Software Architecture Documentation

Integrating RUP 4+1 Views with Views and Beyond

A Comprehensive Guide to Mapping the Kruchten Model to SEI Documentation Practices

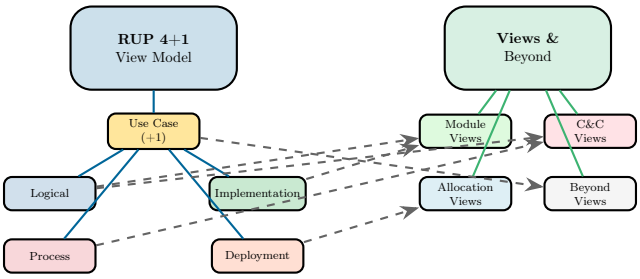
Architecture Documentation Series

Based on Rational Unified Process and SEI Views and Beyond

December 9, 2025

Abstract

The Rational Unified Process (RUP) and its underlying 4+1 View Model, introduced by Philippe Kruchten, has been one of the most influential approaches to software architecture organization. The SEI's Views and Beyond approach provides comprehensive guidance for documenting software architectures using a stakeholder-driven, view-based methodology. This guide provides detailed mappings between the five RUP architectural views (Use Case, Logical, Implementation, Process, and Deployment) and the Views and Beyond view types, covering UML diagram usage, RUP artifacts, phase-specific documentation needs, and practical guidance for creating documentation that satisfies both RUP practitioners and V&B methodologists. Whether you are working in a RUP-based organization or transitioning to modern documentation practices, this guide provides the integration knowledge you need.



4+1 to V&B Integration

Contents

1 Introduction

1.1 Purpose of This Guide

This guide provides comprehensive integration between the Rational Unified Process (RUP) 4+1 View Model and the SEI's Views and Beyond approach. Both frameworks are widely used, and practitioners often need to:

- Translate RUP architectural artifacts to V&B documentation
- Create documentation that satisfies stakeholders familiar with either approach
- Understand how UML diagrams map to V&B view types
- Integrate RUP phase-based development with V&B documentation practices
- Leverage the detailed guidance of V&B within a RUP project

1.2 The 4+1 View Model

Definition

4+1 View Model: An architectural view model introduced by Philippe Kruchten in 1995, adopted by RUP. It organizes architecture description into four fundamental views (Logical, Implementation, Process, Deployment) plus one unifying view (Use Case) that ties the others together through scenarios.

The "+1" (Use Case View) is central because it:

- Provides scenarios that drive and validate the other four views
- Captures architecturally significant use cases
- Serves as input for test cases
- Represents stakeholder requirements

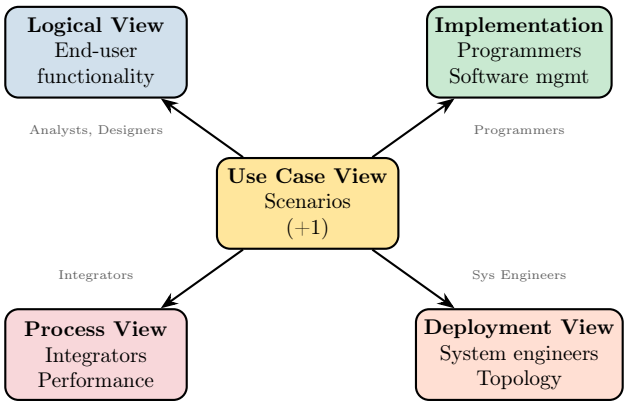


Figure 1: The 4+1 View Model

1.3 Framework Comparison

Table 1: Framework Comparison

Aspect	RUP 4+1	Views and Beyond
Origin	Rational Software (1995)	SEI Research (2002)
View Organization	4+1 fixed views	3 view categories + styles
Notation	UML-centric	Notation-agnostic
Scenarios	Central (+1 view)	Behavior documentation
Process Integration	RUP phases/iterations	Flexible integration
Documentation Unit	UML diagrams	View packets
Flexibility	Prescribed views	Stakeholder-driven selection
Quality Attributes	Implicit in views	Explicit scenarios/tactics

1.4 Core Conceptual Mappings

Table 2: Conceptual Term Mappings

RUP Term	V&B Term	Notes
View	View	Work product showing structure
Use Case	Scenario / Behavior	Behavioral specification
Package	Module	Grouping of elements
Class	Module element	Code-time unit
Component	Component (C&C)	Runtime unit
Node	Environment element	Physical resource
Artifact	Documentation unit	Work product
Subsystem	Module / Component	Context-dependent

2 RUP Overview

2.1 RUP Phases and Architecture

RUP defines four phases with architecture playing different roles in each:

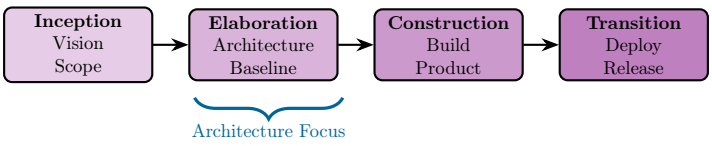


Figure 2: RUP Phases and Architecture Focus

Table 3: Architecture Documentation by RUP Phase

Phase	Architecture Focus	Documentation Needs
Inception	Initial architecture vision	High-level views; Key use cases; Initial deployment concepts
Elaboration	Architecture baseline	Complete 4+1 views; Architectural mechanisms; Prototype validation
Construction	Architecture refinement	Detailed component specs; Interface documentation; Build procedures
Transition	Architecture validation	Deployment documentation; Operations support; Migration guides

2.2 RUP Architectural Artifacts

Table 4: RUP Architectural Artifacts

Artifact	Description	V&B Equivalent
Software Architecture Document (SAD)	Main architecture description	Architecture documentation package
Use Case Model	Use case diagrams and specifications	Behavior documentation; Scenarios
Design Model	Class and interaction diagrams	Module views; C&C views
Implementation Model	Component organization	Module decomposition view
Deployment Model	Node and artifact mapping	Deployment view
Data Model	Persistent data structure	Data model view
Analysis Model	Conceptual model	Context; High-level views

2.3 UML Diagrams in RUP

RUP relies heavily on UML for architectural representation:

Table 5: UML Diagrams by RUP View

RUP View	Primary UML Diagrams	Secondary Diagrams	V&B View Type
Use Case	Use Case Diagram	Activity Diagram	Beyond Views
Logical	Class Diagram; Package Diagram	Object Diagram; Sequence Diagram	Module; C&C

Implementation	Component Diagram; Package Diagram	Class Diagram	Module
Process	Activity Diagram; Sequence Diagram; State Machine	Communication Diagram	C&C
Deployment	Deployment Diagram	Component Diagram	Allocation

3 Use Case View Mapping

RUP Use Case View

Purpose: Contains architecturally significant use cases and scenarios that drive and validate the architecture. The "+1" that ties all other views together.

Concerns Addressed:

- System functionality from user perspective
- Architecturally significant requirements
- Scenario-based validation
- Stakeholder communication

UML Diagrams:

- Use Case Diagrams (actors, use cases, relationships)
- Activity Diagrams (workflow for complex use cases)
- Sequence Diagrams (scenario realizations)

Stakeholders: All stakeholders; particularly customers, users, analysts

Views and Beyond Equivalent

Primary Mapping: Documentation Beyond Views + Behavior Documentation
Use cases in V&B are not a separate view but serve as:

- **Behavior Documentation:** Use cases specify behavior and can be associated with any view where the behavior is realized
- **Quality Attribute Scenarios:** Architecturally significant use cases map to quality attribute scenarios
- **System Overview:** High-level use cases appear in documentation beyond views
- **View Rationale:** Use cases justify architectural decisions

Key Insight: V&B treats scenarios/use cases as cross-cutting documentation that illuminates views rather than as a separate view type.

Implementation Mapping for Use Case View

RUP Use Case Diagram → V&B Context + Behavior
- Actors → External entities in context diagram
- Use cases → Behavior documentation; Quality scenarios
- System boundary → Context diagram boundary

RUP Use Case Specifications → V&B Behavior Documentation
- Main flow → Behavior traces
- Alternative flows → Variability in behavior
- Pre/post conditions → Scenario constraints

RUP Sequence Diagrams (Realizations) → V&B View Behavior
- Scenario realizations → Behavior documentation within C&C view
- Component interactions → Sequence/trace notation

Architecturally Significant Use Cases → V&B Quality Attribute Scenarios
- Performance-critical scenarios → Performance QA scenarios
- Security scenarios → Security QA scenarios
- Failure scenarios → Availability QA scenarios

Example

Use Case to V&B Mapping Example
RUP Architecturally Significant Use Case:

> UC-042: Process High-Volume Order Batch
> Actor: Batch Scheduler
> Precondition: Order queue contains 10,000+ orders
> Main Flow: System processes all orders within 30 minutes maintaining data integrity
> Quality: Performance-critical; Availability-critical

V&B Documentation:
Quality Attribute Scenario (Performance):
- Source: Batch Scheduler
- Stimulus: 10,000 orders submitted
- Artifact: Order Processing System
- Environment: Normal batch window
- Response: Process all orders
- Measure: Complete within 30 minutes

C&C View Behavior Documentation: Sequence diagram showing OrderProcessor, ValidationService, InventoryService, and FulfillmentService interactions during batch processing.
Rationale Reference: "Decision to use asynchronous processing with Kafka (ADR-007) directly supports UC-042 batch processing requirements."

4 Logical View Mapping

RUP Logical View

Purpose: Describes the system's key abstractions and mechanisms, primarily through classes, packages, and their relationships. Shows the logical organization that supports functionality.

Concerns Addressed:

- Functional requirements realization
- Key abstractions and their responsibilities
- Inheritance and composition relationships
- Package organization
- Design patterns and mechanisms

UML Diagrams:

- Class Diagrams (classes, interfaces, relationships)
- Package Diagrams (logical groupings)
- Object Diagrams (runtime instances)
- Interaction Diagrams (collaborations)

Stakeholders: Designers, developers, analysts

Views and Beyond Equivalent

Primary Mapping: Module Views (structural) + C&C Views (runtime)
The Logical View maps to multiple V&B view types:

For Structural/Static Aspects:

- **Decomposition Style:** Package hierarchy and module organization
- **Generalization Style:** Inheritance relationships
- **Uses Style:** Dependencies between modules
- **Data Model Style:** Persistent class structures

For Runtime/Dynamic Aspects:

- **Client-Server C&C:** Request-response interactions
- **Object-Oriented C&C:** Object collaborations
- **Shared-Data C&C:** Repository interactions

Key Insight: RUP's Logical View conflates code-time structure (Module) with runtime behavior (C&C). V&B separates these concerns.

Implementation Mapping for Logical View

RUP Class Diagram (Structure) → V&B Module Views
- Classes → Modules in decomposition
- Interfaces → Module interfaces
- Inheritance → Generalization view (is-a relations)
- Dependencies → Uses view (allowed-to-use relations)
- Associations → Module relations in element catalog

RUP Package Diagram → V&B Decomposition View
- Packages → Modules (parent modules)
- Package dependencies → Uses relations
- Package hierarchy → Decomposition hierarchy

RUP Interaction Diagrams → V&B C&C View + Behavior
- Objects → Component instances
- Messages → Connector interactions
- Sequence → Behavior documentation (traces)

RUP Design Patterns/Mechanisms → V&B Rationale
- Architectural mechanisms → Documented in rationale
- Pattern applications → Decision records

Warning

Common Mapping Pitfall
RUP's Logical View often mixes static class structure with dynamic object interactions in a single view. When mapping to V&B:

1. **Separate structural from behavioral:** Class relationships go in Module views; object interactions go in C&C views with behavior documentation
2. **Distinguish compile-time from runtime:** Class diagrams showing code structure are Module; diagrams showing runtime objects are C&C
3. **Be explicit about instances:** If showing specific runtime instances, use C&C; if showing types/classes, use Module

5 Implementation View Mapping

RUP Implementation View

Purpose: Describes the organization of static software elements (code, data, components) in the development environment. Shows how design elements map to implementation artifacts.
Concerns Addressed:
- Source code organization
- Component packaging
- Build dependencies
- Configuration management
- Layer organization

UML Diagrams:

- Component Diagrams (software components)
- Package Diagrams (source organization)
- Class Diagrams (implementation classes)

Stakeholders: Programmers, software managers, build engineers

Views and Beyond Equivalent

Primary Mapping: Module Views + Implementation Allocation View

- **Decomposition Style:** Module hierarchy showing component organization
- **Layered Style:** Layer structure with allowed dependencies
- **Uses Style:** Build-time dependencies
- **Implementation View (Allocation):** Mapping to source files and directories

Key Distinction: RUP's "Implementation View" focuses on component packaging. V&B's "Implementation View" (allocation style) specifically maps modules to file system. Both address developer concerns but with different scope.

Implementation Mapping for Implementation View

RUP Component Diagram → V&B Module Views
- Components → Modules in decomposition
- Component interfaces → Module interfaces
- Component dependencies → Uses relations
- Provided/Required interfaces → Interface documentation

RUP Source Organization → V&B Implementation View (Allocation)
- Source directories → File system structure
- Module-to-file mapping → Implementation allocation
- Build artifacts → Build output mapping

RUP Layer Structure → V&B Layered View
- Layers → Layers in layered view
- Layer rules → Allowed-to-use constraints
- Layer contents → Layer decomposition

RUP Subsystems → V&B Module Decomposition
- Subsystems → Top-level modules
- Subsystem interfaces → Module public interfaces
- Internal components → Child modules

Example

Implementation View Mapping Example
RUP Component Diagram:

```
<<component>> OrderManagement
  - provides: IOrderService
  - requires: IInventory, IPayment
  - contains: OrderValidator, OrderProcessor, OrderRepository
```

V&B Module Decomposition:
Element Catalog Entry:
Module: OrderManagement
Responsibilities: Order lifecycle management; validation; persistence
Interfaces:
• IOrderService (provided): Order CRUD operations
Children:
• OrderValidator: Input validation logic
• OrderProcessor: Business logic orchestration
• OrderRepository: Data access layer
Uses Relations:
• Uses InventoryManagement.IInventory
• Uses PaymentProcessing.IPayment
V&B Implementation View (Allocation):

```
OrderManagement -> /src/main/java/com/acme/orders/
  OrderValidator -> OrderValidator.java
  OrderProcessor -> OrderProcessor.java
  OrderRepository -> OrderRepository.java
  IOrderService -> interfaces/IOrderService.java
```

6 Process View Mapping

RUP Process View

Purpose: Describes the system's concurrency and synchronization aspects. Shows processes, threads, and their interactions to address performance, scalability, and throughput.
Concerns Addressed:
• Concurrency and parallelism
• Process/thread structure
• Synchronization and locking
• Inter-process communication
• Performance and scalability
• Fault tolerance
UML Diagrams:
• Activity Diagrams (parallel flows, synchronization)
• Sequence Diagrams (process interactions)

- State Machine Diagrams (process states)
- Communication Diagrams (process topology)

Stakeholders: System integrators, performance engineers, developers

Views and Beyond Equivalent

Primary Mapping: Communicating-Processes C&C Style

- **Communicating-Processes Style:** Primary style for process/thread structure
- **Pipe-and-Filter Style:** For data processing pipelines
- **Shared-Data Style:** For shared memory communication
- **Client-Server Style:** For request-handling processes

C&C View Elements:

- Components: Processes, threads, tasks
- Connectors: IPC mechanisms, synchronization primitives, message queues
- Properties: Thread safety, concurrency constraints, scheduling

Behavior Documentation: State machines and interaction traces document process behavior.

Implementation Mapping for Process View

RUP Process Structure → V&B C&C Communicating-Processes

- Processes → Process components
- Threads → Thread components (or component properties)
- Process groups → Component groupings

RUP IPC Mechanisms → V&B Connectors

- Message passing → Message connector type
- Shared memory → Shared-data connector
- RPC/RMI → Call-return connector
- Events/Signals → Event connector

RUP Synchronization → V&B Connector Properties

- Locks/Mutexes → Synchronization properties
- Semaphores → Concurrency constraints
- Barriers → Synchronization points

RUP Activity Diagrams (Concurrent) → V&B Behavior Documentation

- Parallel flows → Concurrent trace notation
- Join/Fork → Synchronization in behavior
- Swimlanes → Process allocation

RUP State Machines → V&B Behavior Documentation

- Process states → State diagrams in C&C view
- State transitions → State machine notation

7 Deployment View Mapping

RUP Deployment View

Purpose: Describes the mapping of software to hardware, showing nodes, their connections, and the software artifacts deployed on each node.

Concerns Addressed:

- Hardware topology
- Software-to-hardware mapping
- Network configuration
- Distribution and communication
- System installation

UML Diagrams:

- Deployment Diagrams (nodes, artifacts, connections)
- Component Diagrams (deployed components)

Stakeholders: System engineers, operators, network administrators

Views and Beyond Equivalent

Primary Mapping: Deployment Style (Allocation Views)

- **Deployment View:** Primary allocation view for hardware mapping
- **Install View:** For installation structure (complements deployment)

Deployment View Elements:

- Software elements: Components from C&C views
- Environmental elements: Servers, containers, VMs, network zones
- Allocation relation: "Deployed-on" mapping

The V&B Deployment view closely matches RUP's Deployment view, making this one of the most direct mappings.

Implementation Mapping for Deployment View

RUP Deployment Diagram → V&B Deployment View
- Nodes → Environmental elements (hosts)
- Artifacts → Software elements
- Deployed artifacts → Allocation relations
- Communication paths → Network connections

RUP Node Properties → V&B Element Catalog
- Hardware specs → Node properties
- OS/middleware → Environment properties
- Capacity → Resource properties

RUP Network Topology → V&B Deployment View
- Network connections → Communication paths
- Protocols → Connection properties
- Network zones → Environmental regions

RUP Artifacts → V&B Software Elements
- Executables → Deployable units
- Libraries → Shared components
- Configuration files → Configuration elements

Example

Deployment View Mapping Example
RUP Deployment Diagram:

```
<<device>> WebServer
  <<artifact>> webapp.war
  <<artifact>> nginx.conf

<<device>> AppServer [3 instances]
  <<artifact>> orderservice.jar
  <<artifact>> inventoryservice.jar

<<device>> DatabaseServer
  <<artifact>> postgresql

WebServer --HTTP--> AppServer --JDBC--> DatabaseServer
```

V&B Deployment View Element Catalog:
Environmental Elements:
- **WebServer:** Nginx reverse proxy; 4 CPU, 8GB RAM; DMZ zone
- **AppServer:** Java application server; 8 CPU, 32GB RAM; Application zone; 3 instances for HA
- **DatabaseServer:** PostgreSQL 15; 16 CPU, 128GB RAM; Data zone; Primary + standby

Allocation Relations:
- WebServer hosts: webapp.war (frontend), nginx.conf (routing rules)
- AppServer hosts: OrderService, InventoryService, PaymentService
- DatabaseServer hosts: PostgreSQL (orders, inventory, payments schemas)

Communication Paths:
- WebServer → AppServer: HTTPS/443, load-balanced round-robin
- AppServer → DatabaseServer: JDBC/5432, connection pooled (max 50)

8 UML Notation Mapping

8.1 UML Diagrams to V&B View Types

Table 6: UML Diagram to V&B View Type Mapping

UML Diagram	RUP View	V&B View Type	Usage Notes
Use Case Diagram	Use Case	Beyond Views; Context	System boundary; actors
Class Diagram	Logical	Module Views	Static structure
Package Diagram	Logical; Impl	Module Decomposition	Module hierarchy
Object Diagram	Logical	C&C View	Runtime instances

UML Diagram	RUP View	V&B View Type	Usage Notes
Sequence Diagram	Use Case; Logical; Process	Behavior Documentation	Interaction traces
Communication Diagram	Logical; Process	C&C + Behavior	Object/process topology
Activity Diagram	Use Case; Process	Behavior Documentation	Workflows; concurrency
State Machine	Logical; Process	Behavior Documentation	Element states
Component Diagram	Implementation	Module Views	Component organization
Deployment Diagram	Deployment	Deployment View (Allocation)	Hardware mapping
Composite Structure	Logical	C&C View	Internal structure

8.2 Notation Considerations

UML in V&B Documentation

V&B is notation-agnostic, but UML is a natural fit. When using UML:

For Module Views:

- Use Class Diagrams with "is-part-of" for decomposition
- Use Class Diagrams with generalization for generalization view
- Use Package Diagrams for high-level decomposition
- Annotate with `«module»` stereotype if needed

For C&C Views:

- Use Component Diagrams with `«component»` stereotype
- Use Composite Structure for internal C&C
- Clearly distinguish components from modules
- Show ports and connectors explicitly

For Allocation Views:

- Use Deployment Diagrams directly
- Show artifacts deployed on nodes
- Include communication paths with protocols

For Behavior:

- Use Sequence Diagrams for traces
- Use State Machines for stateful components
- Use Activity Diagrams for workflows

9 Complete Mapping Reference

Table 7: Complete RUP to V&B Mapping Reference

RUP Element	V&B Equivalent	Implementation Notes
Use Case View (+1)		
Use cases	Behavior documentation	Associate with views
Actors	Context diagram entities	External actors
Scenarios	Quality attribute scenarios	Architecturally significant
Realizations	Behavior traces	In C&C views
Logical View		
Classes (structural)	Module decomposition	Code-time structure
Classes (runtime)	C&C components	Runtime instances
Packages	Module hierarchy	Decomposition view
Inheritance	Generalization view	Is-a relations
Associations	Module relations	Element catalog
Interactions	C&C behavior	Sequence/trace
Patterns	Rationale	Decision documentation
Implementation View		
Components	Modules	Decomposition view
Layers	Layered view	With constraints
Source structure	Implementation view	Allocation style
Dependencies	Uses view	Allowed-to-use
Subsystems	Top-level modules	With interfaces
Process View		
Processes	C&C components	Communicating-processes
Threads	Component properties	Or sub-components
IPC	Connectors	Typed connectors
	Connector properties	Concurrency constraints
Synchronization		
Process states	Behavior documentation	State machines
Deployment View		
Nodes	Environmental elements	Deployment view
Artifacts	Software elements	Deployable units
Deployment	Allocation relations	Deployed-on
Network	Communication paths	With properties

10 Implementation Guidance

10.1 Mapping Strategy

Best Practice

RUP to V&B Migration Strategy:

1. **Inventory existing artifacts:** List all RUP architectural artifacts (SAD sections, UML models)
2. **Map to V&B views:** Use this guide to identify V&B view types for each RUP artifact
3. **Separate concerns:** Split RUP Logical View into Module (structural) and C&C (runtime) views
4. **Extract behavior:** Move use case realizations and scenarios to behavior documentation
5. **Add V&B structure:** Create element catalogs, rationale, and variability guides
6. **Create view packets:** Package views with all V&B documentation components

10.2 Documentation Structure

Table 8: RUP SAD Section to V&B Mapping

RUP SAD Section	V&B Equivalent	Notes
Introduction	Documentation Roadmap	Purpose, scope, definitions
Architectural Goals	Rationale; Quality Scenarios	Driving requirements
Use Case View	Beyond Views; Behavior	Cross-cutting scenarios
Logical View	Module Views; C&C Views	Split static/dynamic
Implementation View	Module Views; Allocation	Code organization
Process View	C&C Views	Concurrency structure
Deployment View	Allocation Views	Hardware mapping
Size and Performance	Quality Scenarios; Element Properties	Quantified requirements
Quality	Quality Scenarios; Rationale	Quality decisions

10.3 Phase-Specific Documentation

Documentation by RUP Phase

Inception Phase:
- V&B: System overview; Initial context diagram; Key quality scenarios
- Focus: Stakeholder identification; Architectural vision

Elaboration Phase:
- V&B: Complete view packets for primary views; Element catalogs; Rationale
- Focus: Architecture baseline; Risk mitigation; View selection

Construction Phase:
- V&B: Detailed element catalogs; Interface documentation; Behavior traces
- Focus: Implementation guidance; Integration support

Transition Phase:
- V&B: Deployment views; Install views; Operational documentation
- Focus: Deployment support; Operations handoff

10.4 View Packet Completeness

V&B View Packet Checklist (from RUP artifacts)

Primary Presentation:

- ☐ UML diagram(s) from RUP view
- ☐ Notation key (if non-standard UML)
- ☐ Diagram purpose statement

Element Catalog:

- ☐ All elements from UML diagrams documented
- ☐ Element responsibilities (from class/component descriptions)
- ☐ Element interfaces
- ☐ Element properties (quality attributes)
- ☐ Relations documented

Context Diagram:

- ☐ System boundary (from Use Case diagram)
- ☐ External actors/systems
- ☐ External interfaces

Behavior Documentation:

- ☐ Use case realizations migrated
- ☐ Sequence diagrams included
- ☐ State machines for stateful elements

Rationale:

- ☐ Design decisions documented
- ☐ Pattern/mechanism justification
- ☐ Alternative analysis

Variability Guide:

- ☐ Variation points identified
- ☐ Configuration options

11 Appendix A: Quick Reference Table

Table 9: Quick Reference: RUP to V&B

RUP View	V&B Approach
Use Case View	Behavior documentation associated with views; Quality attribute scenarios in rationale; Context diagram for system boundary
Logical View	Module views (Decomposition, Generalization, Uses) for structural aspects; C&C views for runtime aspects

Implementation View	Module views (Decomposition, Layered) for component structure; Implementation allocation view for source mapping
Process View	C&C Communicating-Processes style for concurrency; Behavior documentation for state and interaction
Deployment View	Deployment allocation view; Install view for installation structure

12 Appendix B: RUP Artifact Cross-Reference

Table 10: RUP Artifacts to V&B Documentation

RUP Artifact	**V&B Documentation**	**Notes**
Software Architecture Document	Architecture Documentation Package	Complete view set
Use Case Model	Context + Behavior	System boundary; scenarios
Analysis Model	System Overview	High-level concepts
Design Model	Module + C&C Views	Detailed structure
Data Model	Data Model View	Persistent data
Implementation Model	Decomposition + Implementation	Code structure
Deployment Model	Deployment View	Hardware mapping
Architectural Prototype	Rationale	Validation evidence
Reference Architecture	Documentation Beyond Views	Reusable patterns
Design Guidelines	Rationale; Beyond Views	Conventions

13 Appendix C: Glossary

4+1 View Model
: Kruchten's architectural view organization with four views plus scenarios

Artifact
: RUP term for a work product

C&C View
: Component-and-Connector view showing runtime structure

Component
: Runtime unit with defined interfaces

Element Catalog
: V&B detailed element specifications

Module	Code-time unit (class, package, library)
Module View	V&B view showing code-time structure
Process	Concurrent execution unit
RUP	Rational Unified Process
SAD	Software Architecture Document
Scenario	Use case or quality attribute scenario
Subsystem	RUP grouping of related components
View Packet	V&B unit of view documentation

14 Appendix D: References

1. Kruchten, P. (1995). "The 4+1 View Model of Architecture." *IEEE Software*, 12(6), 42-50.
2. Kruchten, P. (2003). *The Rational Unified Process: An Introduction* (3rd ed.). Addison-Wesley.
3. Clements, P., et al. (2010). *Documenting Software Architectures: Views and Beyond* (2nd ed.). Addison-Wesley.
4. Bass, L., Clements, P., & Kazman, R. (2021). *Software Architecture in Practice* (4th ed.). Addison-Wesley.
5. Rumbaugh, J., Jacobson, I., & Booch, G. (2004). *The Unified Modeling Language Reference Manual* (2nd ed.). Addison-Wesley.
6. ISO/IEC/IEEE 42010:2011. *Systems and software engineering—Architecture description.*
7. Jacobson, I., Booch, G., & Rumbaugh, J. (1999). *The Unified Software Development Process.* Addison-Wesley.
8. IBM Rational. (2006). *Rational Unified Process: Best Practices for Software Development Teams.* IBM White Paper.