The ATCP 2+9+1 Modeling Framework

Adaptive Team Collaboration, Inc.
800.837.0677 ■ atcprocess.com

March 22, 2005

Bobbi Underbakke
Adaptive Team Collaboration, Inc.

Chris Armstrong
Armstrong Process Group, Inc.

Objectives

- Value of software architecture
- Related efforts
- ATCP modeling framework overview
- Relationship to Service Oriented Architecture (SOA)
- Relationship to Model Driven Architecture (MDA)
- Application of UML collaborations
- ATCP modeling framework views
- Project-level tailoring

Business Value of Architecture

- Increased ability to respond to new demands
- Increased business value from IT operations
- Greater ability to implement new technology
- Faster, cheaper, more simple procurement
- Faster time to market

-TOGAF, Version 7

Good IT Architecture

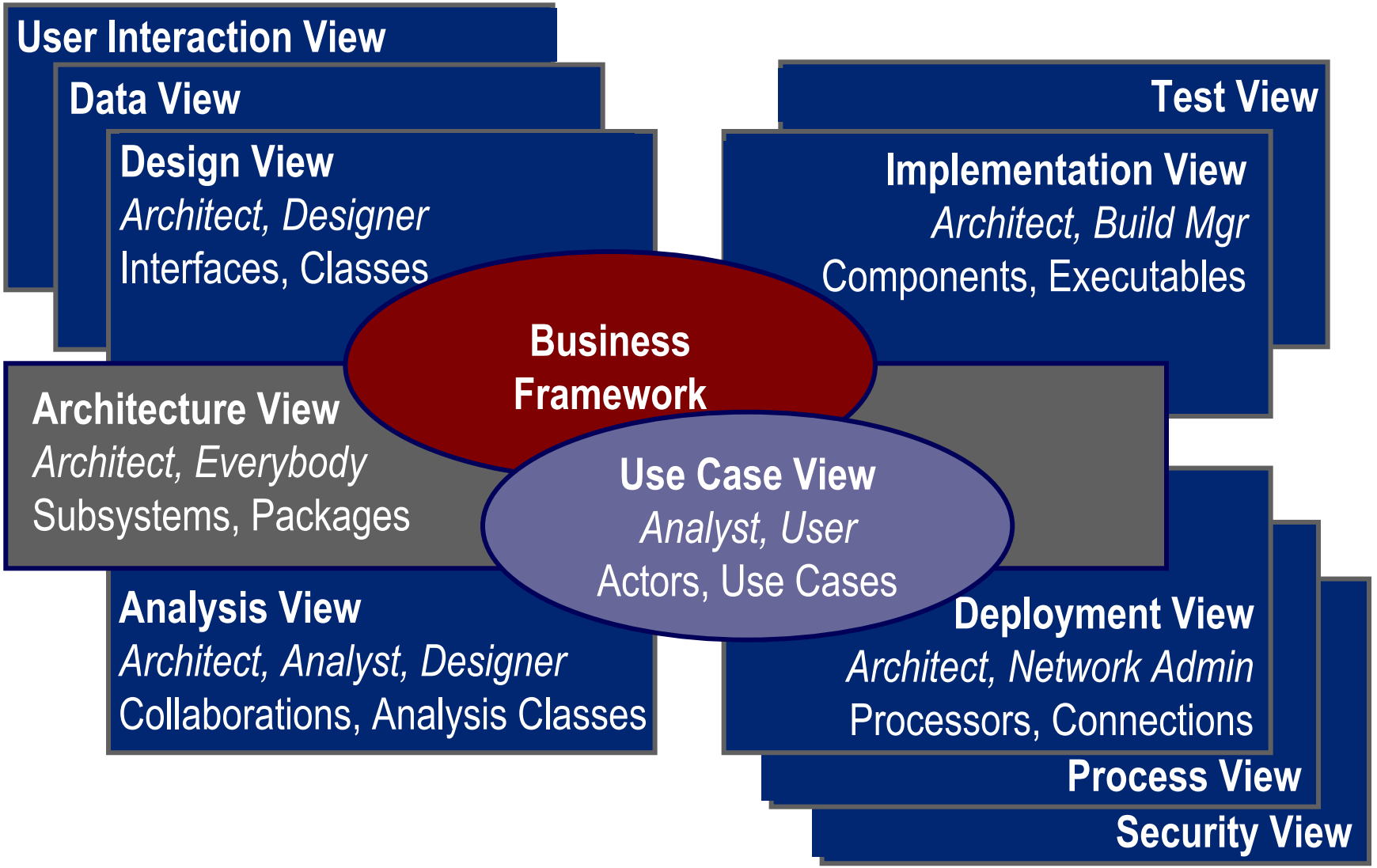
- Directly address enterprise needs
- React to change at the rate dictated by enterprise markets
- Be understood and supported by senior management
- Clearly define structure of existing system
- Provide roadmap and migration strategy for future purchase/developments
- Reduce number and complexity of interfaces between components, improving ease of
  - Application portability
  - Component upgrade and exchange
  - Component development and maintenance

-TOGAF, Version 7

Related Efforts

- 4+1 View Model of Software Architecture
- Federal Enterprise Architecture Framework (FEAF)
- Department of Defense Architecture Framework (DoDAF)
- Treasury Enterprise Architecture Framework (TEAF)
- Open Group Architecture Framework (TOGAF)
- Zachman Framework
- IBM Rational Architecture Description Specification
- IEEE 1471-2000

ATCP Modeling Framework



Service Oriented Architecture (SOA)

- Describes a set of things that provide and consume services
  - Can describe services at any level
  - Current usage is primarily for software application services
- Business, software, and technology services
  - Describes an organization, its operation, and implementation
  - Identifies relationships between functions, capabilities, components, and service flows
- Describe using UML interfaces

ATCP and SOA Principles

- Service-oriented approach produces an adaptive, agile model
  - Creates reusable services
  - Establishes consistent traceability based on flow of services
- ATCP modeling framework
  - Business framework view—organizational services
    - Business strategy, planning, and operation
  - Use case views—service usage
    - Business, system, and subsystem
  - System view—software and technology services
    - Analysis, design, implementation, and deployment
  - Architecture view—relationships, packaging, and interfaces

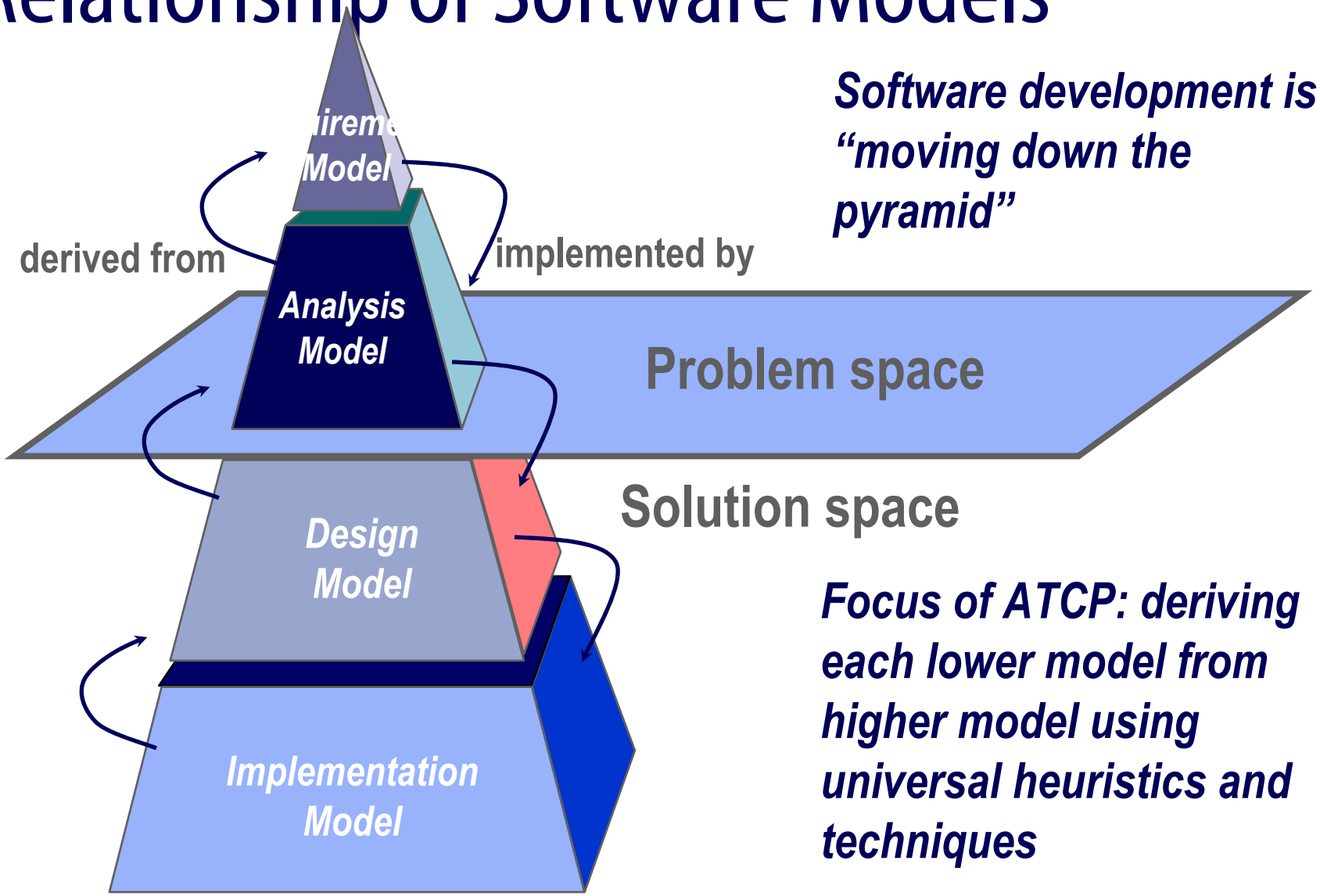
Software and Technology Services

- Extend traditional component-based architecture
- Represent services as subsystems derived from system/subsystem use cases
- Standardization
  - Service consumers
  - Service descriptions
- Implementation example: web services
  - More platform-neutral, but not platform-independent: WSDL, SOAP, UDDI

Managing Complexity

- Many levels of abstraction
- Product development is very complex and complicated
- Complex systems require
  - Good, balanced hierarchies for managing levels of abstraction
  - Different views to address needs of various stakeholders
- Not sufficient to understand using 1- or 2-dimensional viewing and reporting structures

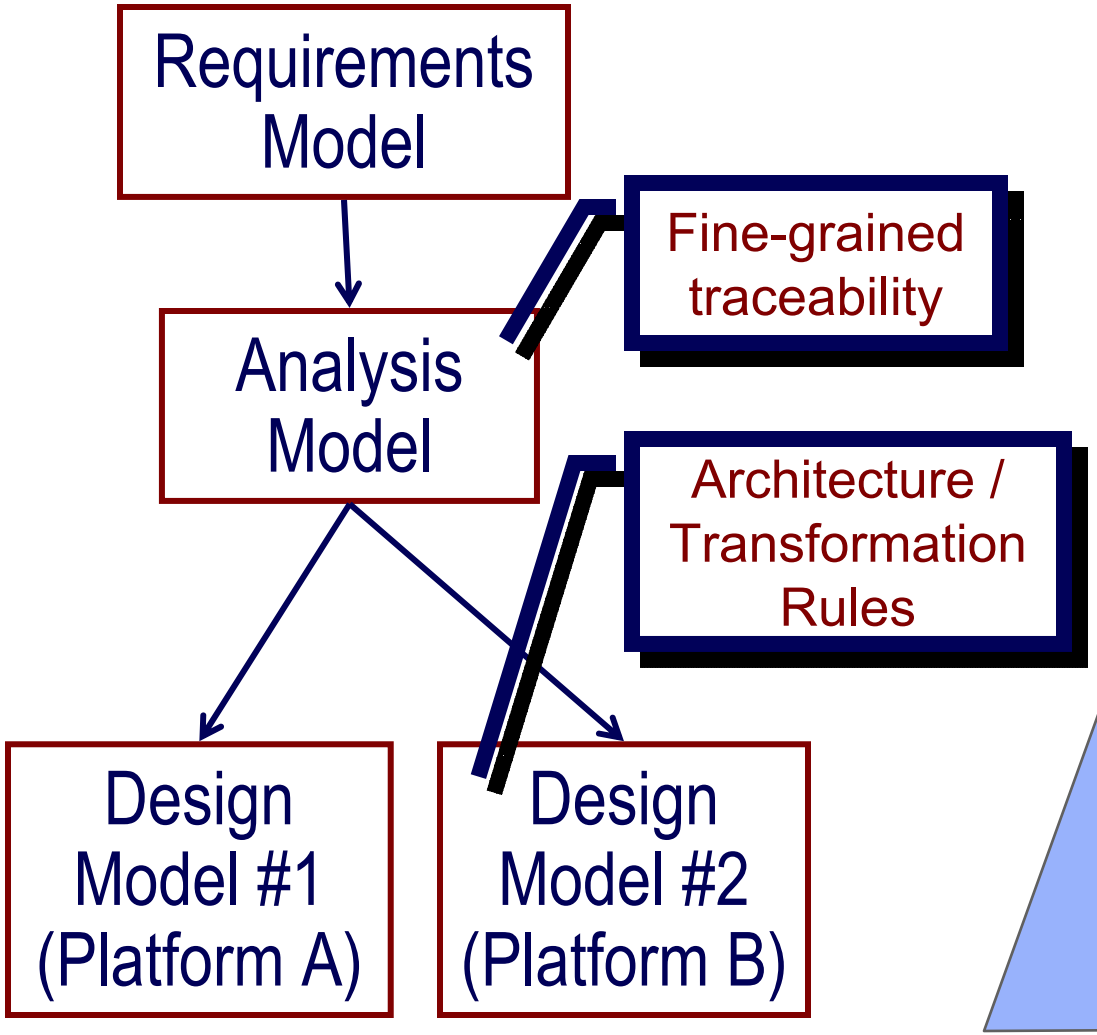
Relationship of Software Models

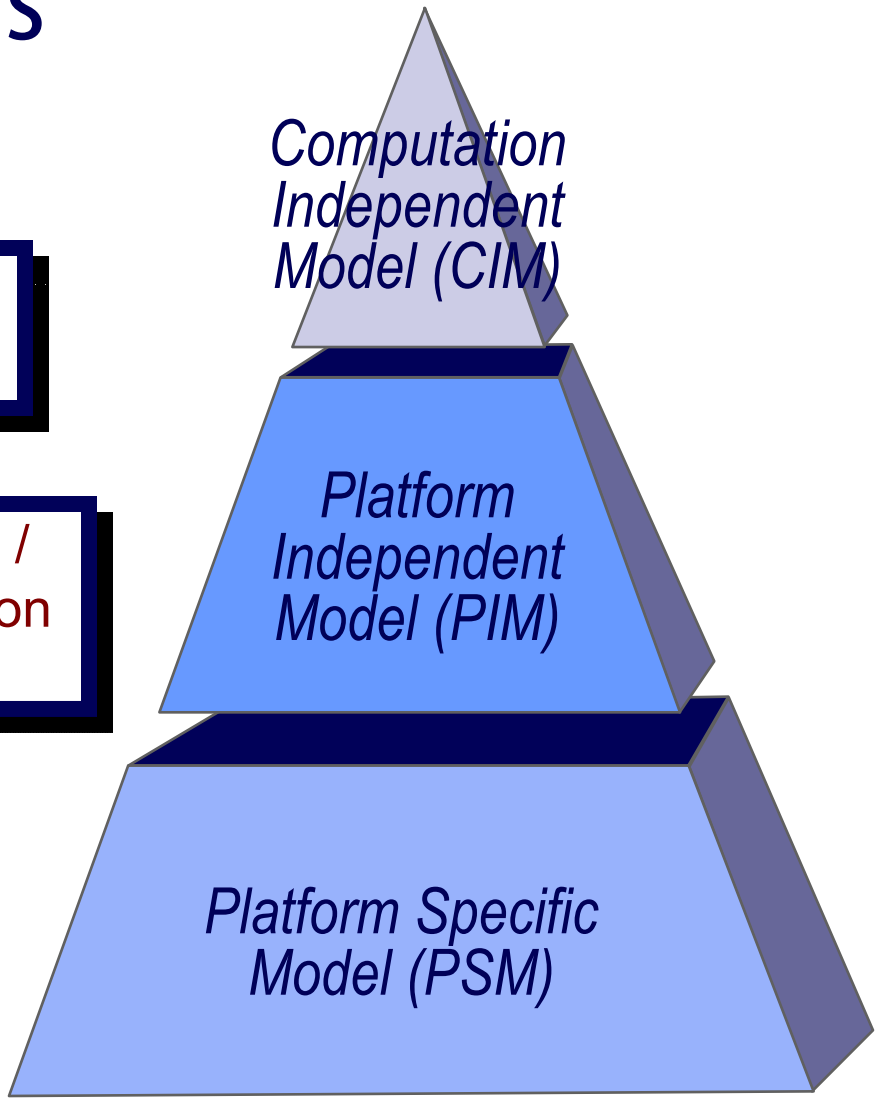


Software development is "moving down the pyramid"

Focus of ATCP: deriving each lower model from higher model using universal heuristics and techniques

ATCP And MDA Models





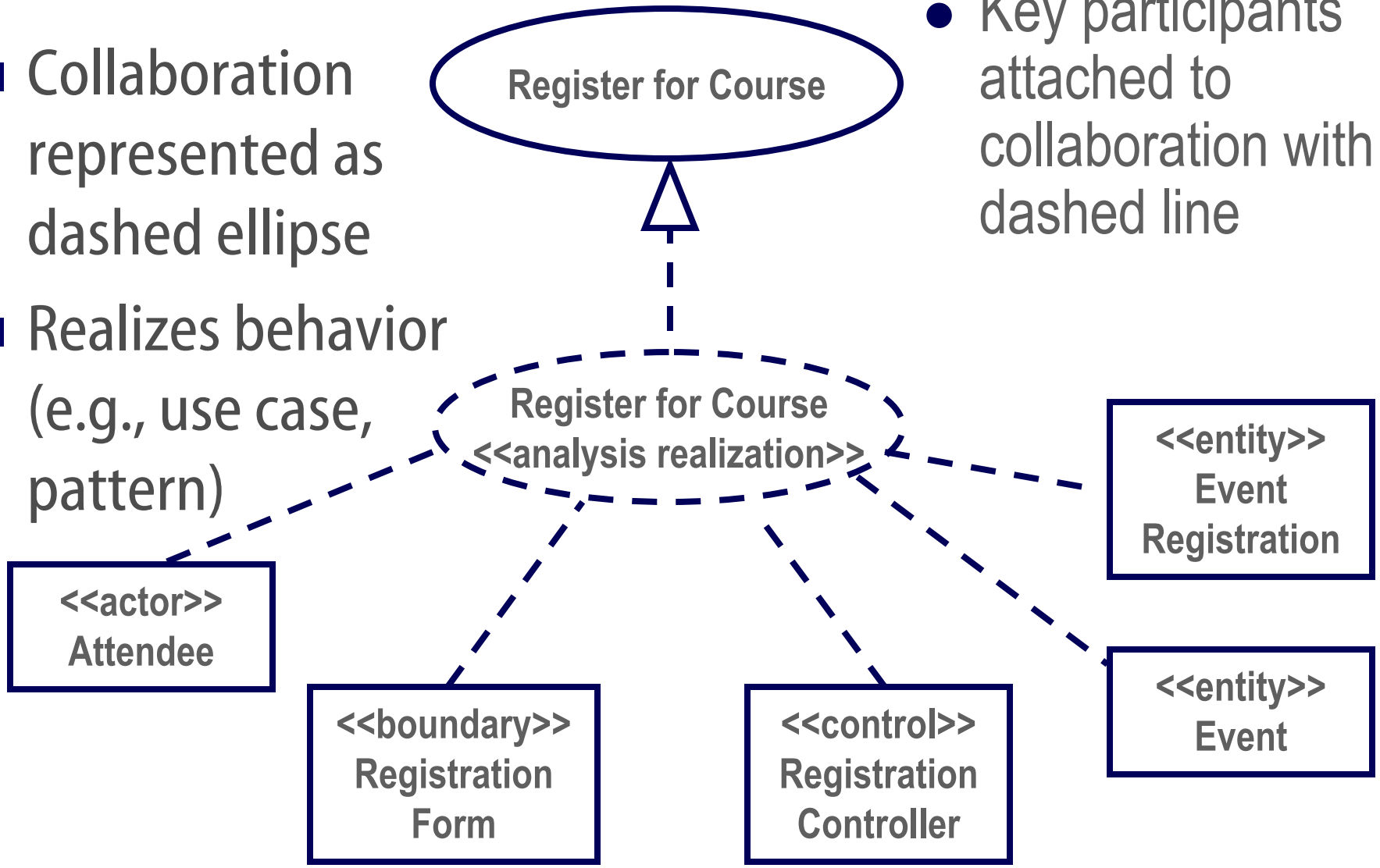
ATCP Universal Design Process Pattern

- Given a set of related behavior's description
  - Find candidate components
  - Describe their services
  - Describe how they interact
  - Group into structures to support their interactions
- Their aggregate behavior constitutes the set of behaviors in question

This way of looking at software can be applied at all levels of abstraction and all degrees of granularity; ATCP applies it to derive one model from another

Collaboration in UML

- Collaboration represented as dashed ellipse
- Realizes behavior (e.g., use case, pattern)



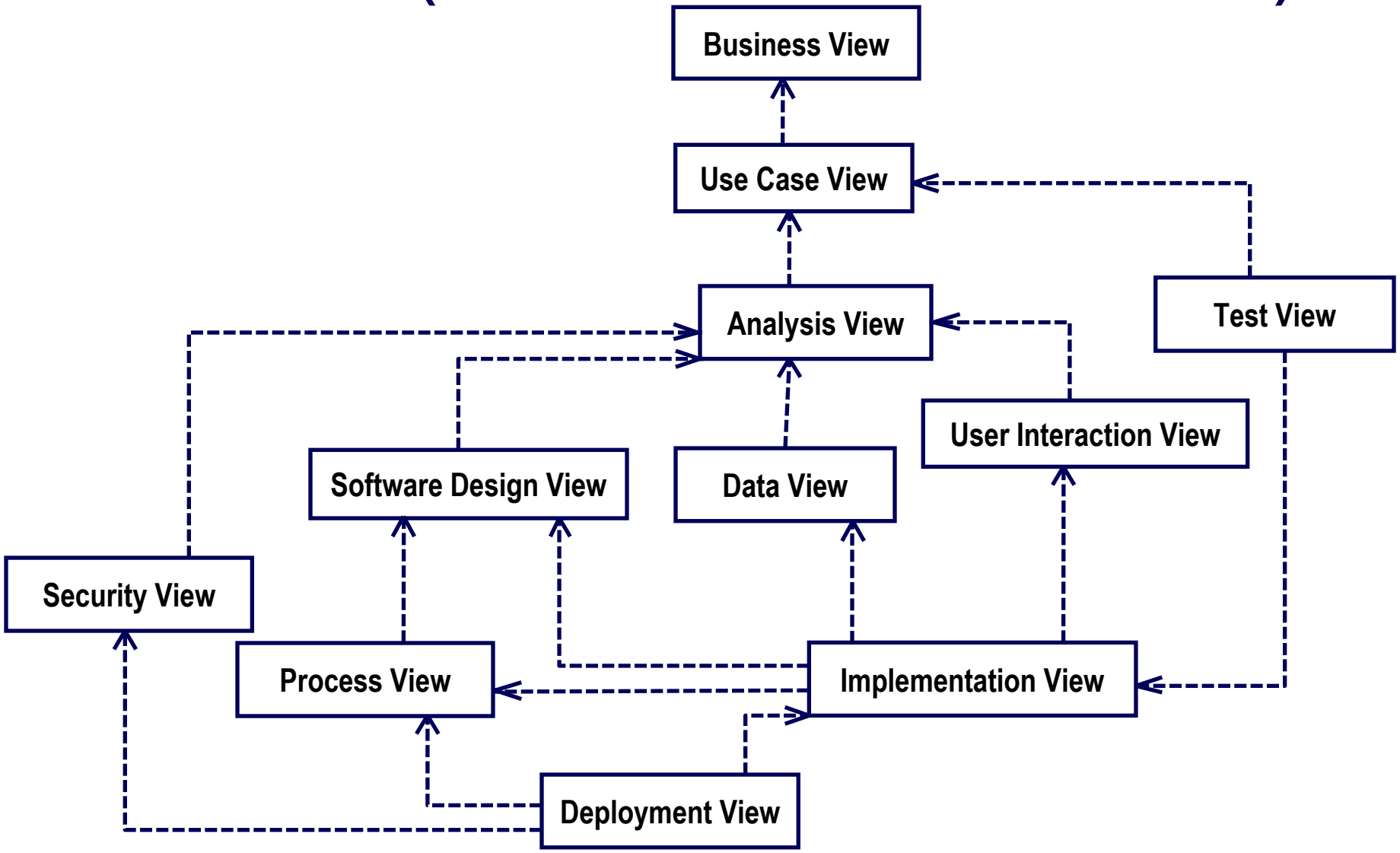
- Key participants attached to collaboration with dashed line

Behavioral and Structural Aspects

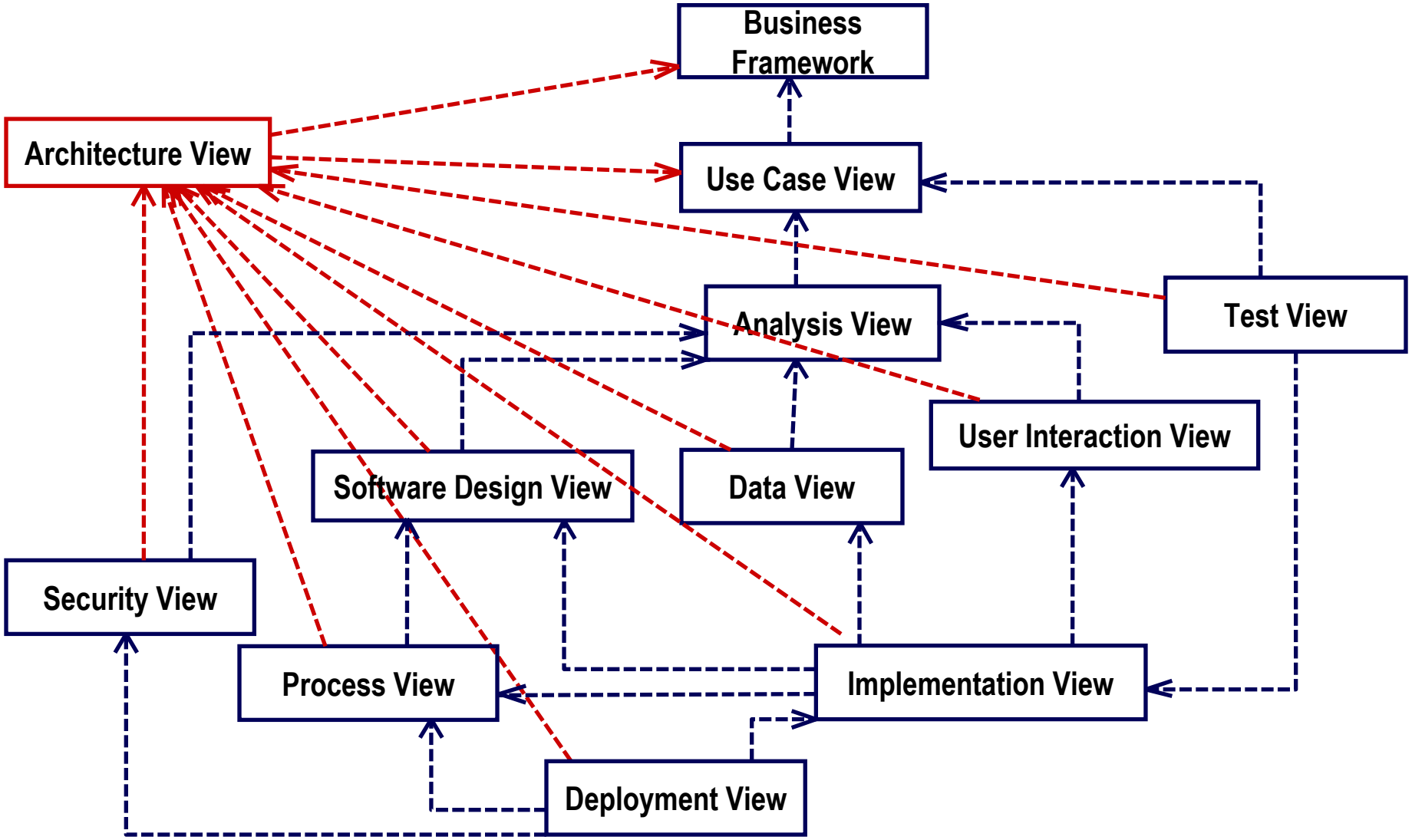
- Shows nature and sequence of interactions between components
- Use interaction diagrams
  - Communication diagram
  - Sequence diagram
- At least one interaction diagram for each collaboration behavior

- Shows structural relationships required to support interactions of participating components/classes
  - Use class diagrams
- At least one participants diagram per collaboration
- Only show elements specific to interactions of the collaboration
  - Classes
  - Relationships
  - Attributes
  - Operations

Framework (without architecture view)



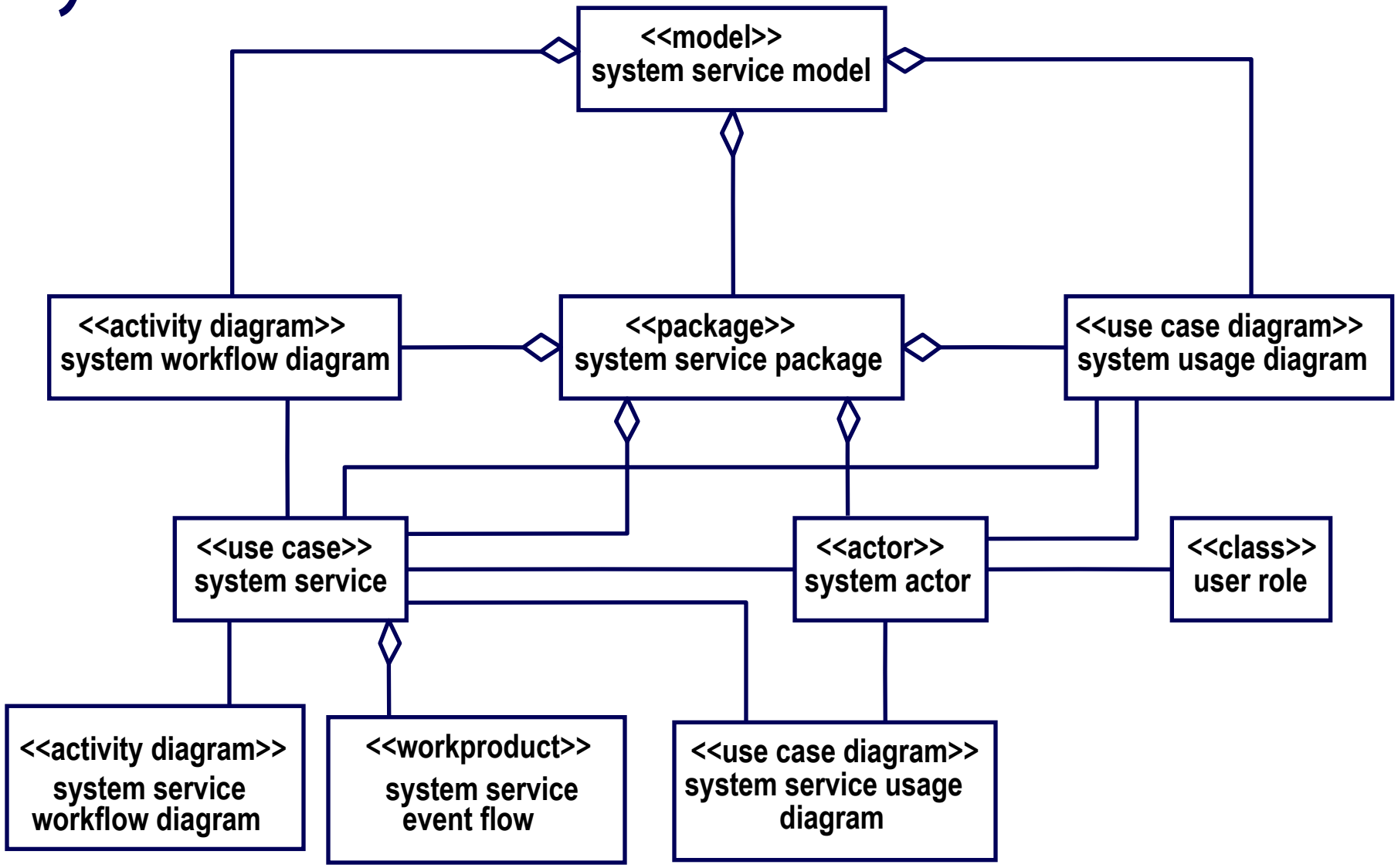
Framework (with architecture view)



Business Framework View

- Concerns
  - Customers
  - External business partners
  - External business services
  - Internal workflows
  - Business rules
- Elements
  - Business actor
  - Business use case
  - Business worker
  - Business entity
  - Business workflow
- Audience
  - Business architect
  - Business designer
  - Sponsor
- The business view can be thought of another level of abstraction (with its own 1+9+1 framework)

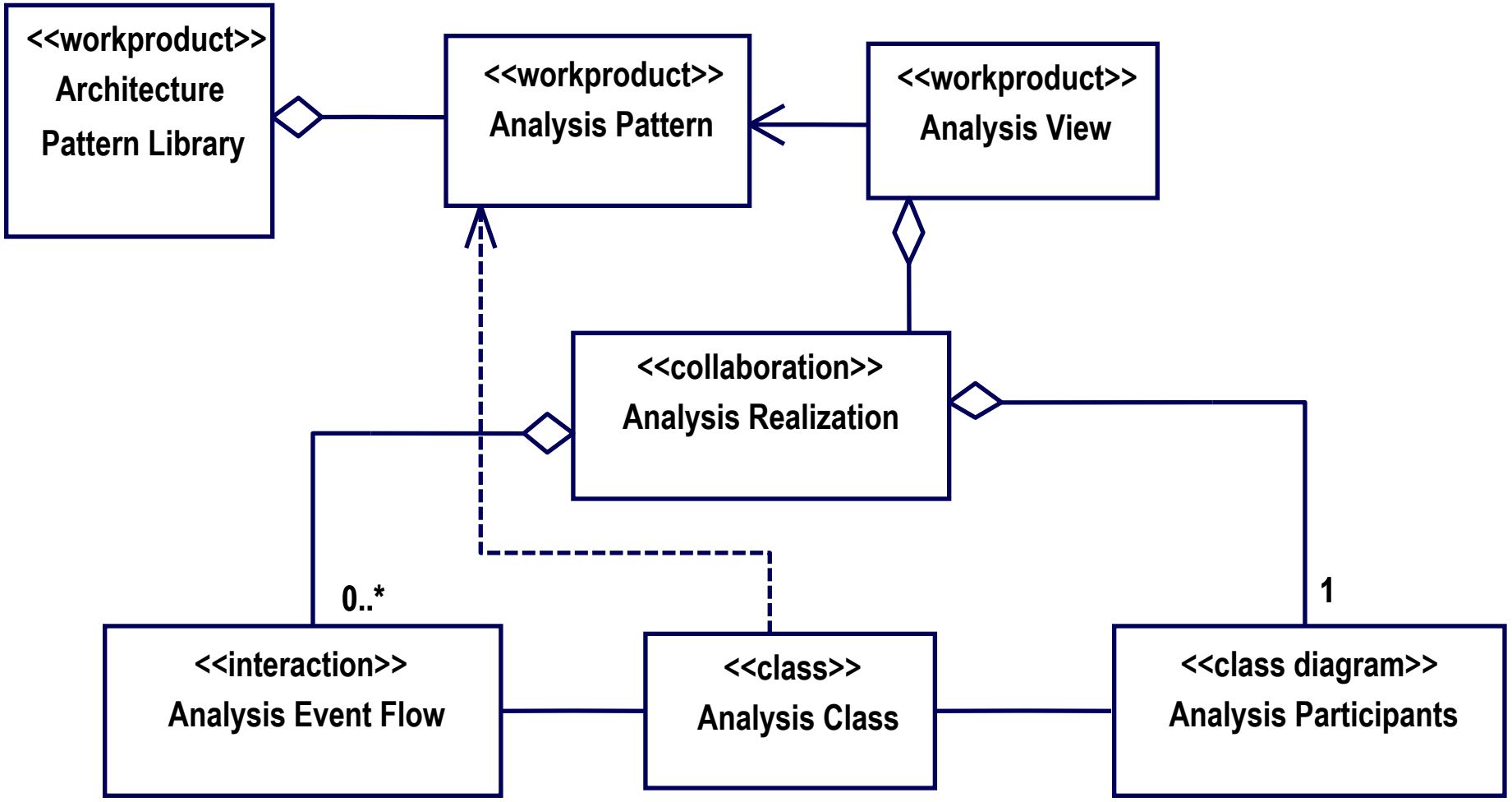
System Use Case View



System Use Case View

- Concerns
  - Users
  - Externally visible application behavior
  - Application requirements
  - External systems integration
- Audience
  - Analyst
  - Customer
  - User
- Elements
  - System use case
  - System actor
  - User role
  - System workflow
  - System use case workflow
- Consider calling system use cases “system services”

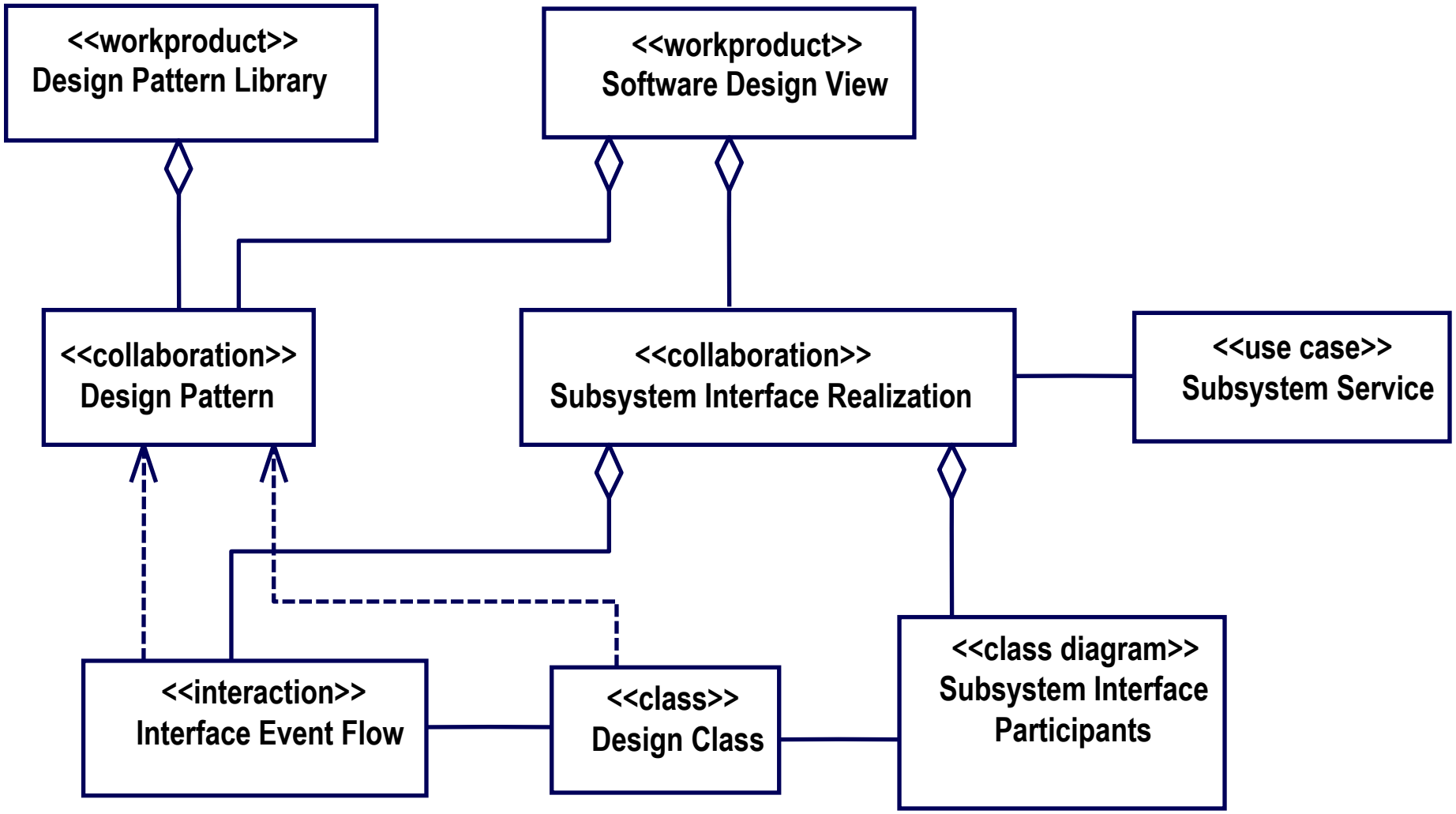
Analysis View



Analysis View

- Concerns
  - Distribution of use case behavior
  - Preliminary, idealized design
  - Platform-independent view
- Audience
  - Analyst
  - Designer
  - Architect
- Elements
  - Analysis pattern
  - Analysis realization
  - Analysis event flow
  - Boundary class
  - Entity class
  - Control class

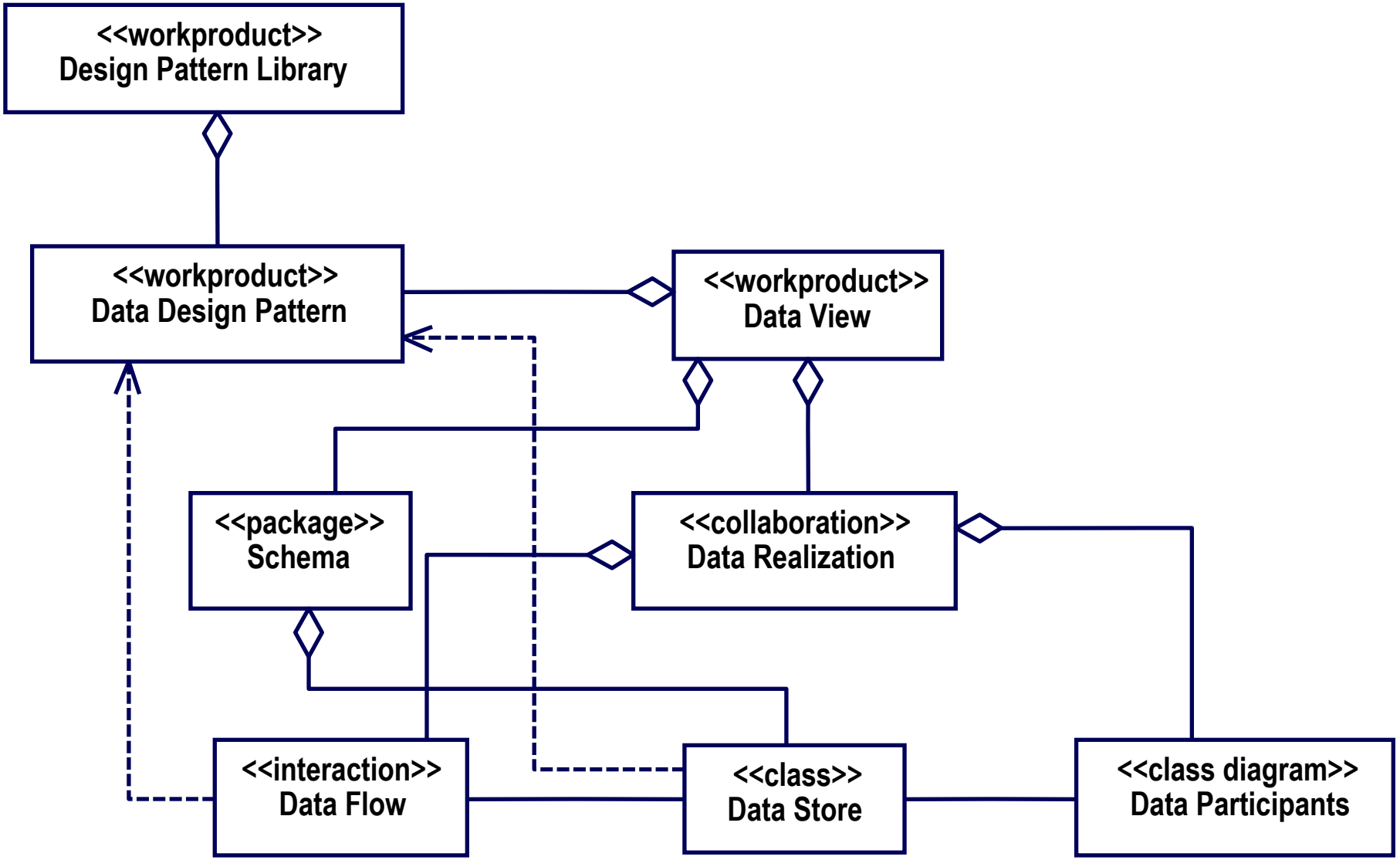
Software Design View



Software Design View

- Concerns
  - Logical design elements
  - Specifications for building software
  - Platform-specific model
  - Influenced by architecture
- Audience
  - Architect
  - Designer
  - Developer
- Elements
  - Subsystem interface realization
  - Subsystem use case
  - Design pattern
  - Interface event flow
  - Design class

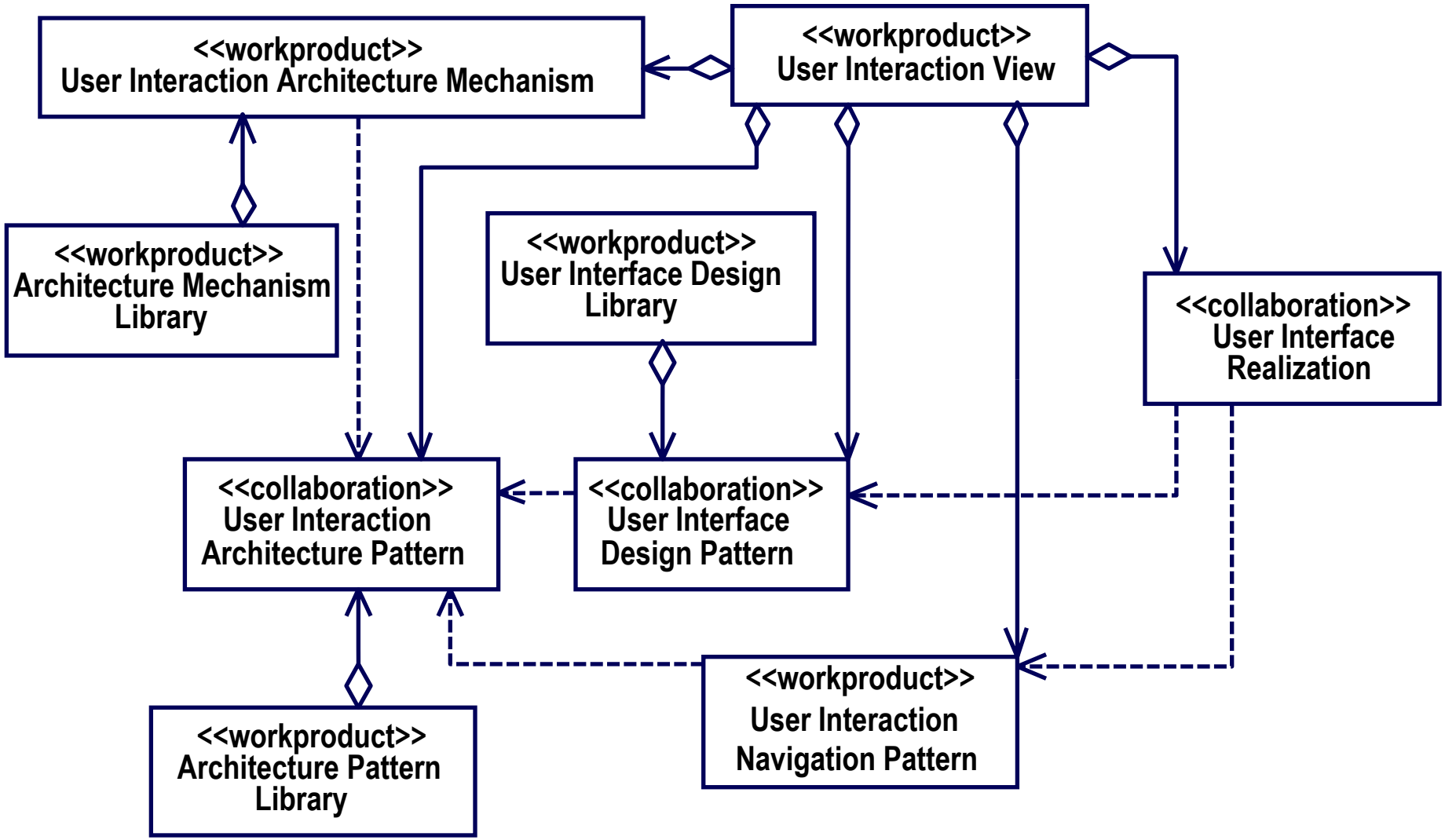
Data View



Data View

- Concerns
  - Physical data structures
  - Performance
  - Normalization
- Audience
  - Architect
  - Data modeler
  - Developer
- Elements
  - Schema
  - Data store
  - Data flow
  - Trigger
  - Stored procedure
  - Constraint

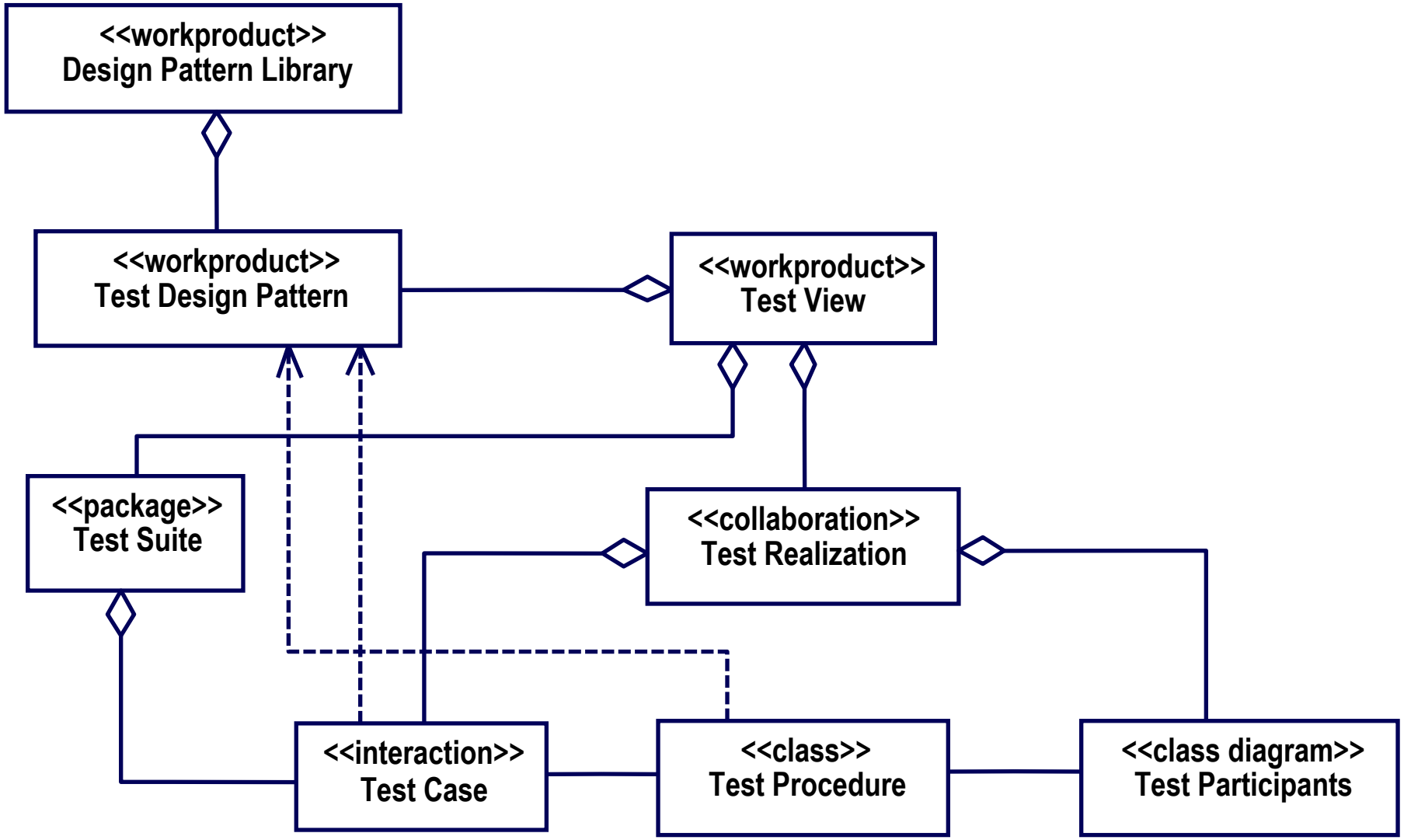
User Interaction View



User Interaction View

- Concerns
  - Human factors
  - Usability
  - Suitability
  - User self-sufficiency
- Audience
  - Architect
  - User
  - Usability engineer
- Elements
  - User interaction architecture pattern
  - User interface navigation pattern
  - User interface realization
  - User interface design pattern

Test View



Test View

- Concerns
  - Verifying application behavior
  - Evaluating traceability consistency
  - Acceptable quality
- Audience
  - User / Customer
  - Test analyst
  - Tester
  - Analyst
  - Developer
- Elements
  - Test realization
  - Test suite
  - Test case
  - Test procedure

Implementation View

- Concerns
  - Physical implementation
  - Integrity of software assets
  - Stable builds
- Audience
  - Architect
  - Developer
  - Release/build manager
  - Configuration manager
- Elements
  - Component
  - Subsystem
  - Executable
  - Script
  - Software asset

Deployment View

- Concerns
  - Distribution
  - Performance
  - Reliability
  - Redundancy
  - Throughput
- Audience
  - Architect
  - System administrator
  - Network engineer
- Elements
  - Device
  - Processor
  - Connector

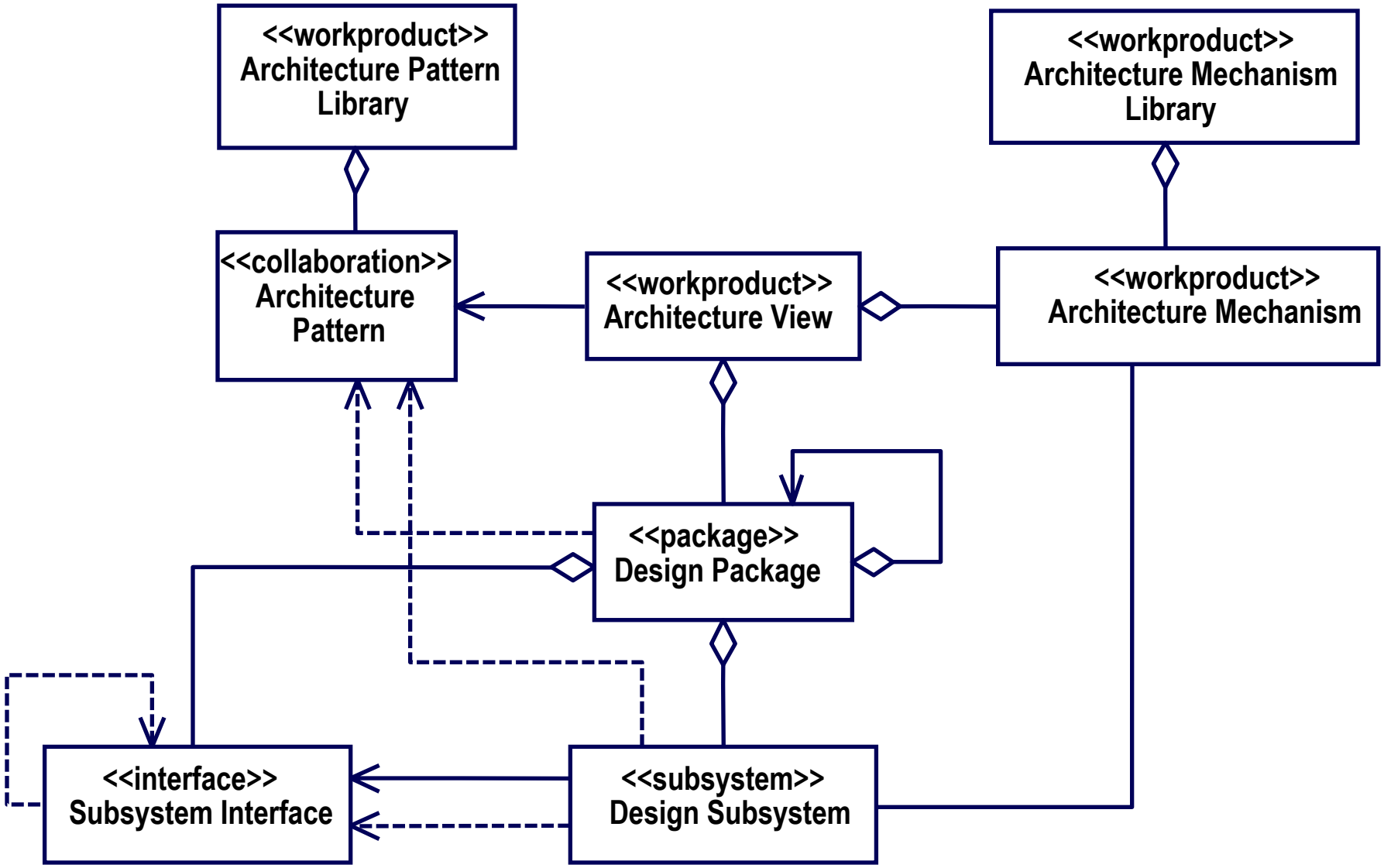
Process View

- Concerns
  - Concurrency
  - Real-time response
  - Performance
  - Resource contention
- Audience
  - Architect
  - Network engineer
  - Designer
  - Developer
- Elements
  - Process
  - Thread
  - Active object

Security View

- Concerns
  - Authentication
  - Certification
  - Integrity
  - Encryption
- Audience
  - Architect
  - System adminstrator
  - Designer
  - Developer
- Elements
  - Roles
  - Privileges
  - Certificates

Architecture View



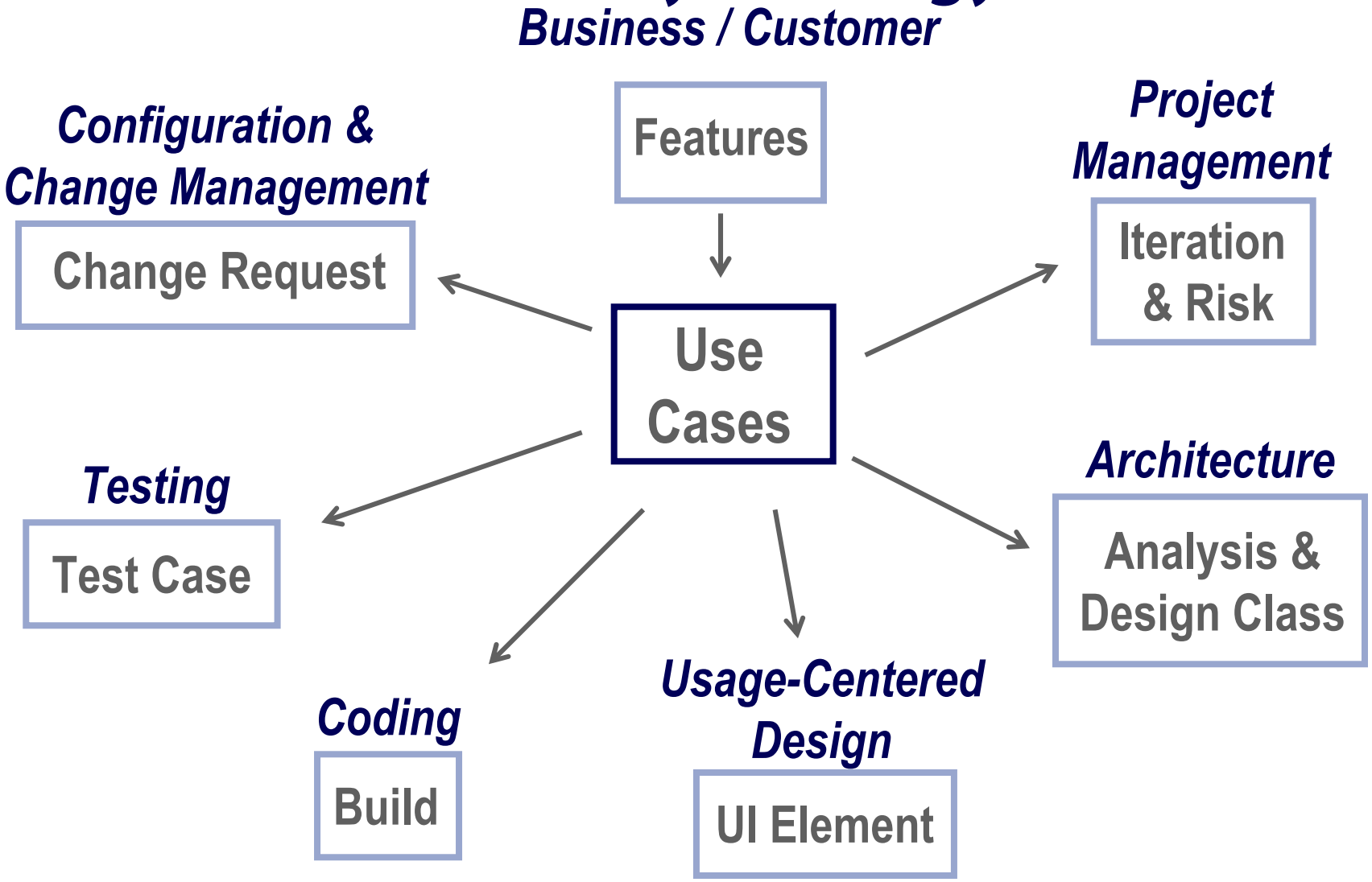
Architecture View

- Concerns
  - Organization and relationships of subsystems and interfaces
  - Platform-specific transformation rules
- Audience
  - Architect
  - Designer
  - Developer
  - Everybody
- Elements
  - Architecture pattern
  - Architecture mechanism
  - Subsystem
  - Interface
  - Design package
- Need to map analysis model to architecture view to generate platform-specific design models

Project Tailoring

- Determine which views provide greatest value
  - Shouldn't model all views because you can
- Determine specific traceability relationships between elements in each view
  - Coarse-grained traceability between collaborations
  - Fine-grained traceability between elements found inside collaborations
- Based on capability of MDA tool set
- Ensure traceability maintenance effort is worthwhile and sustainable

Use Case Traceability Strategy



Thank You

Bobbi Underbakke ■ *Chris Armstrong*