Analysis of Software Architectures

Software Architecture
Lecture 13

Learning Objectives

- Define architectural analysis and enumerate its goals
- Apply ATAM analysis to software architectures
- Apply Model-Based Analysis to software architecture
- Apply Reliability Analysis to software architecture
- Apply XTEAM Analysis to software architecture

What Is Architectural Analysis?

- Architectural analysis is the activity of discovering important system properties using the system's architectural models.
  - Early, useful answers about relevant architectural aspects
  - Available prior to system's construction
- Important to know
  1. which questions to ask
  2. why to ask them
  3. how to ask them
  4. how to ensure that they can be answered

Concerns Relevant to Architectural Analysis

- Goals of analysis
- Scope of analysis
- Primary architectural concern being analyzed
- Level of formality of architectural models
- Type of analysis
- Level of automation
- System stakeholders interested in analysis

Architectural Analysis Goals

- The four "C"s
  - Completeness
  - Consistency
  - Compatibility
  - Correctness

Architectural Analysis Goals – Completeness

- Completeness is both an external and an internal goal
- It is *external* with respect to system requirements
  - Challenged by the complexity of large systems' requirements and architectures
  - Challenged by the many notations used to capture complex requirements as well as architectures
- It is *internal* with respect to the architectural intent and modeling notation
  - Have all elements been fully modeled in the notation?
  - Have all design decisions been properly captured?

Architectural Analysis Goals – Consistency

- Consistency is an internal property of an architectural model
- Ensures that different model elements do not contradict one another
- Dimensions of architectural consistency
  - Name
  - Interface
  - Behavior
  - Interaction
  - Refinement

Name Consistency

- Component and connector names
- Component service names
- May be non-trivial to establish at the architectural level
  - Multiple system elements/services with identical names
  - Loose coupling via publish-subscribe or asynchronous event broadcast
  - Dynamically adaptable architectures

Interface Consistency

- Encompasses name consistency
- Also involves parameter lists in component services
- A rich spectrum of choices at the architectural level
- Example: matching provided and required interfaces

```
ReqInt:   getSubQ(Natural first, Natural last, Boolean remove)
          returns FIFOQueue;

ProvInt1: getSubQ(Index first, Index last)
          returns FIFOQueue;

ProvInt2: getSubQ(Natural first, Natural last, Boolean remove)
          returns Queue;
```

Behavioral Consistency

- Names and interfaces of interacting components may match, but behaviors need not
- Example: subtraction

```
subtract(Integer x, Integer y) returns Integer;
```

  - Can we be sure what the *subtract* operation does?
- Example: QueueClient and QueueServer components

QueueClient

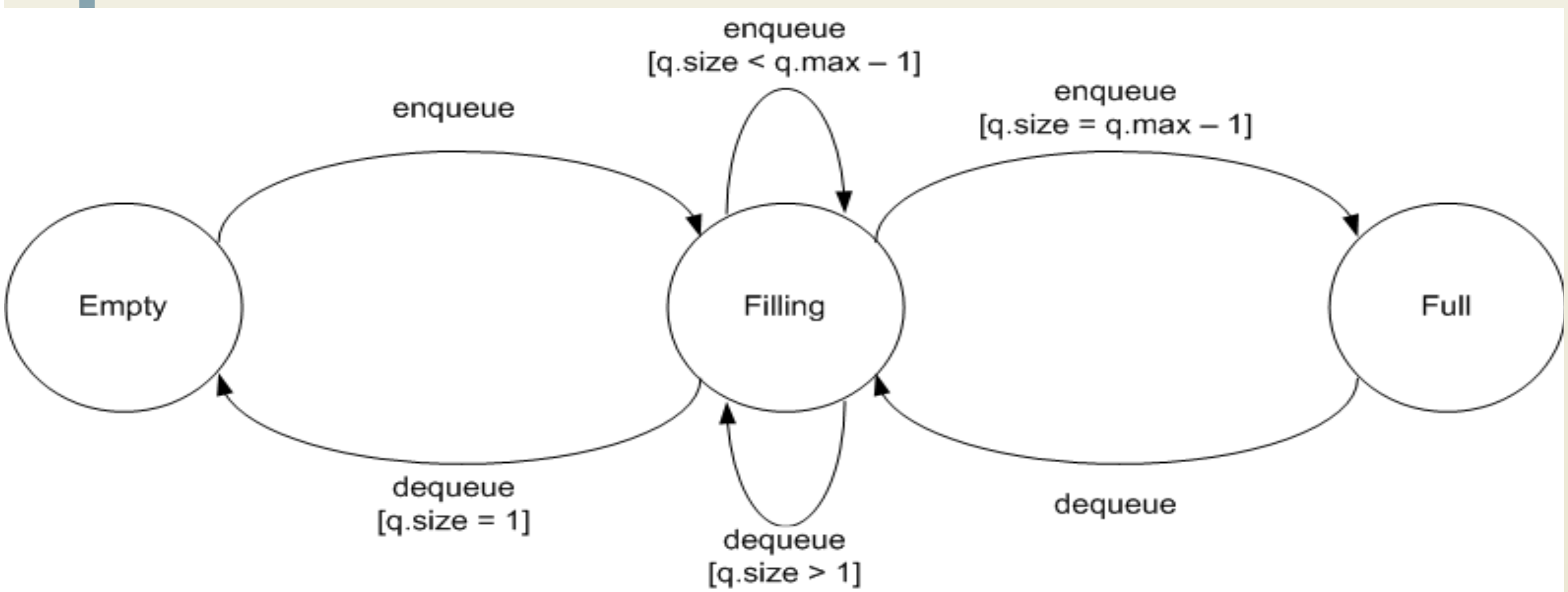
```
precondition  q.size > 0;
postcondition ~q.size = q.size;
```

QueueServer

```
precondition  q.size > 1;
postcondition ~q.size = q.size - 1;
```

Interaction Consistency

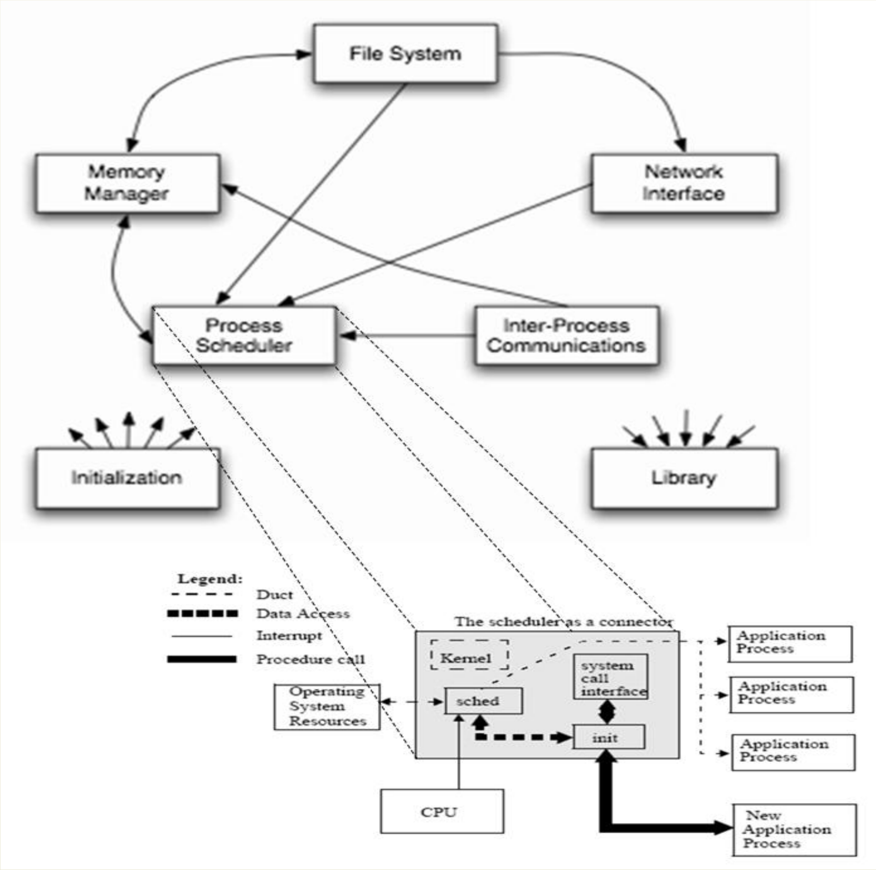
- Names, interfaces, and behaviors of interacting components may match, yet they may still be unable to interact properly
- Example: QueueClient and QueueServer components



Refinement Consistency

- Architectural models are refined during the design process
- A relationship must be maintained between higher and lower level models
  - All elements are preserved in the lower level model
  - All design decisions are preserved in the lower-level model
  - No new design decisions violate existing design decisions

Refinement Consistency Example



Architectural Analysis Goals – Compatibility

- Compatibility is an external property of an architectural model
- Ensures that the architectural model adheres to guidelines and constraints of
  - a style
  - a reference architecture
  - an architectural standard

Architectural Analysis Goals – Correctness

- Correctness is an external property of an architectural model
- Ensures that
  1. the architectural model fully realizes a system specification
  2. the system's implementation fully realizes the architecture
- Inclusion of OTS elements impacts correctness
  - System may include structural elements, functionality, and non-functional properties that are not part of the architecture
  - The notion of *fulfillment* is key to ensuring architectural correctness

Learning Objectives

- Define architectural analysis and enumerate its goals
- Apply ATAM analysis to software architectures
- Apply Model-Based Analysis to software architecture
- Apply Reliability Analysis to software architecture
- Apply XTEAM Analysis to software architecture

Architectural Inspections and Reviews

- Architectural models studied by human stakeholders for specific properties
- The stakeholders define analysis objective
- Manual techniques
  - Can be expensive
- Useful in the case of informal architectural descriptions
- Useful in establishing "soft" system properties
  - E.g., scalability or adaptability
- Able to consider multiple stakeholders' objectives and multiple architectural properties

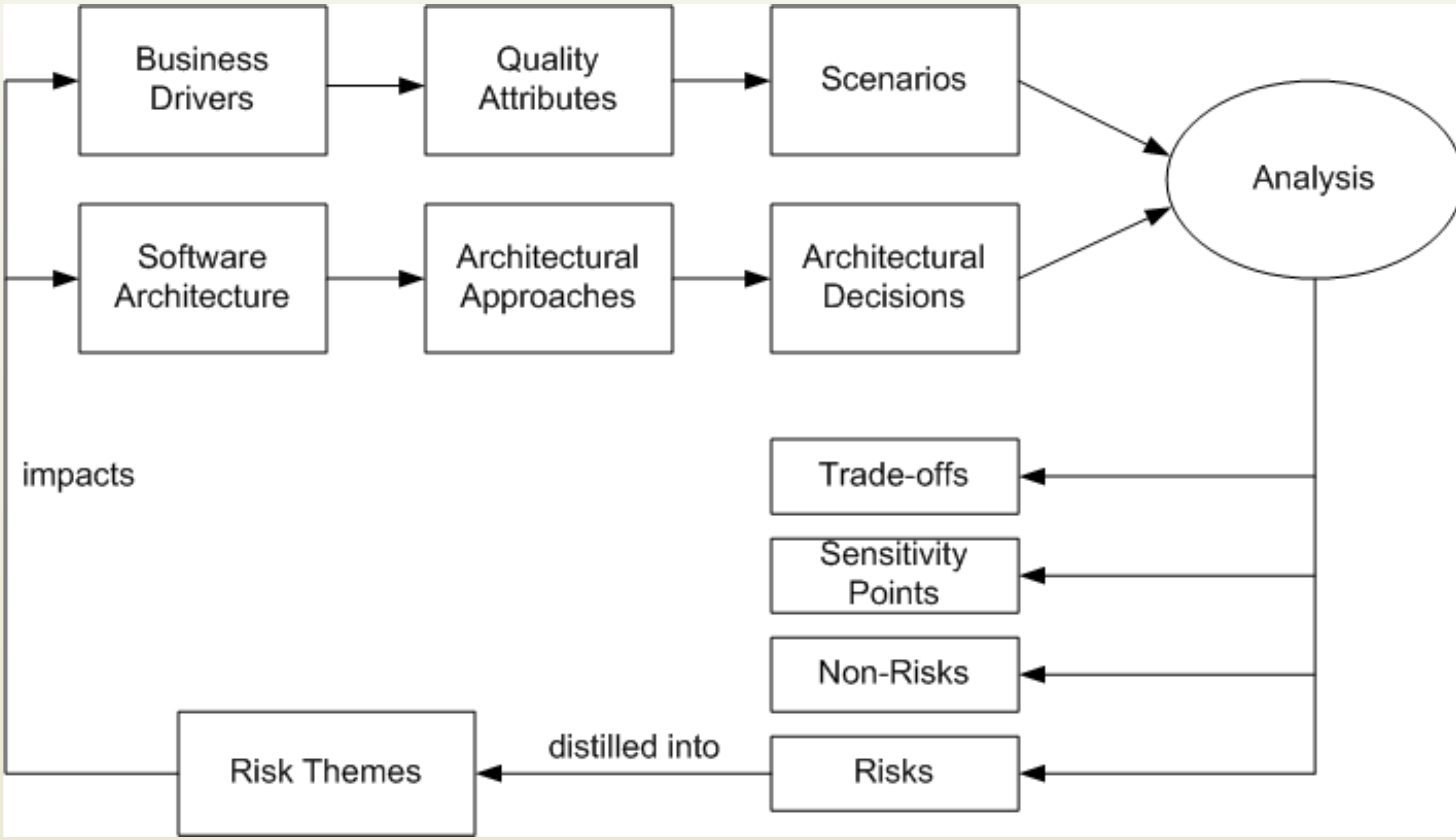
Inspections and Reviews in a Nutshell

- *Analysis Goals* – any
- *Analysis Scope* – any
- *Analysis Concern* – any, but particularly suited for non-functional properties
- *Architectural Models* – any, but must be geared to stakeholder needs and analysis objectives
- *Analysis Types* – mostly static and scenario-based
- *Automation Level* – manual, human intensive
- *Stakeholders* – any, except perhaps component vendors

Example – ATAM

- Stands for architectural trade-off analysis method
- Human-centric process for identifying risks early on in software design
- Focuses specifically on four quality attributes (NFPs)
  - Modifiability
  - Security
  - Performance
  - Reliability
- Reveals how well an architecture satisfies quality goals and how those goals trade-off

ATAM Process



ATAM Business Drivers

- The system’s critical functionality
- Any technical, managerial, economic, or political constraints
- The project’s business goals and context
- The major stakeholders
- The principal quality attribute (NFP) goals

ATAM Scenarios

- Use-case scenarios
  - Describe how the system is envisioned by the stakeholders to be used
- Growth scenarios
  - Describe planned and envisioned modifications to the architecture
- Exploratory scenarios
  - Try to establish the limits of architecture's adaptability with respect to
    - system's functionality
    - operational profiles
    - underlying execution platforms
  - Scenarios are prioritized based on importance to stakeholders

ATAM Architecture

- Technical constraints
  - Required hardware platforms, OS, middleware, programming languages, and OTS functionality
- Any other systems with which the system must interact
- *Architectural approaches* that have been used to meet the quality requirements
  - Sets of architectural design decisions employed to solve a problem
  - Typically architectural patterns and styles

ATAM Analysis

- Key step in ATAM
- Objective is to establish relationship between architectural approaches and quality attributes
- For each architectural approach a set of analysis questions are formulated
  - Targeted at the approach and quality attributes in question
- System architects and ATAM evaluation team work together to answer these questions and identify
  - Risks → these are distilled into risk *themes*
  - Non-Risks
  - Sensitivity points
  - Trade-off points
- Based on answers, further analysis may be performed

ATAM in a Nutshell

Goals	Completeness Consistency Compatibility Correctness`
Scope	Subsystem- and system-level Data exchange
Concern	Non-functional
Models	Informal Semi-formal
Type	Scenario-driven
Automation Level	Manual
Stakeholders	Architects Developers Managers Customers

Learning Objectives

- Define architectural analysis and enumerate its goals
- Apply ATAM analysis to software architectures
- Apply Model-Based Analysis to software architecture
- Apply Reliability Analysis to software architecture
- Apply XTEAM Analysis to software architecture

Model-Based Architectural Analysis

- Analysis techniques that manipulate architectural description to discover architectural properties
- Tool-driven, hence potentially less costly
- Typically useful for establishing “hard” architectural properties only
  - Unable to capture design intent and rationale
- Usually focus on a single architectural aspect
  - E.g., syntactic correctness, deadlock freedom, adherence to a style
- Scalability may be an issue
- Techniques typically used in tandem to provide more complete answers

Model-Based Analysis in a Nutshell

- *Analysis Goals* – consistency, compatibility, internal correctness
- *Analysis Scope* – any
- *Analysis Concern* – structural, behavioral, interaction, and possibly non-functional properties
- *Architectural Models* – semi-formal and formal
- *Analysis Types* – static
- *Automation Level* – partially and fully automated
- *Stakeholders* – mostly architects and developers

Model-Based Analysis in ADLs

- Wright – uses CSP to analyze for deadlocks
- Aesop – ensures style-specific constraints
- MetaH and UniCon – support schedulability analysis via NFPs such as component criticality and priority
- ADL parsers and compilers – ensure syntactic and semantic correctness
  - E.g., Rapide's generation of executable architectural simulations
- Architectural constraint enforcement
  - E.g., Armani or UML's OCL
- Architectural refinement
  - E.g., SADL and Rapide

ADLs' Analysis Foci in a Nutshell

Goals	Consistency Compatibility Completeness (internal)
Scope	Component- and connector-level Subsystem- and system-level Data exchange Different abstraction levels Architecture comparison
Concern	Structural Behavioral Interaction Non-functional
Models	Semi-formal Formal
Type	Static
Automation Level	Partially automated Automated
Stakeholders	Architects Developers Managers Customers

Learning Objectives

- Define architectural analysis and enumerate its goals
- Apply ATAM analysis to software architectures
- Apply Model-Based Analysis to software architecture
- Apply Reliability Analysis to software architecture
- Apply XTEAM Analysis to software architecture

Architectural Reliability Analysis

- *Reliability* is the probability that the system will perform its intended functionality under specified design limits, without failure
- A *failure* is the occurrence of an incorrect output as a result of an input value that is received, with respect to the specification
- An *error* is a mental mistake made by the designer or programmer
- A *fault* or a *defect* is the manifestation of that error in the system
  - An abnormal condition that may cause a reduction in, or loss of, the capability of a component to perform a required function
  - A requirements, design, or implementation flaw or deviation from a desired or intended state

Reliability Metrics

- Time to failure
- Time to repair
- Time between failures

Assessing Reliability at Architectural Level

- Challenged by unknowns
  - Operational profile
  - Failure and recovery history
- Challenged by uncertainties
  - Multiple development scenarios
  - Varying granularity of architectural models
  - Different information sources about system usage
- Architectural reliability values must be qualified by assumptions made to deal with the above uncertainties
- Reliability modeling techniques are needed that deal effectively with uncertainties
  - E.g., Hidden Markov Models (HMMs)

Architectural Reliability Analysis in a Nutshell

Goals	Consistency Compatibility Correctness
Scope	Component- and connector-level Subsystem- and system-level
Concern	Non-functional
Models	Formal
Type	Static Scenario-based
Automation Level	Partially automated
Stakeholders	Architects Managers Customers Vendors

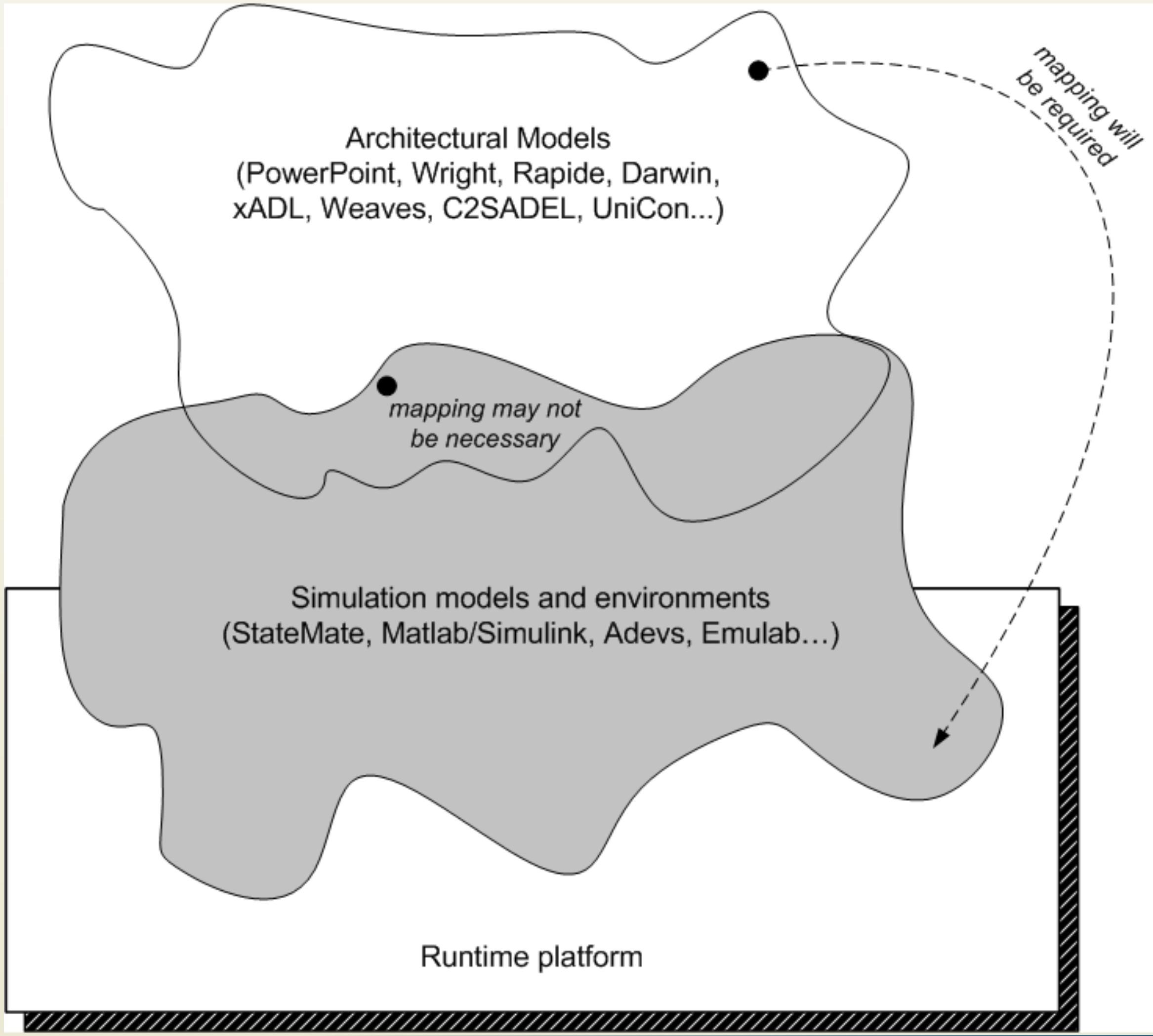
Learning Objectives

- Define architectural analysis and enumerate its goals
- Apply ATAM analysis to software architectures
- Apply Model-Based Analysis to software architecture
- Apply Reliability Analysis to software architecture
- Apply XTEAM Analysis to software architecture

Simulation-Based Analysis

- Requires producing an executable system model
- Simulation need not exhibit identical behavior to system implementation
  - Many low-level system parameters may be unavailable
- It needs to be precise and not necessarily accurate
- Some architectural models may not be amenable to simulation
  - Typically require translation to a simulatable language

Architectural and Simulation Models



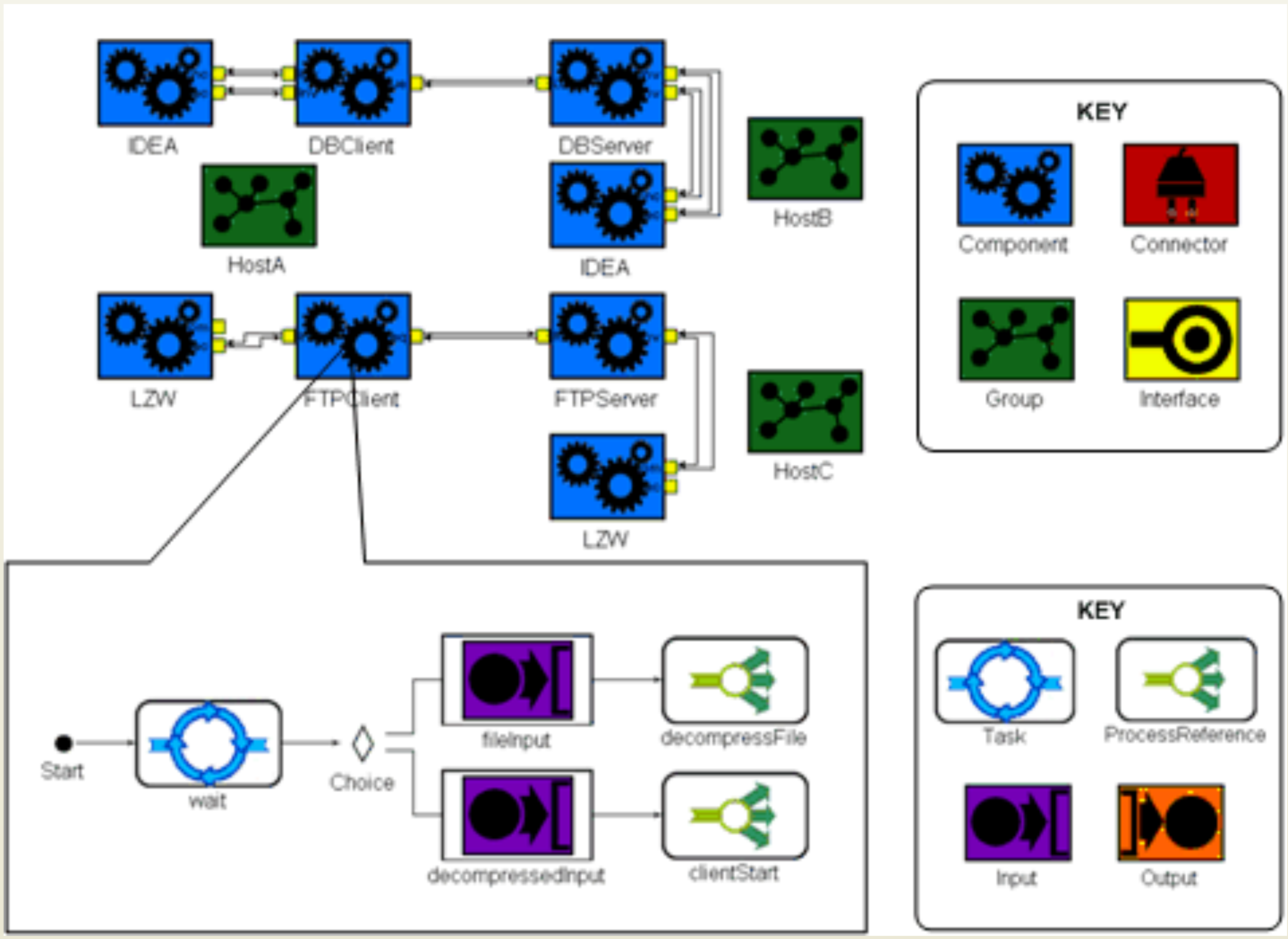
Simulation-Based Analysis in a Nutshell

- *Analysis Goals* – any
- *Analysis Scope* – any
- *Analysis Concern* –behavioral, interaction, and non-functional properties
- *Architectural Models* – formal
- *Analysis Types* – dynamic and scenario-based
- *Automation Level* – fully automated; model mapping may be manual
- *Stakeholders* – any

Example – XTEAM

- eXtensible Tool-chain for Evaluation of Architectural Models
- Targeted at mobile and resource-constrained systems
- Combines two underlying ADLs
  - xADL and FSP
- Maps architectural description to adevs
  - An OTS event simulation engine
- Implements different analyses via ADL extensions and a model interpreter
  - Latency, memory utilization, reliability, energy consumption

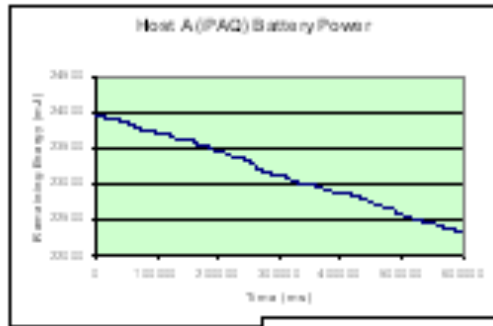
Example XTEAM Model

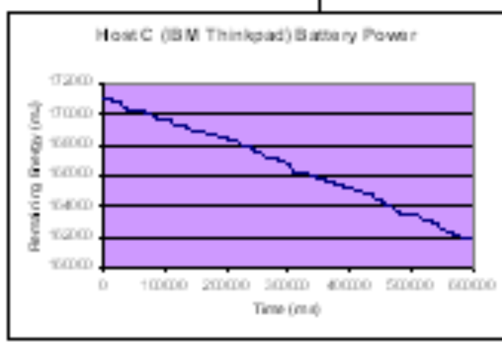


Example XTEAM Analysis

MobileAppPowerUsage6.xls

	A	B	C	D	E	F
1	0	23976	16048	23926.6	38237	23908.1
2	0	23976	16048	23926.6	38237	23889.8
3	3304	23976	16048	23926.5	38238	23889.8
4	3304	23975.9	16048	23926.5	38238	23889.8
5	3304	23975.9	16262	23926.5	38238	23889.7
6	6445	23975.9	16262	23926.5	38238	23889.7
7	6445	23975.9	16262	23926.5	38909	23889.7
8	6445	23975.9	16262	23926.5	38909	23889.7
9	6445	23975.9	25541	23926.5	38909	23889.7
10	7296	23975.9	25541	23926.5	38909	23889.7
11	7296	23975.9	25541	23926.4	44932	23889.7
12	7296	23975.9	25541	23926.4	44932	23889.7
13	7296	23975.9	25772	23926.4	44932	23889.7
14	7297	23975.9	25772	23926.4	44932	23889.7
15	7297	23975.4	25772	23926.4	45751	23889.7
16	7297	23975.4	25772	23926.4	45751	23889.7
17	9432	23975.4	25773	23926.4	45751	23889.7
18	9432	23937.3	25773	23925.5	45751	23889.7
19	9433	23937.3	25773	23925.5	45752	23889.7
20	9433	23937.3	30327	23925.5	45752	23887.5
21	9433	23936.9	30327	23910.8	45752	23887.5





XTEAM in a Nutshell

Goals	Consistency Compatibility Correctness
Scope	Component- and connector-level Subsystem- and system-level Data exchange
Concern	Structural Behavioral Interaction Non-functional
Models	Formal
Type	Dynamic Scenario-based
Automation Level	Automated
Stakeholders	Architects Developers Managers Customers Vendors

Closing Remarks

- Architectural analysis is neither easy nor cheap
- The benefits typically far outweigh the drawbacks
- Early information about the system's key characteristics is indispensable
- Multiple analysis techniques often should be used in concert
- "How much analysis?"
  - This is the key facet of an architect's job
  - Too many will expend resources unnecessarily
  - Too few will carry the risk of propagating defects into the final system
  - Wrong analyses will have both drawbacks

Learning Objectives

- Define architectural analysis and enumerate its goals
- Apply ATAM analysis to software architectures
- Apply Model-Based Analysis to software architecture
- Apply Reliability Analysis to software architecture
- Apply XTEAM Analysis to software architecture