SOMA, RUP and RMC: the right combination for Service Oriented Architecture

WebSphere User Group, Bedfont, 4th March, 2008

Keith Mantell
Senior Solution Architect
IBM Rational
keith_mantell@uk.ibm.com

March 2008

Agenda

- What is SOA?
- Rational Tool Support for SOA
- Development Processes for SOA
  - Rational Unified Process
  - Rational Method Composer
  - RUP SOMA: variations
- Examples

What is Service-Oriented Architecture (SOA) ?

SOA is different things to different people:

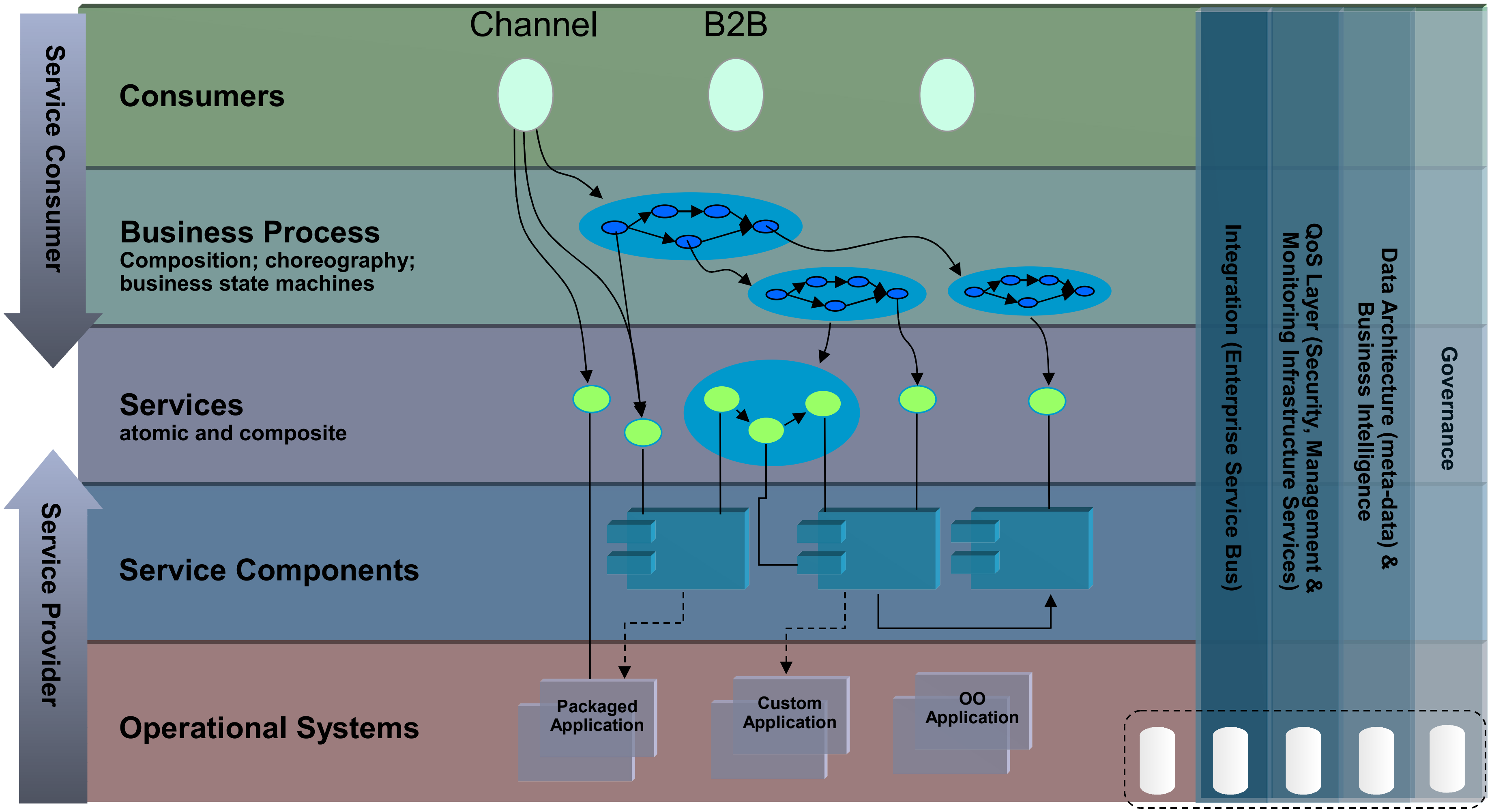
- a **set of services** that a business wants to expose to their customers and partners, or other portions of the organization
- an **architectural style** which requires a service provider, requestor and a service description
- a **set of architectural principles, patterns and criteria** which address characteristics such as *modularity, encapsulation, loose coupling, separation of concerns, reuse, composability*
- a **programming model** complete with standards, tools and technologies such as Web Services
- a **middleware solution** optimized for service assembly, orchestration, monitoring, an management

Business Executive, Analyst

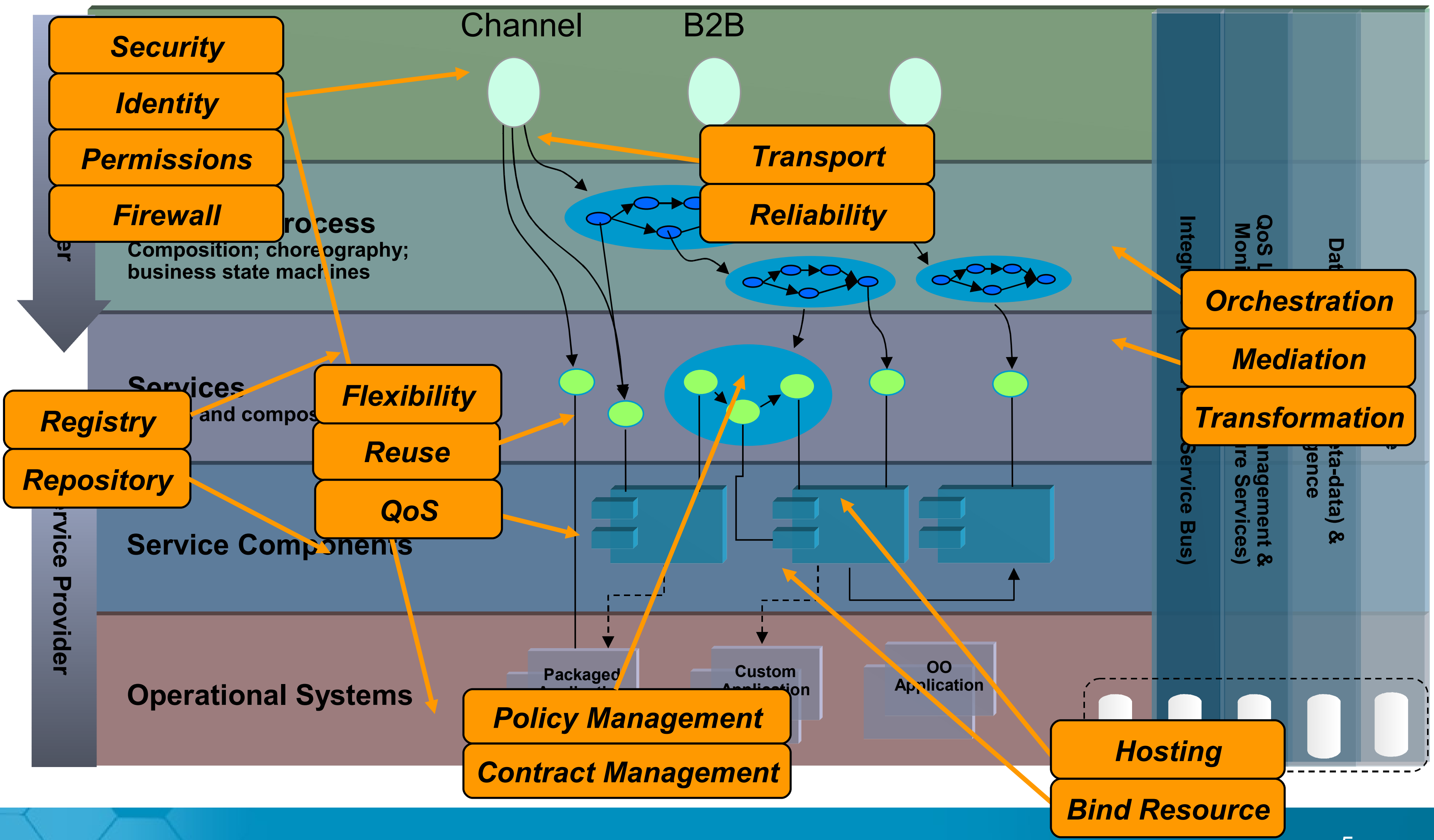
IT Architect

Software and System Developer

Moving to Services-Oriented Solutions – Vision


Service Consumer
Service Provider
Channel
B2B
Consumers
Business Process
Composition; choreography; business state machines
Services
atomic and composite
Service Components
Operational Systems
Packaged Application
Custom Application
OO Application
Integration (Enterprise Service Bus)
QoS Layer (Security, Management & Monitoring Infrastructure Services)
Data Architecture (meta-data) & Business Intelligence
Governance

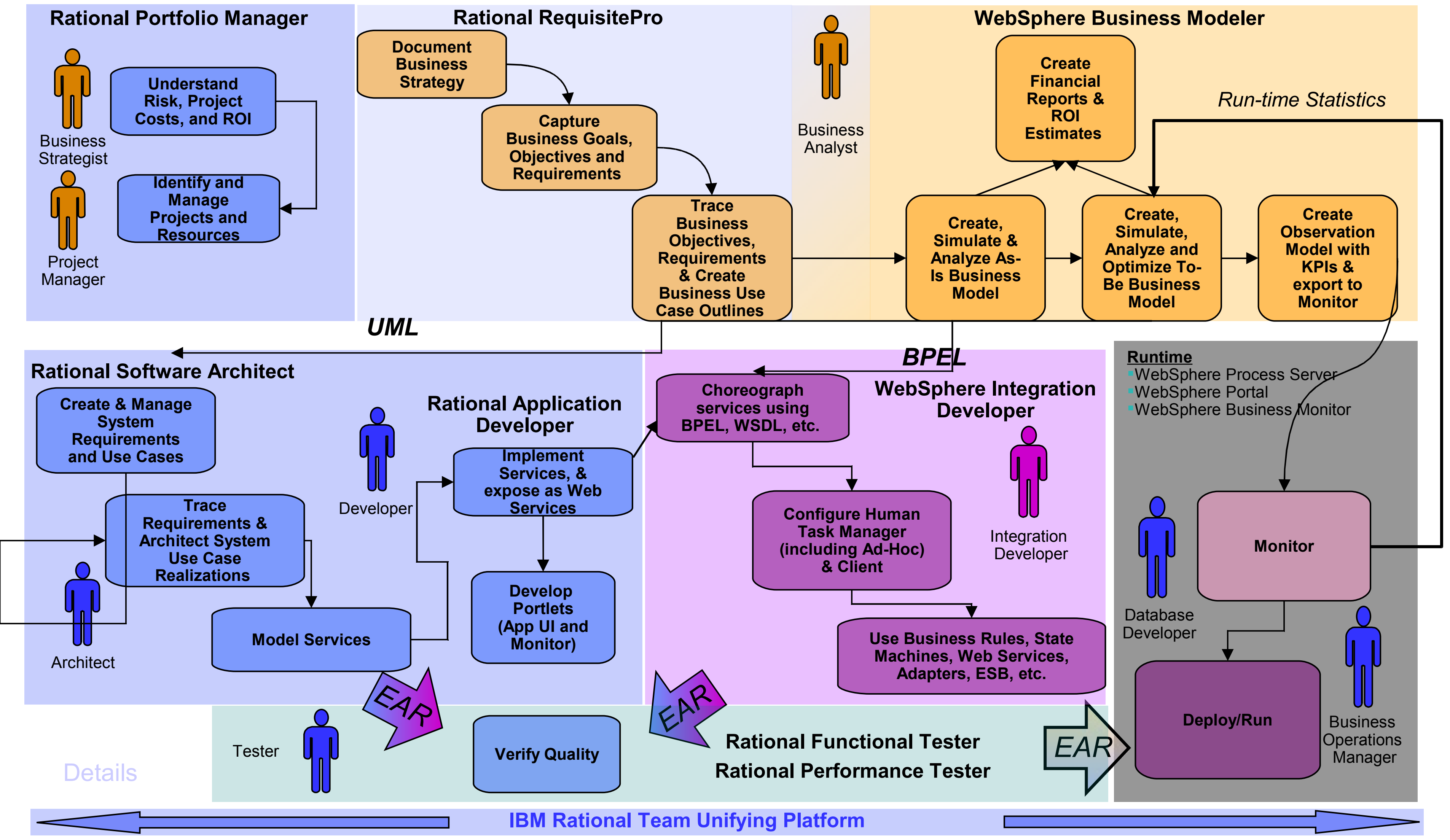
Moving to Services-Oriented Solutions – Challenges



Agenda

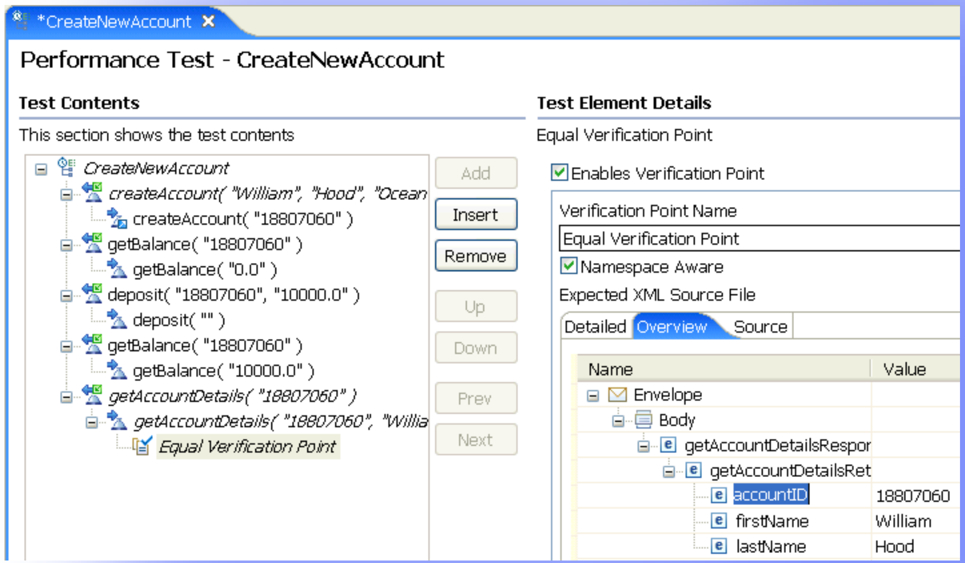
- What is SOA?
- **Rational Tool Support for SOA**
- Development Processes for SOA
  - Rational Unified Process
  - Rational Method Composer
  - RUP SOMA: variations
- Examples

SOA: the Larger Context


Rational Portfolio Manager
Business Strategist
Understand Risk, Project Costs, and ROI
Identify and Manage Projects and Resources
Project Manager
Rational RequisitePro
Document Business Strategy
Capture Business Goals, Objectives and Requirements
Trace Business Objectives, Requirements & Create Business Use Case Outlines
Business Analyst
WebSphere Business Modeler
Create Financial Reports & ROI Estimates
Run-time Statistics
Create, Simulate & Analyze As-Is Business Model
Create, Simulate, Analyze and Optimize To-Be Business Model
Create Observation Model with KPIs & export to Monitor
UML
BPEL
Rational Software Architect
Create & Manage System Requirements and Use Cases
Trace Requirements & Architect System Use Case Realizations
Architect
Model Services
Rational Application Developer
Developer
Implement Services, & expose as Web Services
Develop Portlets (App UI and Monitor)
Choreograph services using BPEL, WSDL, etc.
WebSphere Integration Developer
Integration Developer
Configure Human Task Manager (including Ad-Hoc) & Client
Use Business Rules, State Machines, Web Services, Adapters, ESB, etc.
Runtime
•WebSphere Process Server
•WebSphere Portal
•WebSphere Business Monitor
Database Developer
Monitor
Deploy/Run
Business Operations Manager
EAR
EAR
EAR
Tester
Verify Quality
Rational Functional Tester
Rational Performance Tester
Details
IBM Rational Team Unifying Platform

Service Quality Management

Functional and Performance Testing of Web Services from a common interface



*CreateNewAccount

Performance Test - CreateNewAccount

Test Contents

This section shows the test contents

```
CreateNewAccount
  createAccount( "William", "Hood", "Ocean
    createAccount( "18807060" )
  getBalance( "18807060" )
    getBalance( "0.0" )
  deposit( "18807060", "10000.0" )
    deposit( "" )
  getBalance( "18807060" )
    getBalance( "10000.0" )
  getAccountDetails( "18807060" )
    getAccountDetails( "18807060", "Willia
      Equal Verification Point
```

Test Element Details

Equal Verification Point

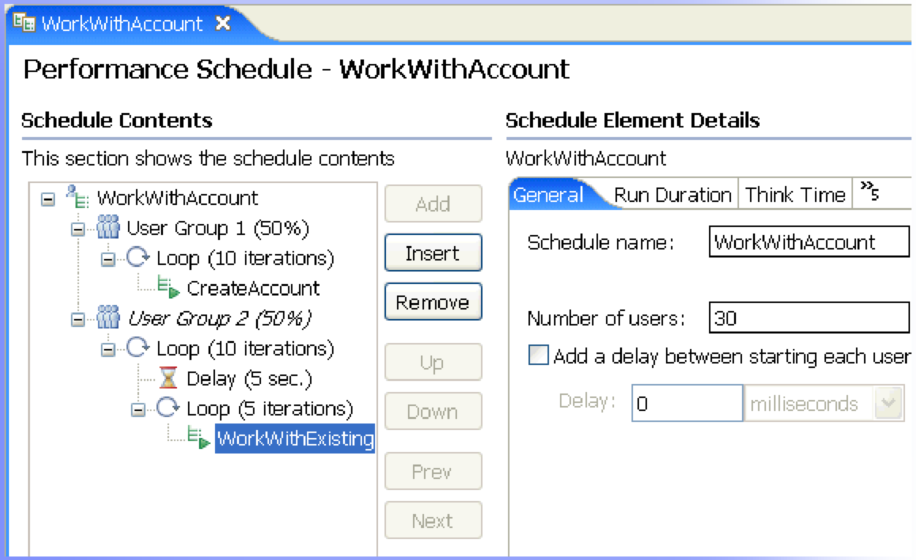
☑ Enables Verification Point

Verification Point Name: Equal Verification Point

☑ Namespace Aware

Expected XML Source File

Name	Value
Envelope	
Body	
getAccountDetailsRespor	
getAccountDetailsRet	
accountID	18807060
firstName	William
lastName	Hood



WorkWithAccount

Performance Schedule - WorkWithAccount

Schedule Contents

This section shows the schedule contents

```
WorkWithAccount
  User Group 1 (50%)
    Loop (10 iterations)
      CreateAccount
  User Group 2 (50%)
    Loop (10 iterations)
      Delay (5 sec.)
      Loop (5 iterations)
        WorkWithExisting
```

Schedule Element Details

WorkWithAccount

Schedule name: WorkWithAccount

Number of users: 30

☐ Add a delay between starting each user

Delay: 0 milliseconds

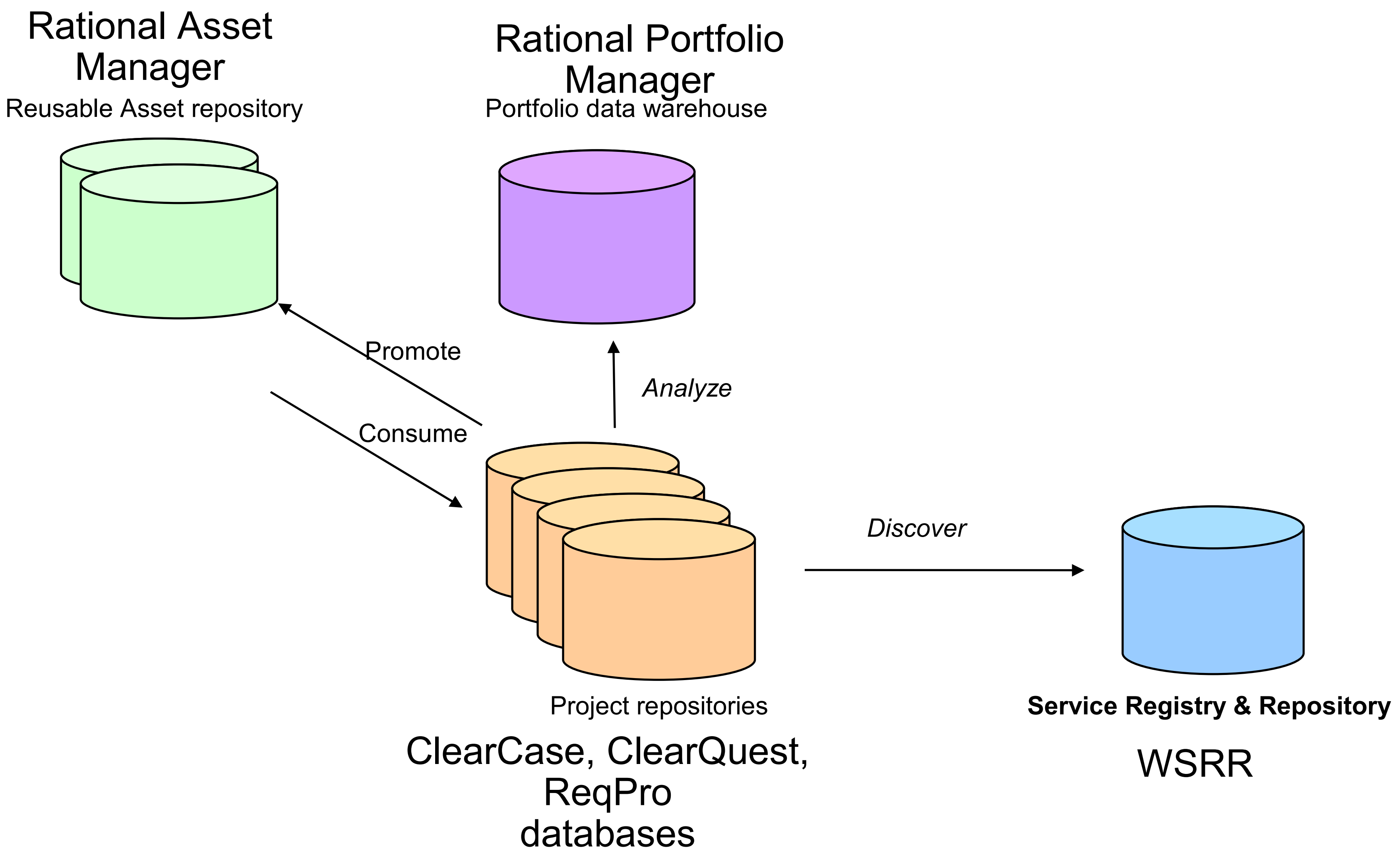
Rational Tester for SOA Quality

Automated regression and functional testing for GUI-less Web services

Rational Performance Tester Extension for SOA Quality

Performance Testing for Web Service based applications

Asset Management – Solutions


Rational Asset Manager
Reusable Asset repository
Rational Portfolio Manager
Portfolio data warehouse
Promote
Consume
Analyze
Discover
Project repositories
ClearCase, ClearQuest, ReqPro databases
Service Registry & Repository
WSRR

Agenda

- What is SOA?
- Rational Tool Support for SOA
- Development Processes for SOA
  - **Rational Unified Process**
  - Rational Method Composer
  - RUP SOMA: variations
- Examples

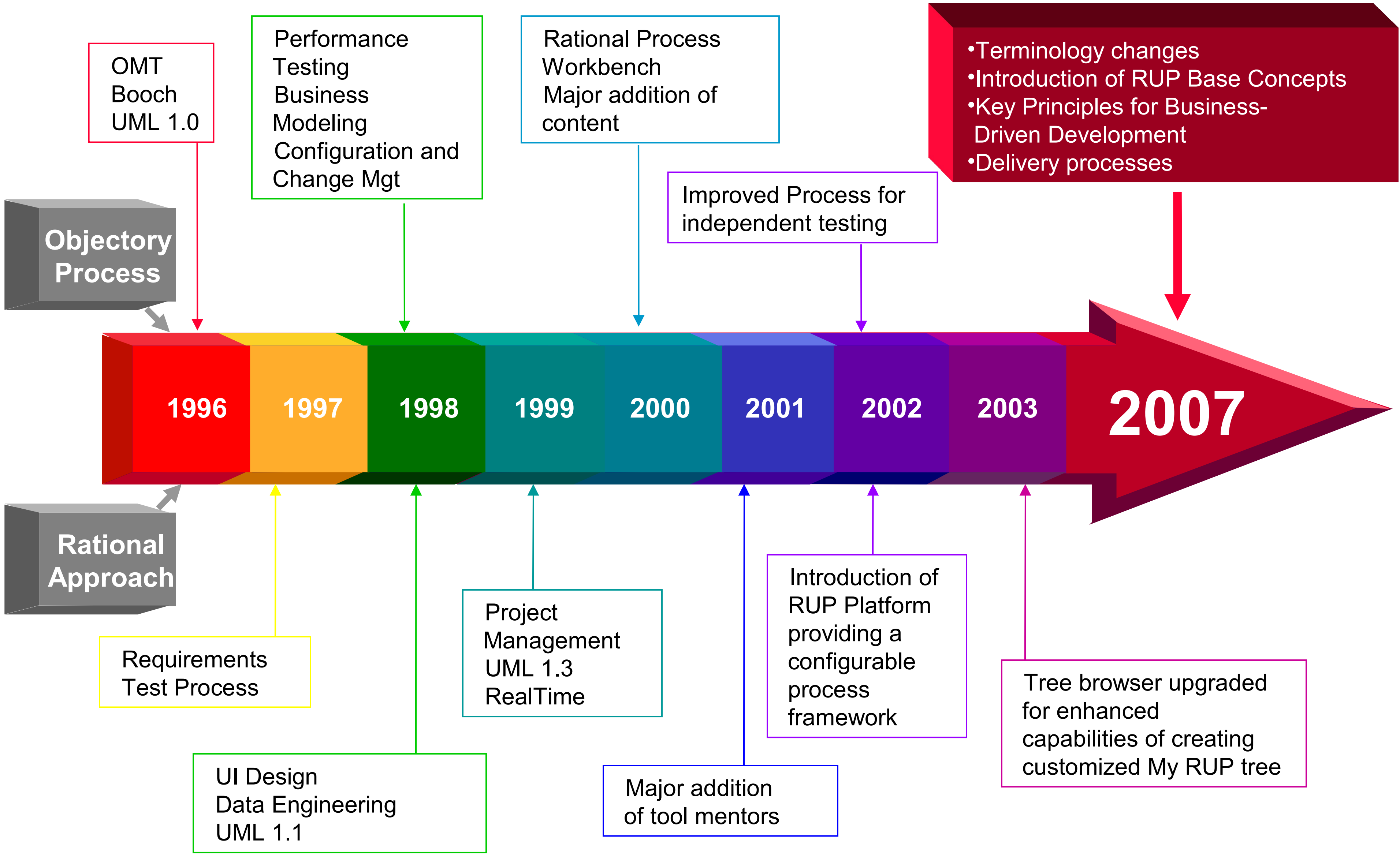
Why Use the Rational Unified Process(RUP)?

- RUP provides a software development practitioner with a standards-based yet configurable process environment. That process environment:
  - Allows a tailored method to be published and made accessible to the entire project team
  - Allows that method to be configured to suit the unique needs of each project
  - Provides each user with customized filtering
- RUP is a body of software engineering practices that are continually improved to reflect changes in industry practices.

Why Should I Use RUP? (cont.)

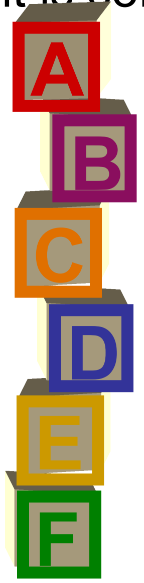
- For stakeholders
  - RUP provides a glossary of terminology and an encyclopedia of knowledge to help you communicate your needs effectively with the software development team.
- For software development practitioners
  - RUP provides a central, common process definition that team members can share, helping to improve communication.
  - RUP provides a wealth of guidance on software development practices
- For managers or team leaders
  - RUP provides you with a process by which you can communicate effectively with your staff, and manage the planning and control of their work accordingly.
- For process engineers
  - RUP provides you with an architectural foundation and wealth of material from which you can construct your process definition.

History of the Rational Unified Process


OMT
Booch
UML 1.0
Performance Testing
Business Modeling
Configuration and Change Mgt
Rational Process Workbench
Major addition of content
Improved Process for independent testing
•Terminology changes
•Introduction of RUP Base Concepts
•Key Principles for Business-Driven Development
•Delivery processes
Objectory Process
Rational Approach
1996
1997
1998
1999
2000
2001
2002
2003
2007
Requirements Test Process
UI Design
Data Engineering
UML 1.1
Project Management
UML 1.3
RealTime
Major addition of tool mentors
Introduction of RUP Platform providing a configurable process framework
Tree browser upgraded for enhanced capabilities of creating customized My RUP tree

Key Principles for Business-Driven Development

- The tried-and-true best practices of the Rational Unified Process have been the basis for the evolution of our tools and processes for more than a decade.
- Today, as software development is becoming a key business capability, our best practices are maturing within the larger context of business-driven development.
- The following six principles re-articulate our best practices for the broader lifecycle of continuously evolving systems, in which the primary evolving element is software:



Adapt The Process

Balance Competing Stakeholder Priorities

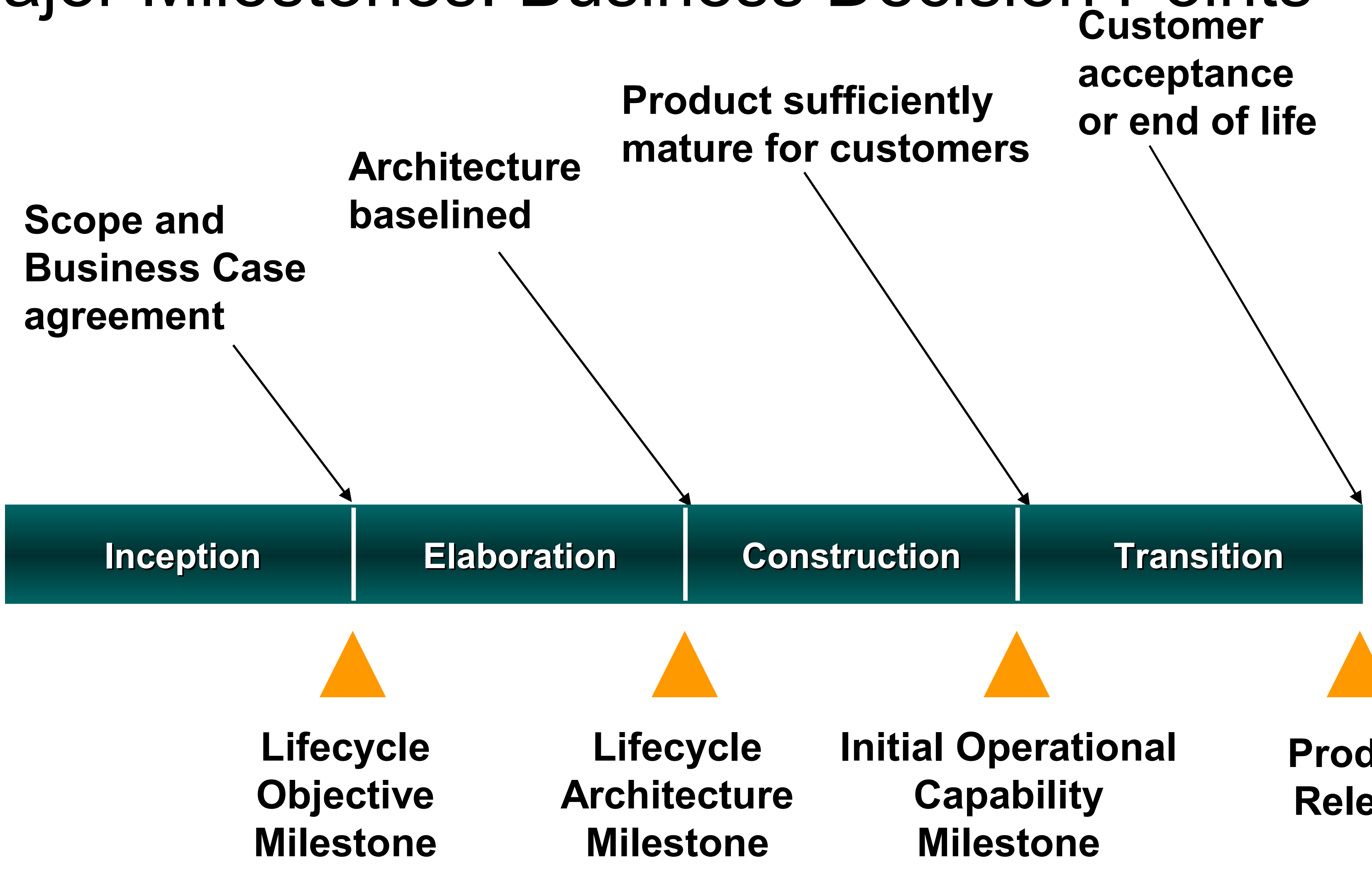
Collaborate Across Teams

Demonstrate Value Iteratively

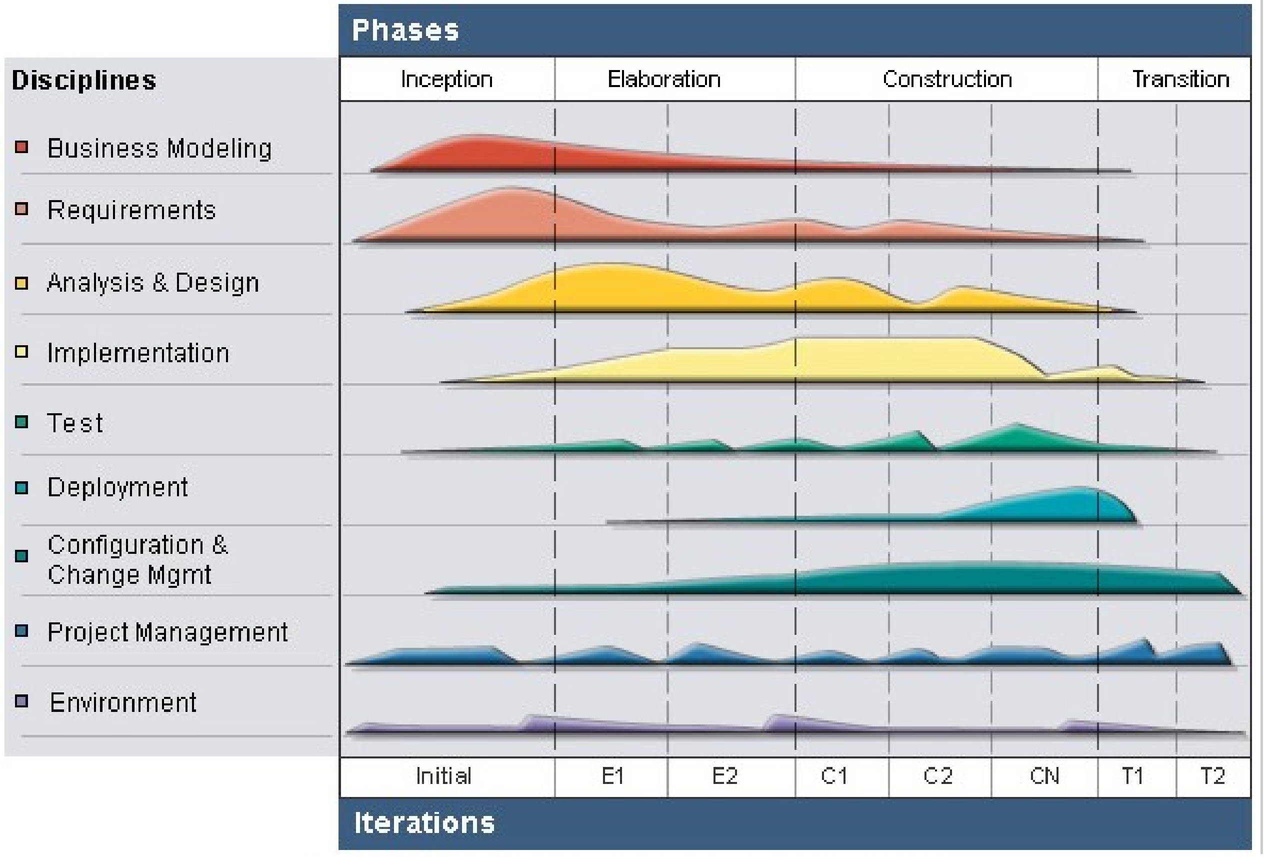
Elevate Level Of Abstraction

Focus Continuously On Quality

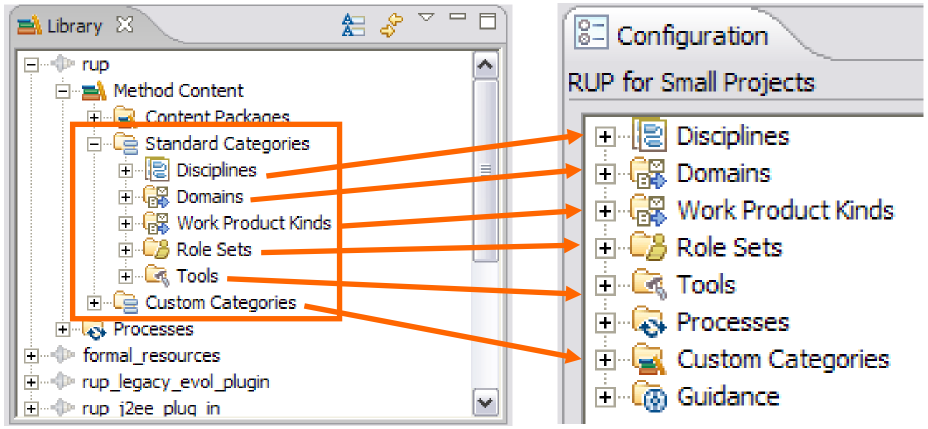
Major Milestones: Business Decision Points


Scope and Business Case agreement
Architecture baselined
Product sufficiently mature for customers
Customer acceptance or end of life
Inception
Elaboration
Construction
Transition
Lifecycle Objective Milestone
Lifecycle Architecture Milestone
Initial Operational Capability Milestone
Product Release

What is Rational Unified Process(RUP)?



More detail!

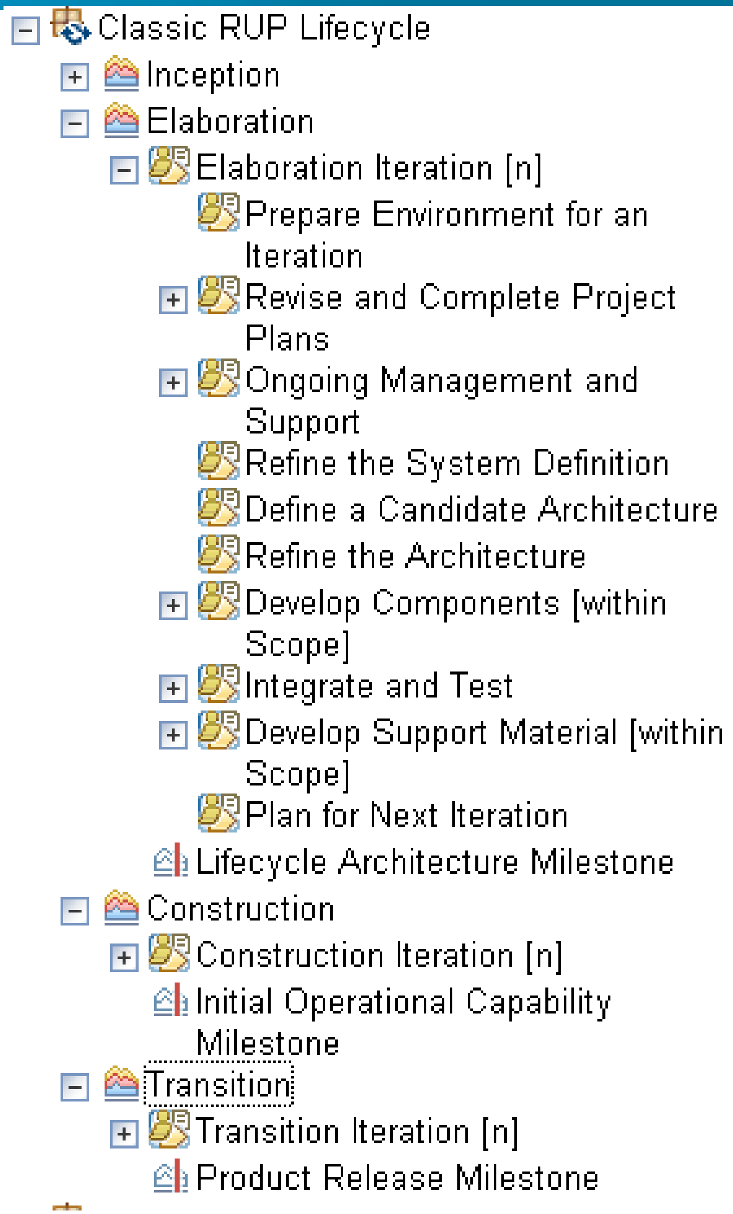

Library

- rup
  - Method Content
    - Content Packages
    - Standard Categories
      - Disciplines
      - Domains
      - Work Product Kinds
      - Role Sets
      - Tools
    - Custom Categories
  - Processes
- formal_resources
- rup_legacy_evol_plugin
- rup_j2ee_plug_in

Configuration

RUP for Small Projects

- Disciplines
- Domains
- Work Product Kinds
- Role Sets
- Tools
- Processes
- Custom Categories
- Guidance


- Classic RUP Lifecycle
  - Inception
  - Elaboration
    - Elaboration Iteration [n]
      - Prepare Environment for an Iteration
      - Revise and Complete Project Plans
      - Ongoing Management and Support
      - Refine the System Definition
      - Define a Candidate Architecture
      - Refine the Architecture
      - Develop Components [within Scope]
      - Integrate and Test
      - Develop Support Material [within Scope]
      - Plan for Next Iteration
    - Lifecycle Architecture Milestone
  - Construction
    - Construction Iteration [n]
    - Initial Operational Capability Milestone
  - Transition
    - Transition Iteration [n]
    - Product Release Milestone

Agenda

- What is SOA?
- Rational Tool Support for SOA
- Development Processes for SOA
  - **Rational Unified Process**
  - Rational Method Composer
  - RUP SOMA: variations
- Examples

Structuring Process Content

Standardize representation and manage libraries of reusable **Method Content**

Content on agile development

Content on managing iterative development

Guidance on serialized java beans

JUnit user guidance

Content on J2EE

Configuration mgmt guidelines

Develop and manage **Processes** for performing projects

Lessons learnt from previous project and iteration

Corporate guidelines on compliance

Process assets patterns

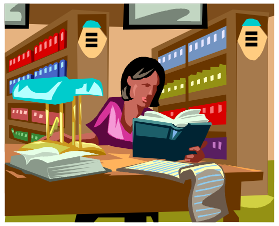
Standard or reference processes

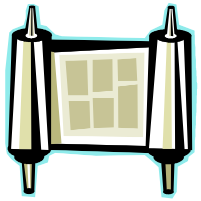
Project plan templates

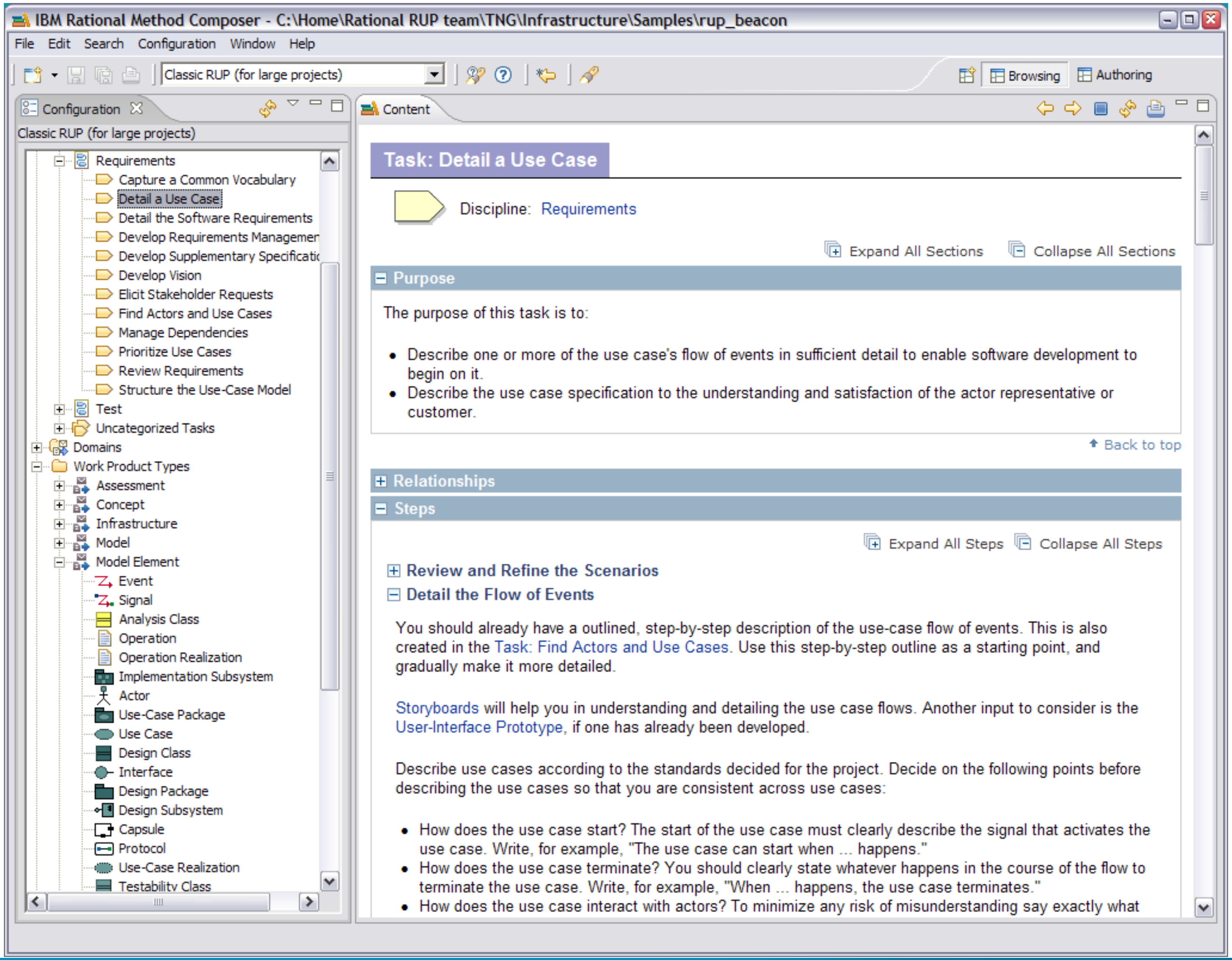
Configure a cohesive process framework customized for my project needs

Create project plan templates for **Enactment** of process in the context of my project

Method Content Example







IBM Rational Method Composer - C:\Home\Rational RUP team\TNG\Infrastructure\Samples\rup_beacon

File Edit Search Configuration Window Help
Classic RUP (for large projects)
Browsing
Authoring
Configuration
Content

Classic RUP (for large projects)

- Requirements
  - Capture a Common Vocabulary
  - Detail a Use Case
  - Detail the Software Requirements
  - Develop Requirements Managemer
  - Develop Supplementary Specificati
  - Develop Vision
  - Elicit Stakeholder Requests
  - Find Actors and Use Cases
  - Manage Dependencies
  - Prioritize Use Cases
  - Review Requirements
  - Structure the Use-Case Model
- Test
- Uncategorized Tasks
- Domains
- Work Product Types
  - Assessment
  - Concept
  - Infrastructure
  - Model
  - Model Element
    - Event
    - Signal
    - Analysis Class
    - Operation
    - Operation Realization
    - Implementation Subsystem
    - Actor
    - Use-Case Package
    - Use Case
    - Design Class
    - Interface
    - Design Package
    - Design Subsystem
    - Capsule
    - Protocol
    - Use-Case Realization
    - Testability Class

Task: Detail a Use Case

Discipline: Requirements

Expand All Sections   Collapse All Sections

Purpose

The purpose of this task is to:

- Describe one or more of the use case's flow of events in sufficient detail to enable software development to begin on it.
- Describe the use case specification to the understanding and satisfaction of the actor representative or customer.

Back to top

Relationships

Steps

Expand All Steps   Collapse All Steps

Review and Refine the Scenarios

Detail the Flow of Events

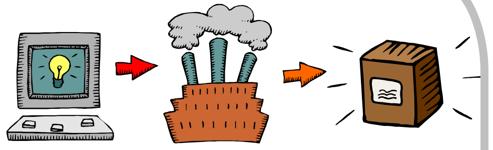
You should already have a outlined, step-by-step description of the use-case flow of events. This is also created in the Task: Find Actors and Use Cases. Use this step-by-step outline as a starting point, and gradually make it more detailed.

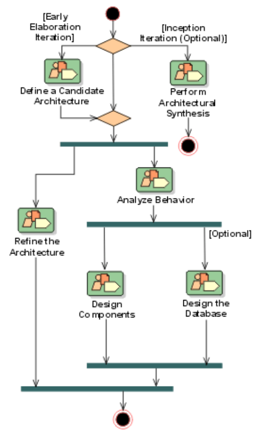
Storyboards will help you in understanding and detailing the use case flows. Another input to consider is the User-Interface Prototype, if one has already been developed.

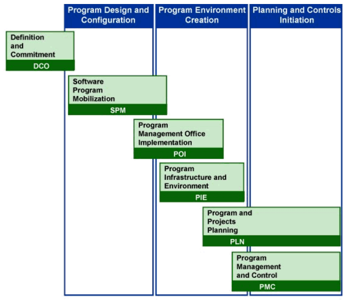
Describe use cases according to the standards decided for the project. Decide on the following points before describing the use cases so that you are consistent across use cases:

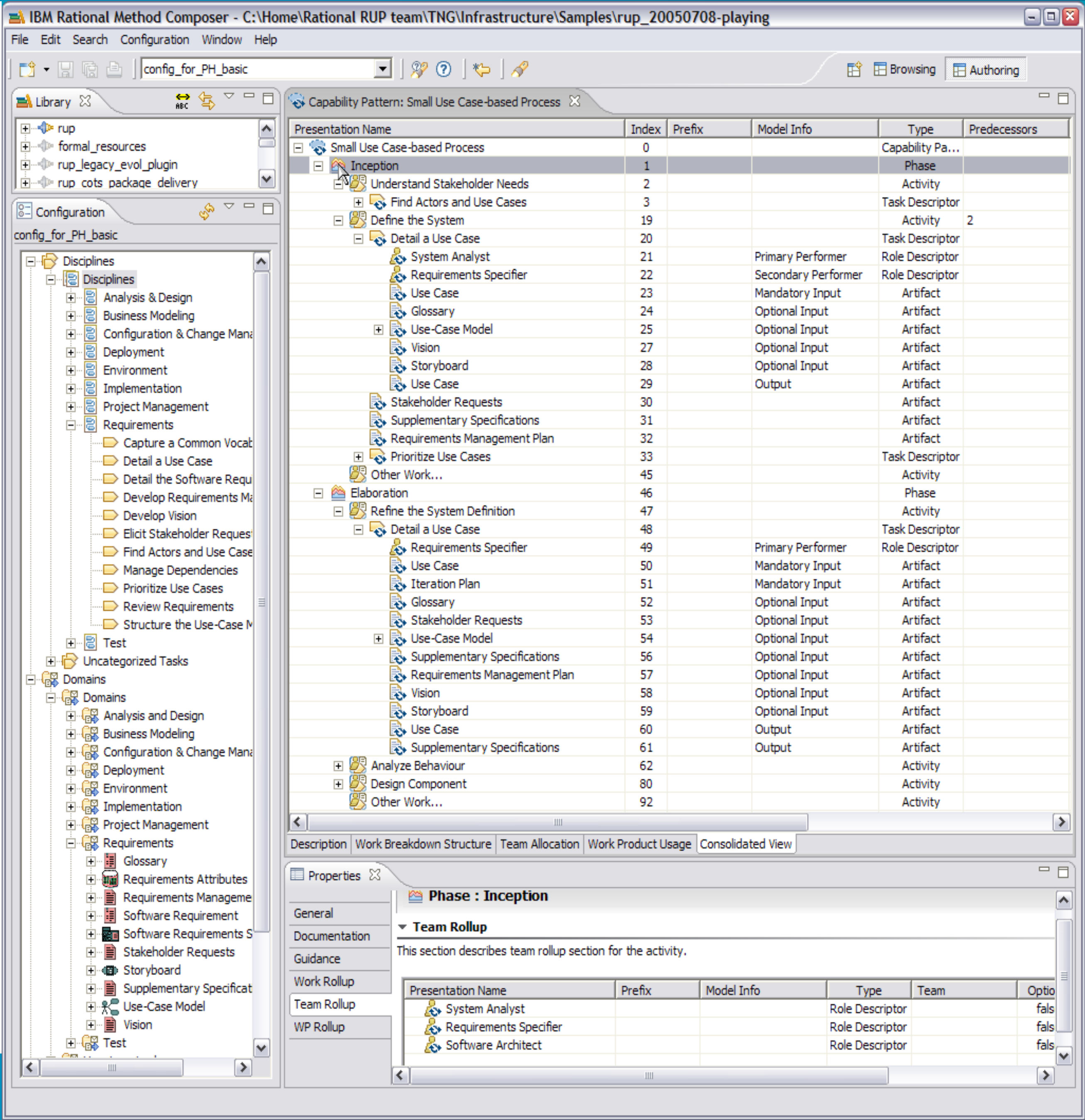
- How does the use case start? The start of the use case must clearly describe the signal that activates the use case. Write, for example, "The use case can start when ... happens."
- How does the use case terminate? You should clearly state whatever happens in the course of the flow to terminate the use case. Write, for example, "When ... happens, the use case terminates."
- How does the use case interact with actors? To minimize any risk of misunderstanding say exactly what

Process Example




[Early Elaboration Iteration]
[Inception Iteration (Optional)]
Define a Candidate Architecture
Perform Architectural Synthesis
Analyze Behavior
Refine the Architecture
[Optional]
Design Components
Design the Database


Program Design and Configuration
Program Environment Creation
Planning and Controls Initiation
Definition and Commitment
DCO
Software Program Mobilization
SPM
Program Management Office Implementation
POI
Program Infrastructure and Environment
PIE
Program and Projects Planning
PLN
Program Management and Control
PMC



Library:
- rup
- formal_resources
- rup_legacy_evol_plugin
- rup_cots_package_delivery

Configuration: config_for_PH_basic

- Disciplines
  - Disciplines
    - Analysis & Design
    - Business Modeling
    - Configuration & Change Mana
    - Deployment
    - Environment
    - Implementation
    - Project Management
    - Requirements
      - Capture a Common Vocab
      - Detail a Use Case
      - Detail the Software Requ
      - Develop Requirements Ma
      - Develop Vision
      - Elicit Stakeholder Reques
      - Find Actors and Use Case
      - Manage Dependencies
      - Prioritize Use Cases
      - Review Requirements
      - Structure the Use-Case M
    - Test
  - Uncategorized Tasks
- Domains
  - Domains
    - Analysis and Design
    - Business Modeling
    - Configuration & Change Mana
    - Deployment
    - Environment
    - Implementation
    - Project Management
    - Requirements
      - Glossary
      - Requirements Attributes
      - Requirements Manageme
      - Software Requirement
      - Software Requirements S
      - Stakeholder Requests
      - Storyboard
      - Supplementary Specificat
      - Use-Case Model
      - Vision
    - Test

Presentation Name	Index	Prefix	Model Info	Type	Predecessors
Small Use Case-based Process	0			Capability Pa...	
Inception	1			Phase	
Understand Stakeholder Needs	2			Activity	
Find Actors and Use Cases	3			Task Descriptor	
Define the System	19			Activity	2
Detail a Use Case	20			Task Descriptor	
System Analyst	21		Primary Performer	Role Descriptor	
Requirements Specifier	22		Secondary Performer	Role Descriptor	
Use Case	23		Mandatory Input	Artifact	
Glossary	24		Optional Input	Artifact	
Use-Case Model	25		Optional Input	Artifact	
Vision	27		Optional Input	Artifact	
Storyboard	28		Optional Input	Artifact	
Use Case	29		Output	Artifact	
Stakeholder Requests	30			Artifact	
Supplementary Specifications	31			Artifact	
Requirements Management Plan	32			Artifact	
Prioritize Use Cases	33			Task Descriptor	
Other Work...	45			Activity	
Elaboration	46			Phase	
Refine the System Definition	47			Activity	
Detail a Use Case	48			Task Descriptor	
Requirements Specifier	49		Primary Performer	Role Descriptor	
Use Case	50		Mandatory Input	Artifact	
Iteration Plan	51		Mandatory Input	Artifact	
Glossary	52		Optional Input	Artifact	
Stakeholder Requests	53		Optional Input	Artifact	
Use-Case Model	54		Optional Input	Artifact	
Supplementary Specifications	56		Optional Input	Artifact	
Requirements Management Plan	57		Optional Input	Artifact	
Vision	58		Optional Input	Artifact	
Storyboard	59		Optional Input	Artifact	
Use Case	60		Output	Artifact	
Supplementary Specifications	61		Output	Artifact	
Analyze Behaviour	62			Activity	
Design Component	80			Activity	
Other Work...	92			Activity	

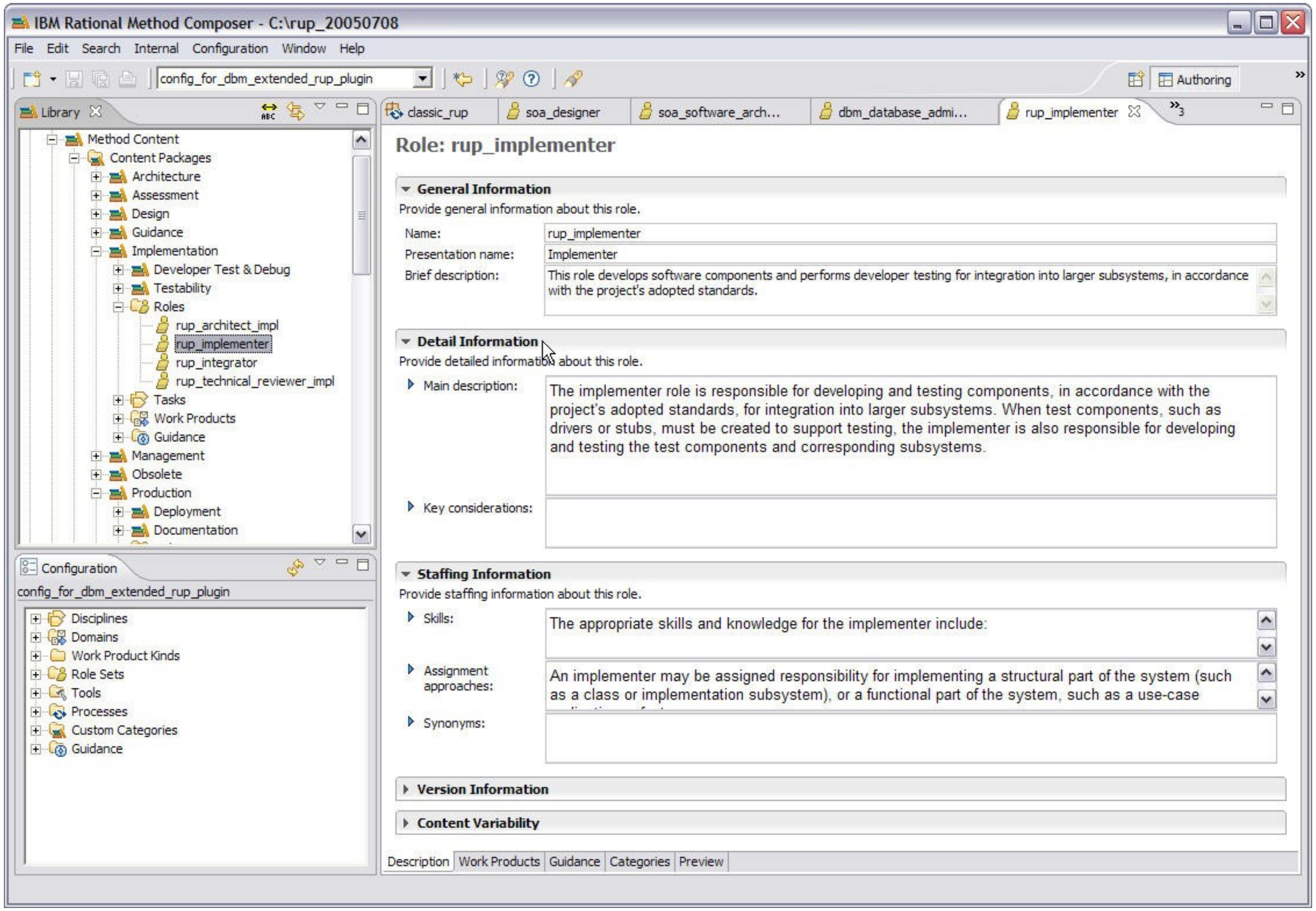
Phase : Inception

Team Rollup

This section describes team rollup section for the activity.

Presentation Name	Prefix	Model Info	Type	Team	Optio
System Analyst			Role Descriptor		fals
Requirements Specifier			Role Descriptor		fals
Software Architect			Role Descriptor		fals

Tools - Authoring, configuring and viewing capabilities



Role: rup_implementer

General Information

Provide general information about this role.

Name:	rup_implementer
Presentation name:	Implementer
Brief description:	This role develops software components and performs developer testing for integration into larger subsystems, in accordance with the project's adopted standards.

Detail Information

Provide detailed information about this role.

Main description: The implementer role is responsible for developing and testing components, in accordance with the project's adopted standards, for integration into larger subsystems. When test components, such as drivers or stubs, must be created to support testing, the implementer is also responsible for developing and testing the test components and corresponding subsystems.

Key considerations:

Staffing Information

Provide staffing information about this role.

Skills: The appropriate skills and knowledge for the implementer include:

Assignment approaches: An implementer may be assigned responsibility for implementing a structural part of the system (such as a class or implementation subsystem), or a functional part of the system, such as a use-case

Synonyms:

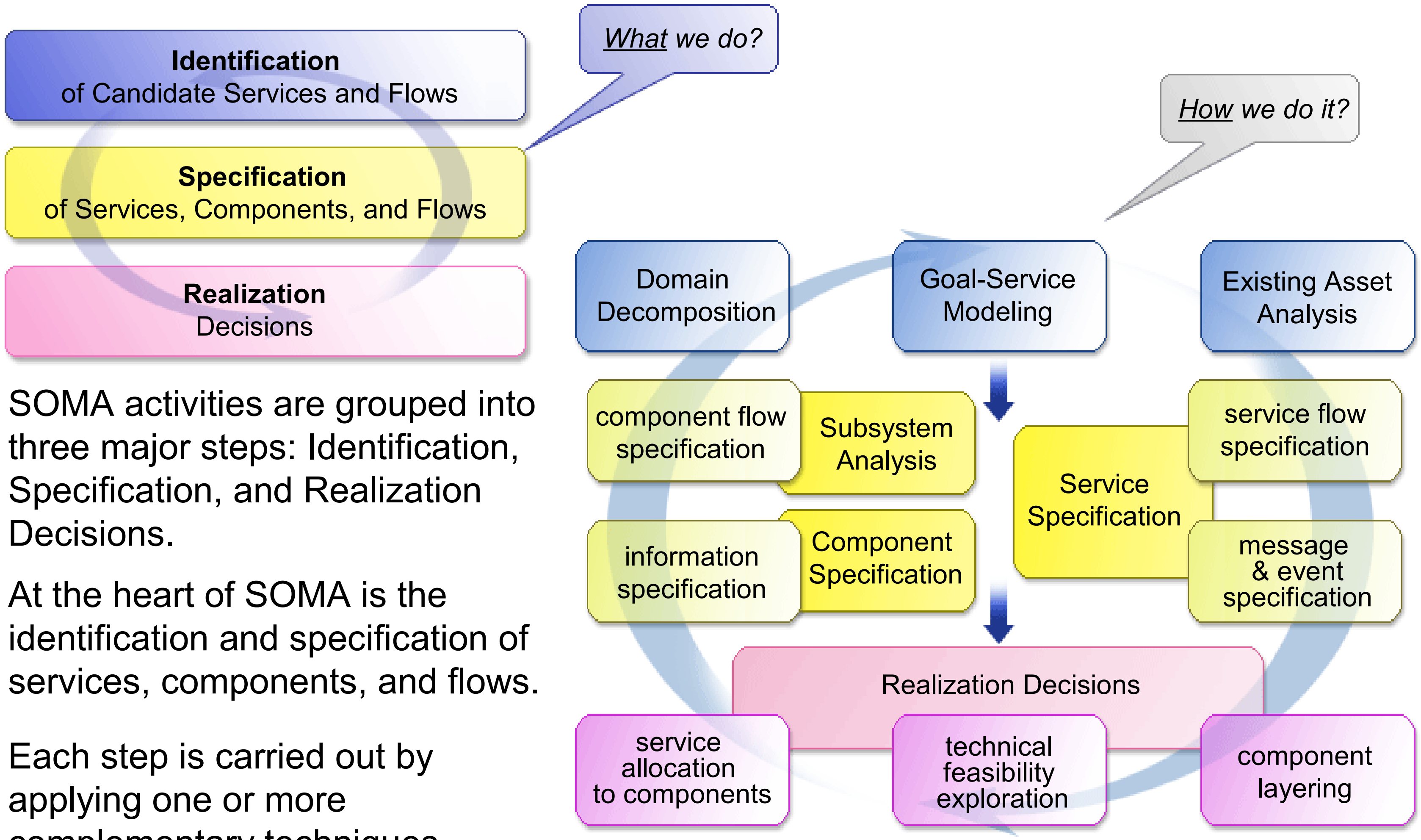
Version Information

Content Variability

Agenda

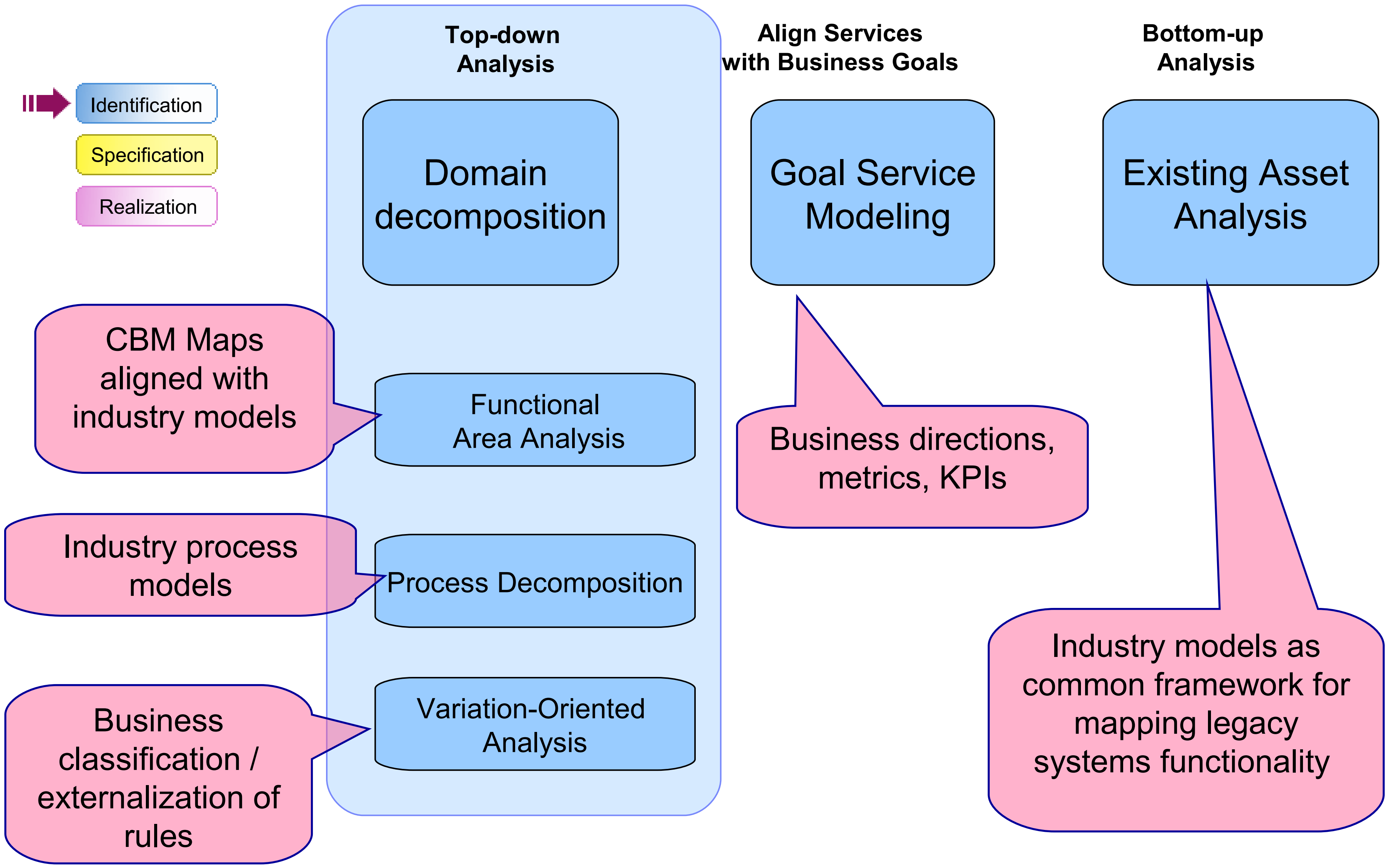
- What is SOA?
- Rational Tool Support for SOA
- Development Processes for SOA
  - **Rational Unified Process**
  - Rational Method Composer
  - RUP SOMA: variations
- Examples

SOMA activities are grouped into three major steps



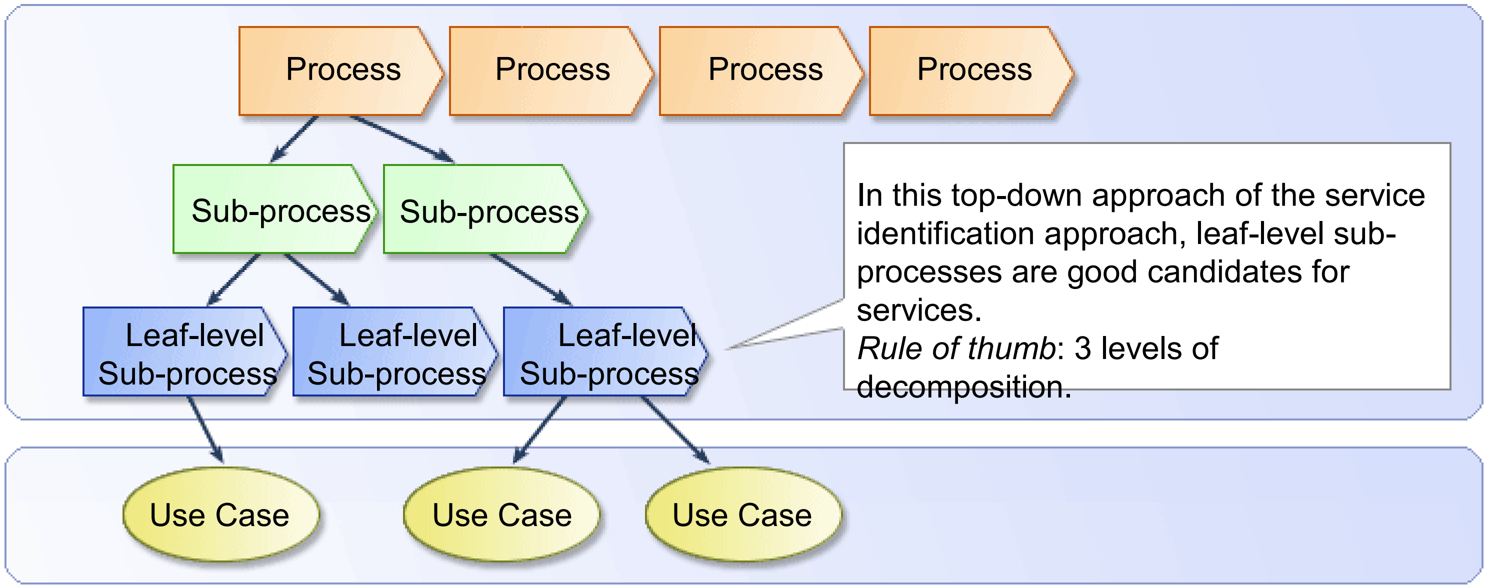
- SOMA activities are grouped into three major steps: Identification, Specification, and Realization Decisions.
- At the heart of SOMA is the identification and specification of services, components, and flows.
- Each step is carried out by applying one or more complementary techniques.

Service Identification


Identification
Specification
Realization
Top-down Analysis
Align Services with Business Goals
Bottom-up Analysis
Domain decomposition
Goal Service Modeling
Existing Asset Analysis
CBM Maps aligned with industry models
Functional Area Analysis
Business directions, metrics, KPIs
Industry process models
Process Decomposition
Industry models as common framework for mapping legacy systems functionality
Business classification / externalization of rules
Variation-Oriented Analysis

Process decomposition helps identify candidate services

Process Decomposition

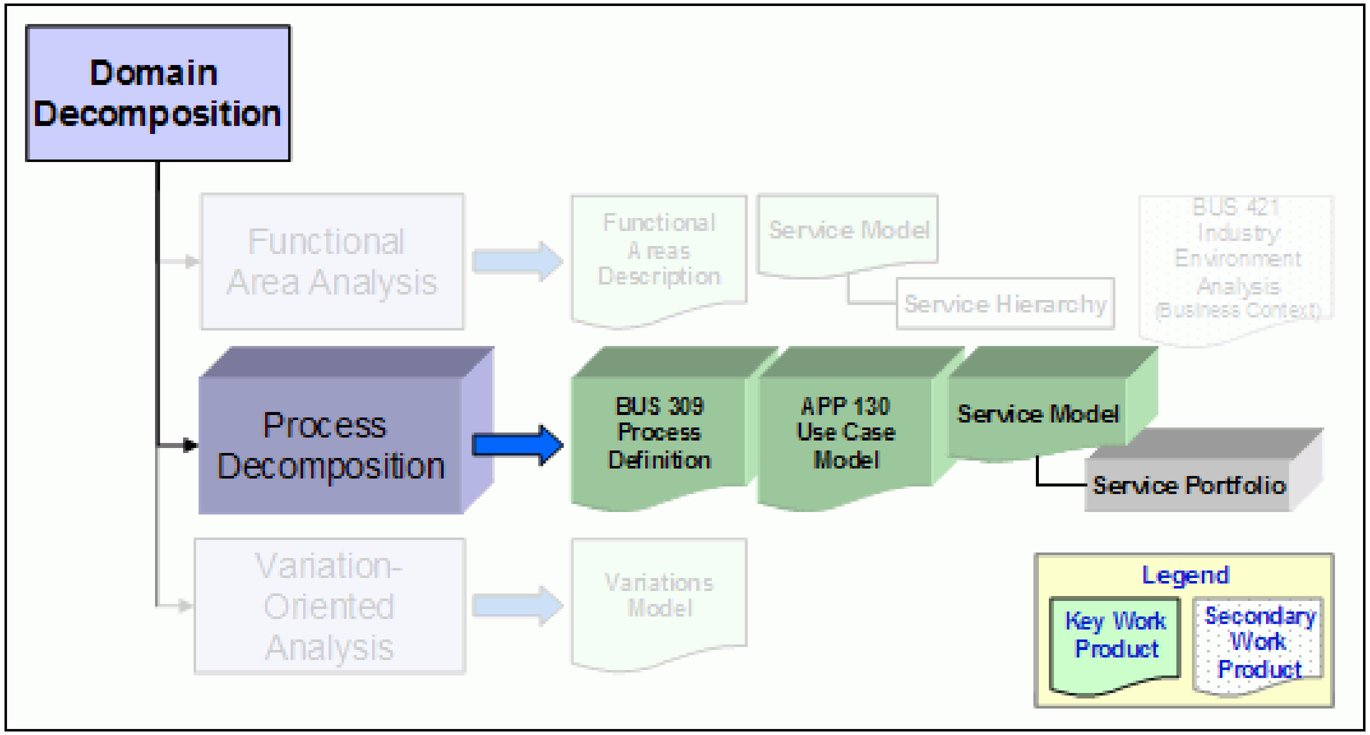


In this top-down approach of the service identification approach, leaf-level sub-processes are good candidates for services.
Rule of thumb: 3 levels of decomposition.

- A sub-process is a convenient construct used to denote further levels of refinement to a process into its constituent parts (sub-processes), recursively.
- Sub-processes are used to identify candidate services.
- The list of use cases provides the initial scope for system design (“business as usual”).

Process Decomposition work products

Process Decomposition



Process Definition

Use Case Model

Service Model

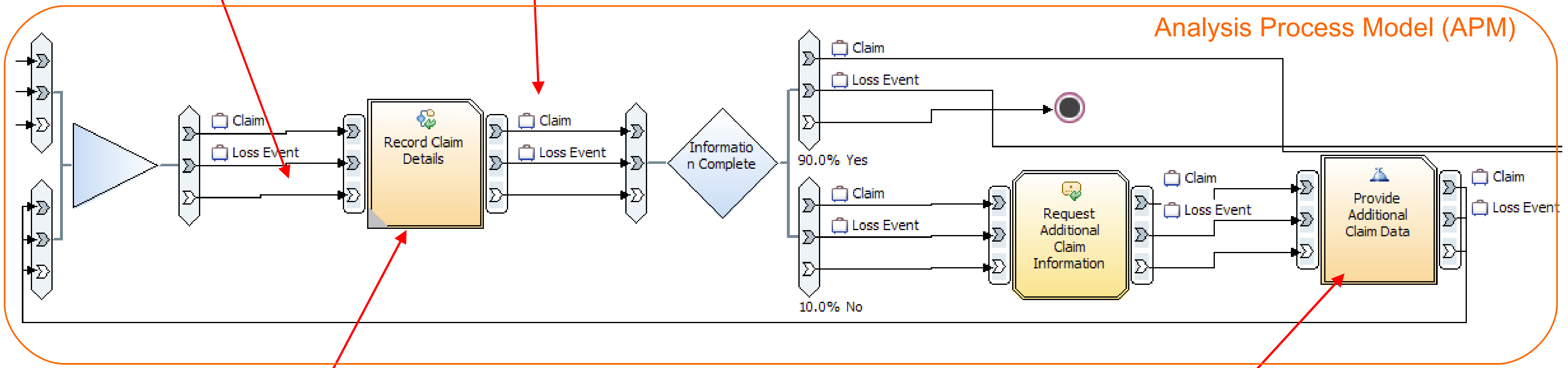
Detailed process analysis

Process models

Process Definition

✓Customize Control Flow

✓Define Business Concepts And Information Flows

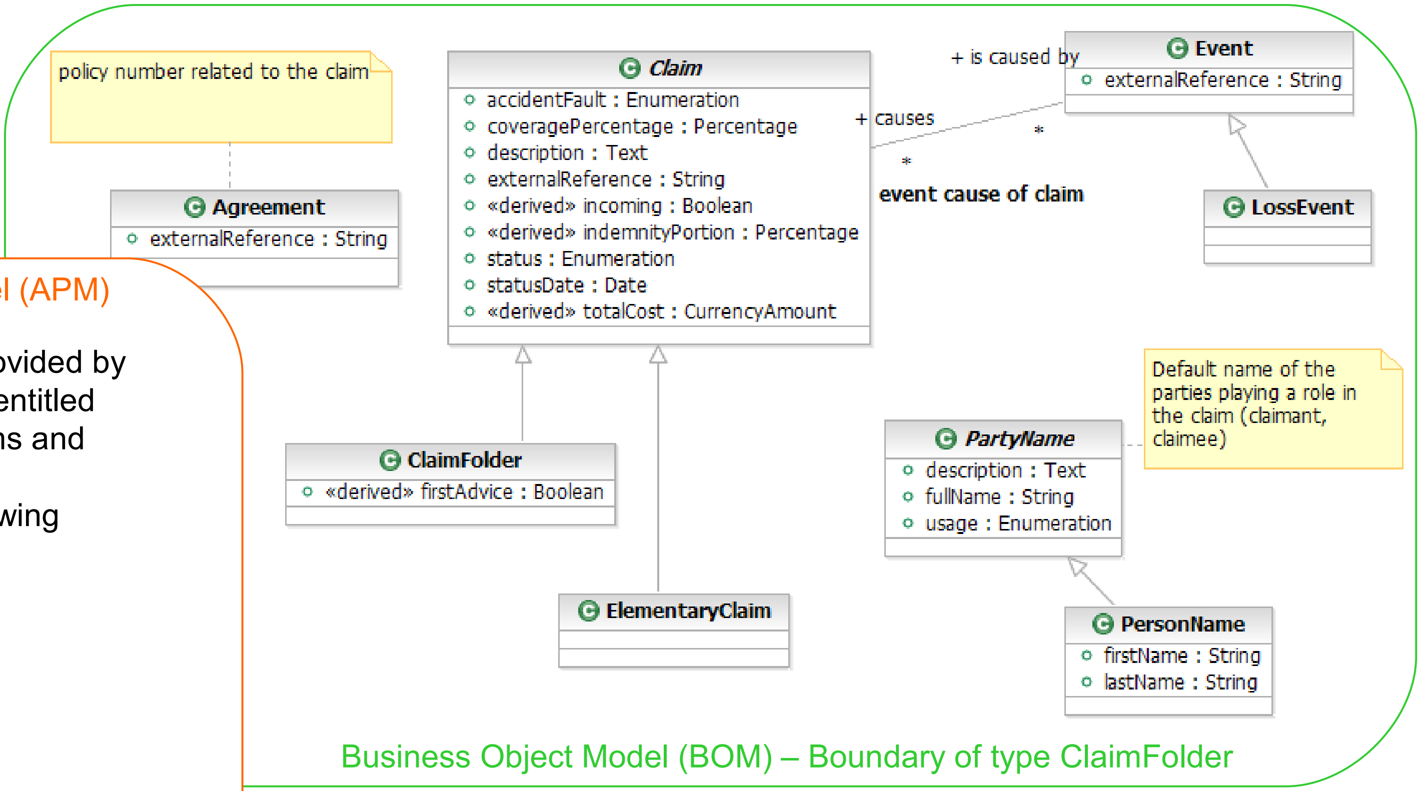


✓Identify Automated Activities

✓Assign Roles

Service Analysis with RSM/RSA

✓Extend the Class model based on the information requirements of the use case
✓Define boundary of type class diagram


policy number related to the claim
Agreement
externalReference : String
Claim
accidentFault : Enumeration
coveragePercentage : Percentage
description : Text
externalReference : String
«derived» incoming : Boolean
«derived» indemnityPortion : Percentage
status : Enumeration
statusDate : Date
«derived» totalCost : CurrencyAmount
+ is caused by
+ causes
*
*
event cause of claim
Event
externalReference : String
LossEvent
ClaimFolder
«derived» firstAdvice : Boolean
ElementaryClaim
PartyName
description : Text
fullName : String
usage : Enumeration
Default name of the parties playing a role in the claim (claimant, claimee)
PersonName
firstName : String
lastName : String
Business Object Model (BOM) – Boundary of type ClaimFolder

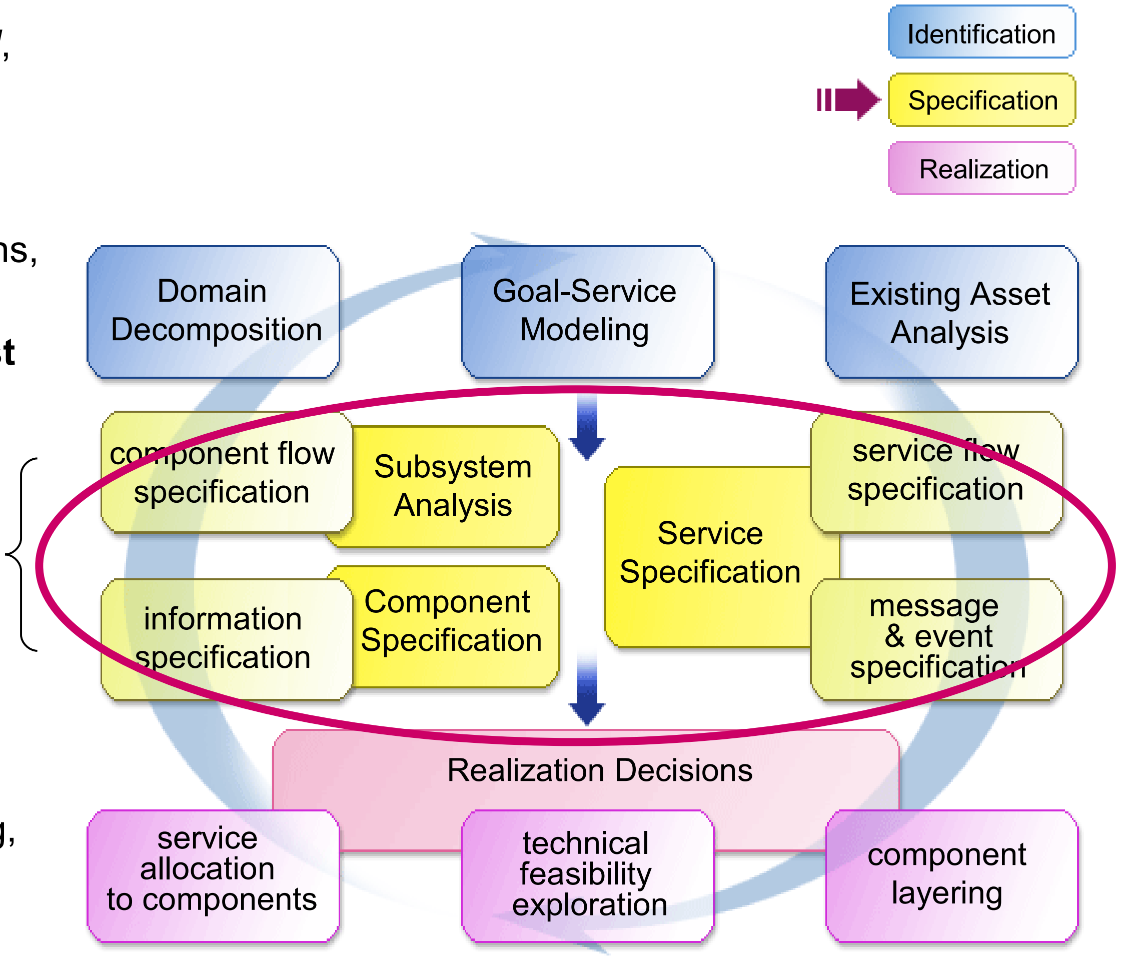
Analysis Process Model (APM)

Claim

Request for benefit to be provided by an insurer to the insured or entitled beneficiaries, under the terms and conditions of a policy.
The Claim includes the following information:
- Claim description
- Claim open date
- Claim reference number
- Claim type
- Claimant name
- Claimee name
- Claim status
- Claim underlying insurance policy
- Claim related Loss event

SOMA Specification Specifies Services, Service Components, and Flows

- Service Specification
  - Elaborates the *Service Model*, for example, service dependencies, composition, non-functional requirements, service message specifications, design decisions, and so on
  - Includes **Service Litmus Test** that "gate" service exposure decisions
- Subsystem Analysis
  - Partitioning into service components that will be responsible for service realization
- Component Specification
  - Detailed component modeling, flow, information architecture, and messages

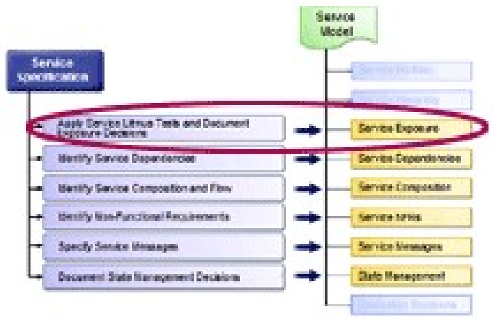


Service Specification Steps

Service Model

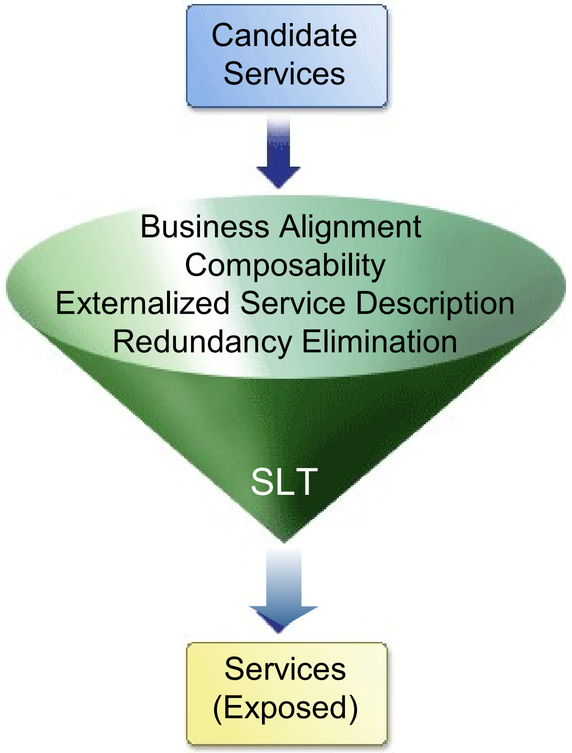
Service Specification	Service Specification defines the dependencies, composition, exposure decisions, messages, quality of service constraints and decisions regarding the management of state within a service.
Apply Service Litmus Test to Make Exposure Decisions	Use **Service Litmus Test** to make service exposure decisions – "From my candidate services, which ones should be exposed?"
Identify Service Dependencies	Detailed review of the service may expose service dependencies on other services or applications that will be used to realize the functionality of the service.
Identify Services Composition and Flow	Review of functional areas and business processes will elaborate the composition of services from other services and their flow to enable the business function. (Service) Flow Specification describes the choreography between services.
Identify Non-Functional Requirements	Use non-functional requirements to specify the desired quality of service.
Specify Service Messages	Identify and specify the format and content of input and output messages of a service.
Document State Management Decisions	Sometimes, the composition of services requires management of state. Document these decisions such as the answer to "What kind of persistence will be used and how will it be enabled?"

Service Litmus Test

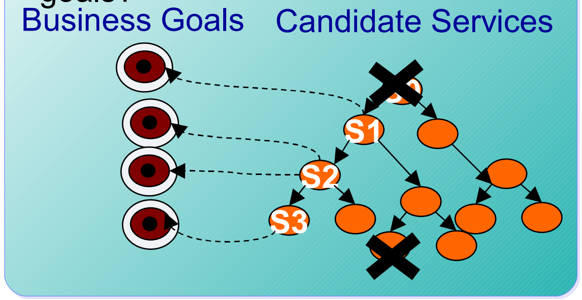

Service Model
Service specification
Apply Service Litmus Tests and Document Exposure Decisions
Identify Service Dependencies
Identify Service Composition and Flow
Identify Non-Functional Requirements
Specify Service Messages
Document State Management Decisions
Service Exposure
Service Dependencies
Service Composition
Service NFRs
Service Messages
State Management

During the Service Specification, we make **service exposure decisions**: “From all the candidate services, which ones should we expose?”

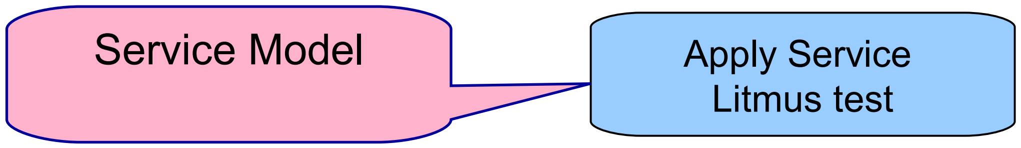
- Not all candidate services should be exposed.
- Every implemented service has costs and risks.
- SOMA “**Service Litmus Test**” helps make exposure decisions.



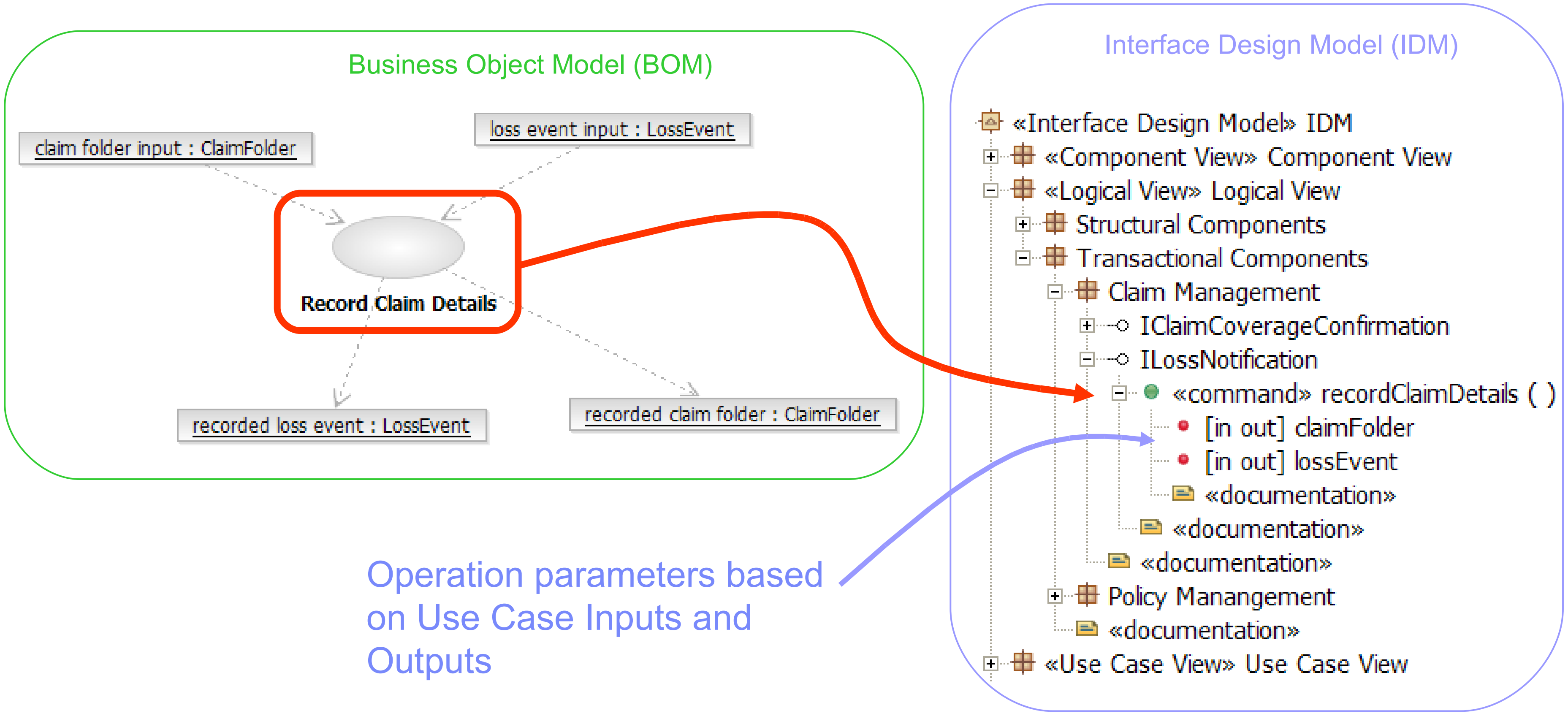
SLT 1-1 : Does the service provide a required unit of business functionality that supports business processes and goals?



Apply service Litmus test



✓Define services within IDM based on BOM use cases


Business Object Model (BOM)
claim folder input : ClaimFolder
loss event input : LossEvent
Record Claim Details
recorded loss event : LossEvent
recorded claim folder : ClaimFolder
Interface Design Model (IDM)

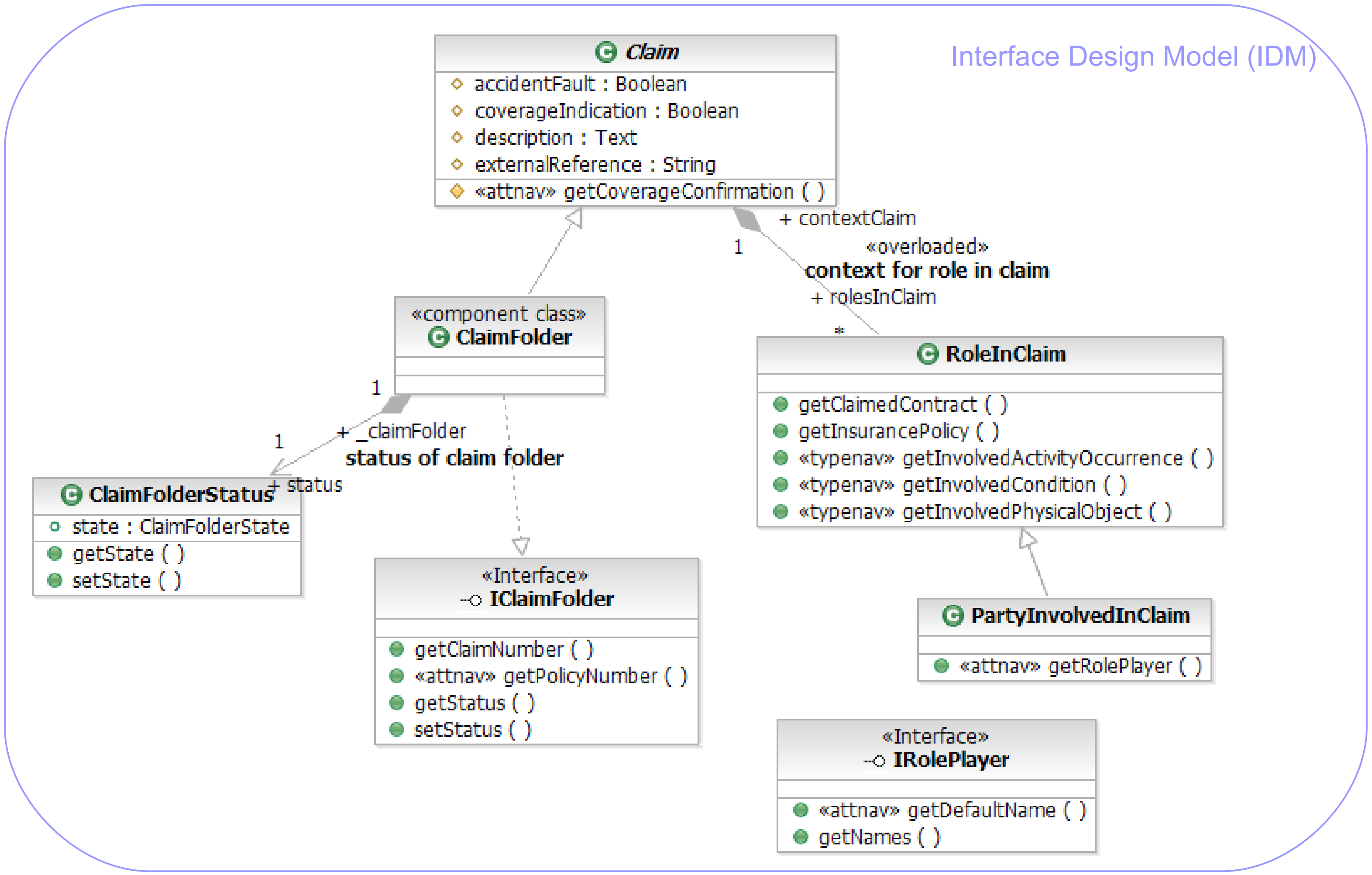
```
«Interface Design Model» IDM
  «Component View» Component View
  «Logical View» Logical View
    Structural Components
    Transactional Components
      Claim Management
        IClaimCoverageConfirmation
        ILossNotification
          «command» recordClaimDetails ( )
            [in out] claimFolder
            [in out] lossEvent
            «documentation»
          «documentation»
        «documentation»
      Policy Manangement
      «documentation»
  «Use Case View» Use Case View
```

Operation parameters based on Use Case Inputs and Outputs

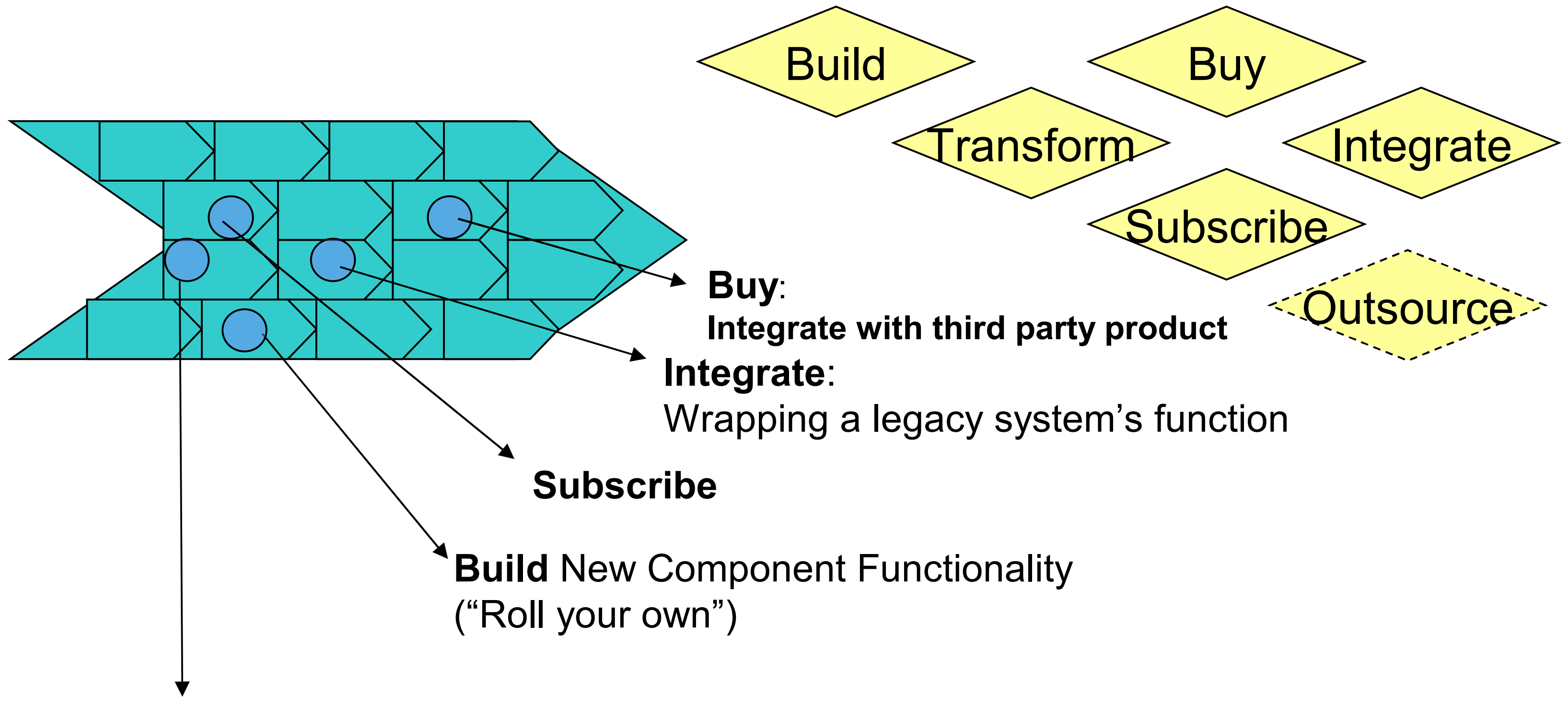
Specify services messages

Service Model

Specify service messages



Service Realization: Services May be Implemented in Many Ways



Buy:
Integrate with third party product

Integrate:
Wrapping a legacy system's function

Subscribe

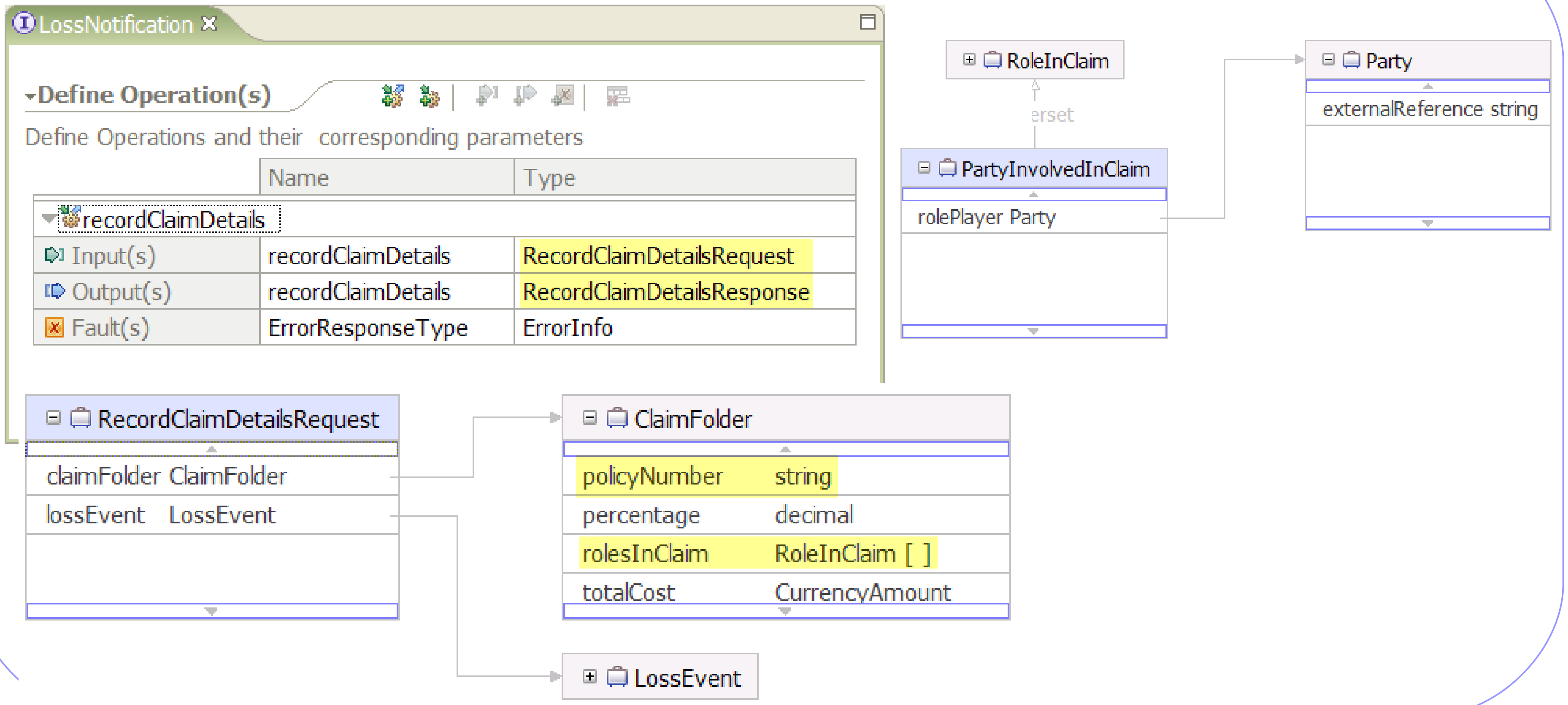
Build New Component Functionality
("Roll your own")

Transform legacy to enable functionality exposure for this service to reuse

Service generation

- ✓Export WSDL/XSD definitions of these IDM services using the generator plug-in
- ✓WSDL includes Request/Response Types
- ✓XSD built based on aggregations and stereotypes

WSDL Interface and XSD (WID)



LossNotification

Define Operation(s)

Define Operations and their corresponding parameters

	Name	Type
recordClaimDetails		
Input(s)	recordClaimDetails	RecordClaimDetailsRequest
Output(s)	recordClaimDetails	RecordClaimDetailsResponse
Fault(s)	ErrorResponseType	ErrorInfo

RecordClaimDetailsRequest

claimFolder	ClaimFolder
lossEvent	LossEvent

ClaimFolder

policyNumber	string
percentage	decimal
rolesInClaim	RoleInClaim [ ]
totalCost	CurrencyAmount

LossEvent

RoleInClaim

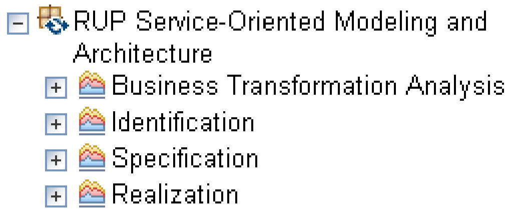
PartyInvolvedInClaim

rolePlayer	Party

Party

externalReference	string

RUP SOMA – Service Infrastructure definition

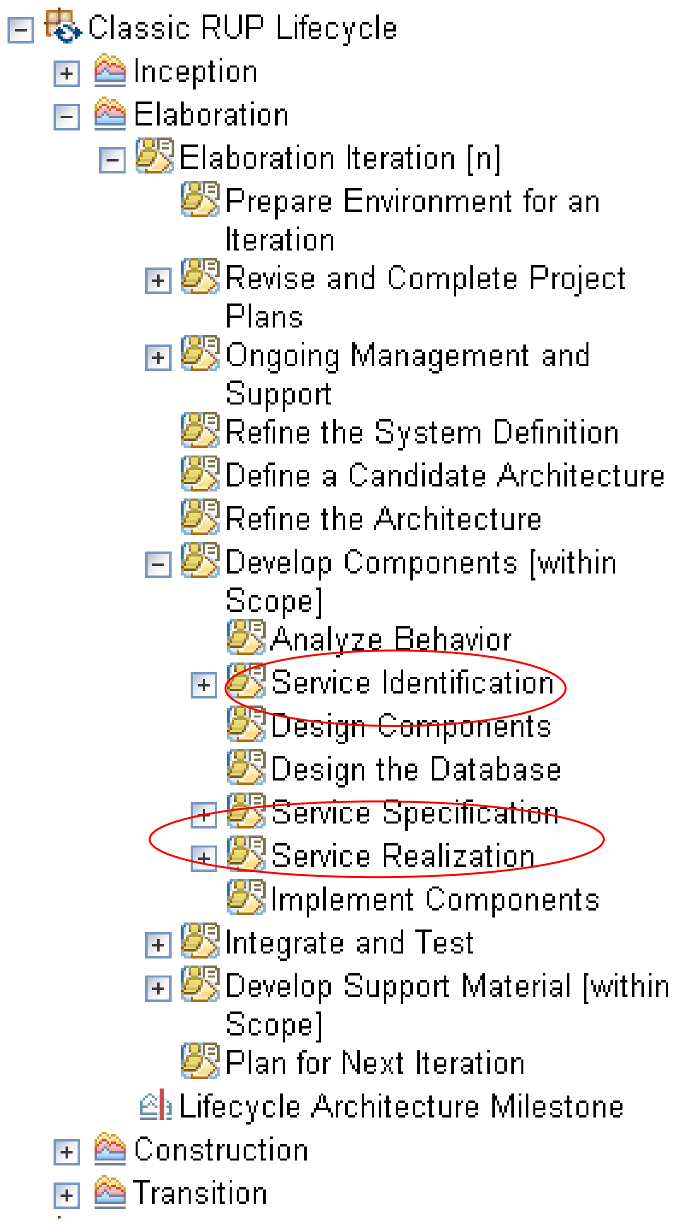

- RUP Service-Oriented Modeling and Architecture
  - Business Transformation Analysis
  - Identification
  - Specification
  - Realization

- Business Transformation Analysis
  - Business Models including Business Processes
- Identification
  - Identify services by analysing business models
  - Confirm viable Services
- Specification
  - Detailed definition of service interfaces and data
- Realization
  - Decide on approach to implement services, including make/buy/subscribe decision
  - Make: How to design a service is not in the scope of this method

Emphasis is on development of services (in a large project or an enterprise)
- Not on services in a wider project context

Classic RUP with SOMA

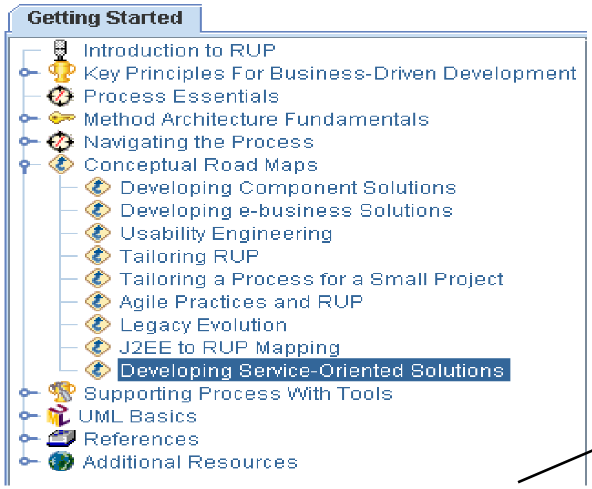
- Classic RUP Phases, Iterations and Activities
  - Now with added SOMA!!
- Thus SOA is integrated into a complete software project development process
- We chose this version as the basis of our work



```
⊟ Classic RUP Lifecycle
  ⊞ Inception
  ⊟ Elaboration
    ⊟ Elaboration Iteration [n]
        Prepare Environment for an
        Iteration
      ⊞ Revise and Complete Project
        Plans
      ⊞ Ongoing Management and
        Support
        Refine the System Definition
        Define a Candidate Architecture
        Refine the Architecture
      ⊟ Develop Components [within
        Scope]
          Analyze Behavior
        ⊞ Service Identification
          Design Components
          Design the Database
        ⊞ Service Specification
        ⊞ Service Realization
          Implement Components
      ⊞ Integrate and Test
      ⊞ Develop Support Material [within
        Scope]
        Plan for Next Iteration
      Lifecycle Architecture Milestone
  ⊞ Construction
  ⊞ Transition
```

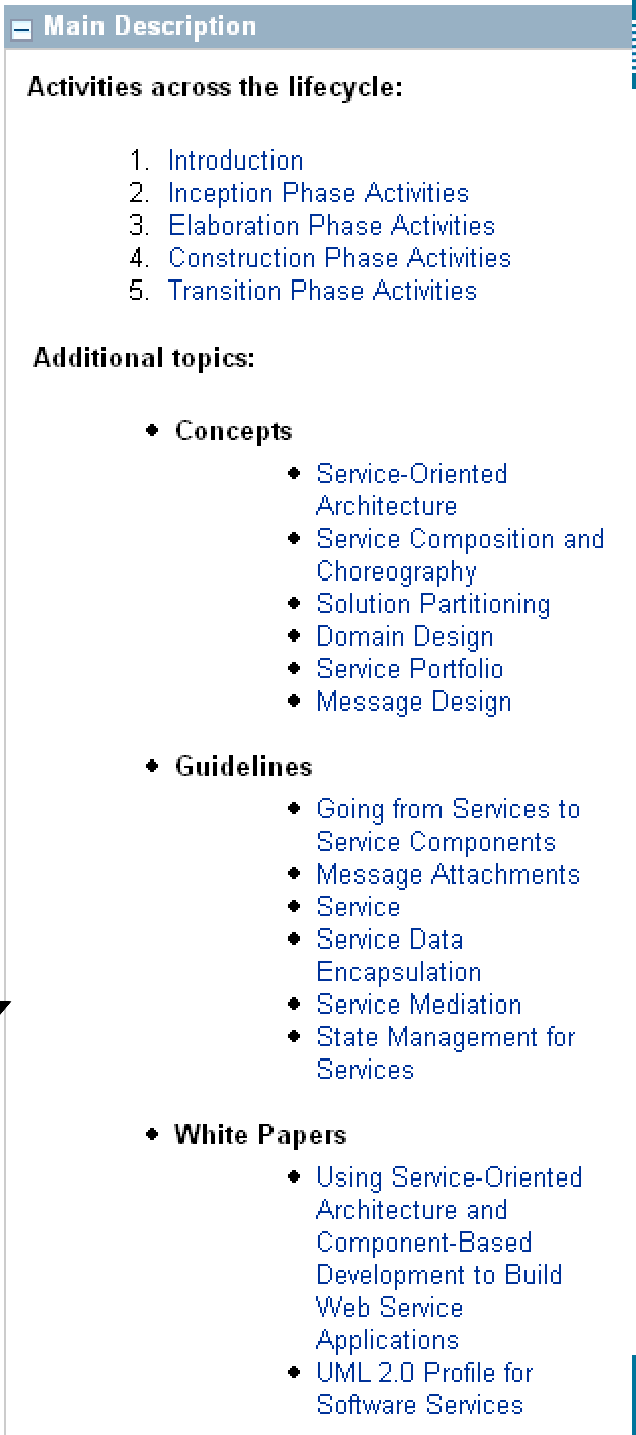
RUP for SOA

- The Rational Unified Process (RUP) describes many useful service specification and design techniques
- A good place to start understanding RUP for SOA is the Developing Service-Oriented Solutions conceptual road map
- RUP for SOA concentrates on the Analysis and Design discipline



Getting Started

- Introduction to RUP
- Key Principles For Business-Driven Development
- Process Essentials
- Method Architecture Fundamentals
- Navigating the Process
- Conceptual Road Maps
  - Developing Component Solutions
  - Developing e-business Solutions
  - Usability Engineering
  - Tailoring RUP
  - Tailoring a Process for a Small Project
  - Agile Practices and RUP
  - Legacy Evolution
  - J2EE to RUP Mapping
  - Developing Service-Oriented Solutions
- Supporting Process With Tools
- UML Basics
- References
- Additional Resources



Main Description

Activities across the lifecycle:

1. Introduction
2. Inception Phase Activities
3. Elaboration Phase Activities
4. Construction Phase Activities
5. Transition Phase Activities

Additional topics:

- Concepts
  - Service-Oriented Architecture
  - Service Composition and Choreography
  - Solution Partitioning
  - Domain Design
  - Service Portfolio
  - Message Design
- Guidelines
  - Going from Services to Service Components
  - Message Attachments
  - Service
  - Service Data Encapsulation
  - Service Mediation
  - State Management for Services
- White Papers
  - Using Service-Oriented Architecture and Component-Based Development to Build Web Service Applications
  - UML 2.0 Profile for Software Services

Thank You