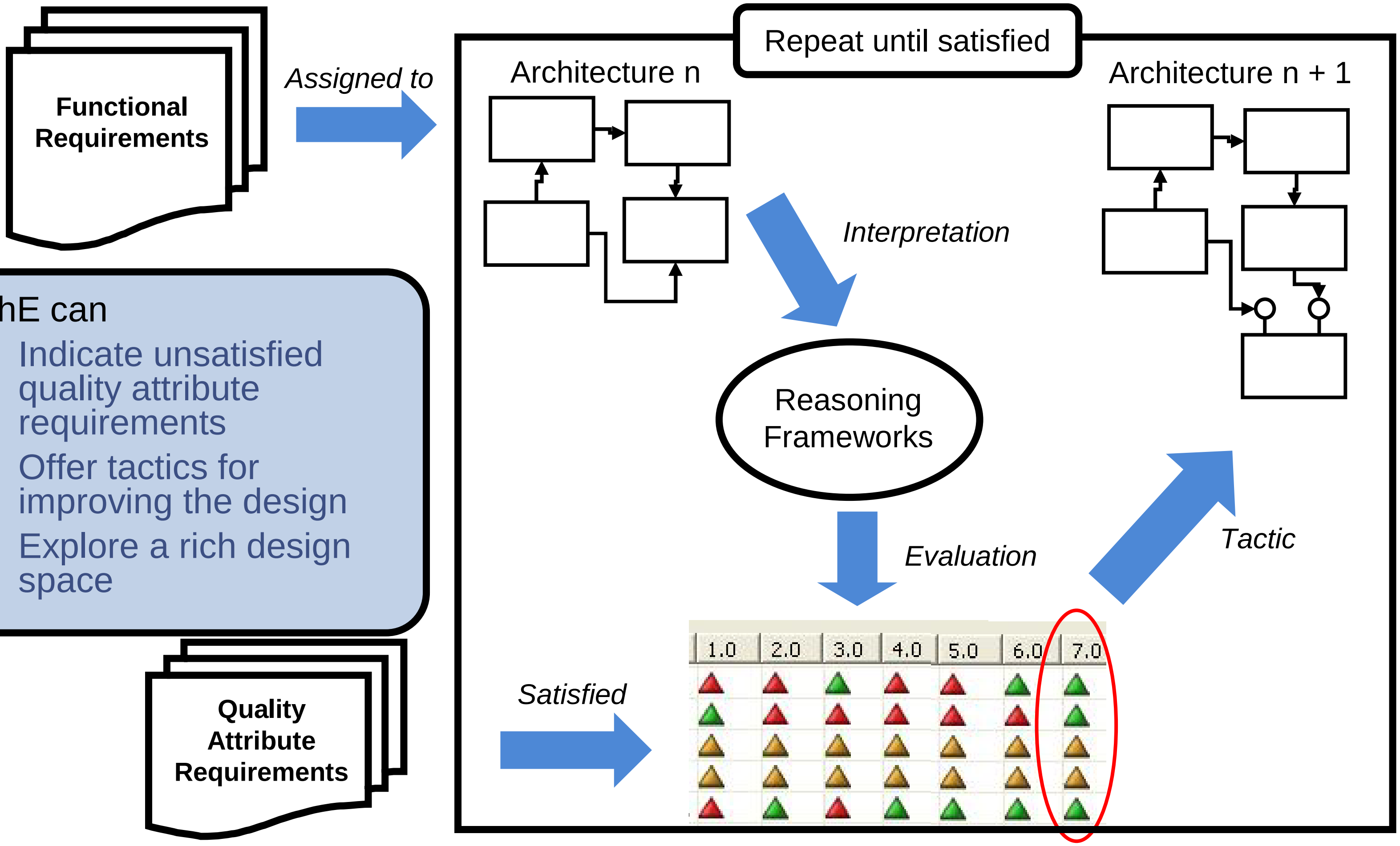
- *Our Research: Using real options to determine the value of flexibility.*

Ensuring that our architecture models and methods apply to emerging technologies and contexts such as service-oriented architectures, system of systems, and ultra-large scale systems

- *Our Research: Applying our methods to service-oriented architectures and determining architecture concepts and approaches relevant to system of systems and ultra-large-scale systems.*

Predictability by Design

Problem: guide an architect in producing a design satisfying multiple (possibly conflicting) quality attribute requirements.



ArchE can

- Indicate unsatisfied quality attribute requirements
- Offer tactics for improving the design
- Explore a rich design space

Application of Real Options to Architecture

An option is the right, but not the obligation to take an action in the future when there is uncertainty

Architecture evolution involves uncertainty

- managing uncertainty requires flexibility in making design decisions
- real options provide a tool to guide evolution
- flexibility is valuable; how much is it worth?

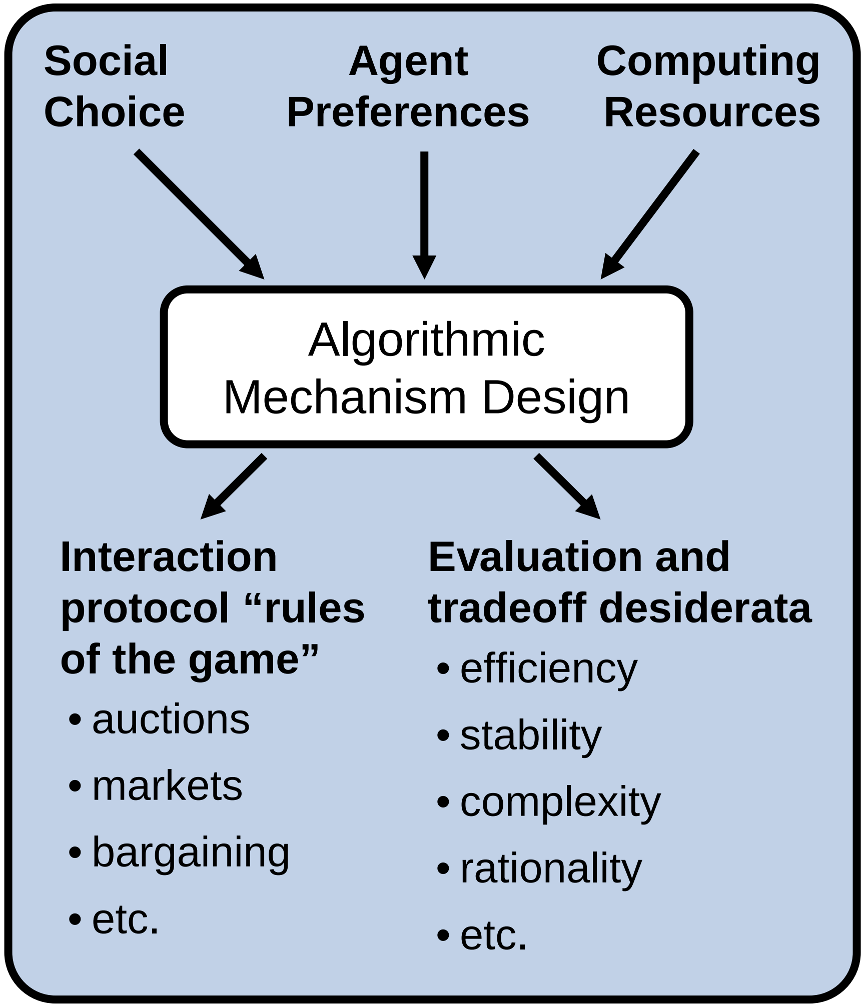
Sources of uncertainty that have implications for software architecture

- business goals (e.g. time to market, interoperability)
- resources (e.g. access to information for the decision and developer time)
- key quality attributes (e.g. search latency should be less than 1 second and system should by 99.99999% available)

Mechanism Design

Problem: What if there is not central locus of control to manage and coordinate system design and evolution?

We are investigating: Mechanism design uses economics and game theory to obtain desired global solutions for systems that have many self-interested participants



Social Choice

Agent Preferences

Computing Resources

Algorithmic Mechanism Design

Interaction protocol "rules of the game"

- auctions
- markets
- bargaining
- etc.

Evaluation and tradeoff desiderata

- efficiency
- stability
- complexity
- rationality
- etc.

We want your input!

Our ongoing goals are to

- Respond to the needs of the world
- Increase our level of impact
- Base techniques and methods on theoretically sound foundations

We are very much looking forward to getting your thoughts!