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
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, compositability
- a **programming model** complete with standards, tools and technologies such as Web Services
- a **middleware solution** optimized for service assembly, orchestration, monitoring, and management
Moving to Services-Oriented Solutions – Vision

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

- Security
- Identity
- Permissions
- Firewall
- Registry
- Repository
- Policy Management
- Contract Management
- Flexibility
- Reuse
- QoS
- Orchestration
- Mediation
- Transformation
- Hosting
- Bind Resource

Process: Composition; choreography; business state machines

Service Components

Operational Systems

Integration (Enterprise Service Bus)

QoS Layer (Security, Management & Monitoring Infrastructure Services)

Data Architecture (meta-data) & Business Intelligence

Governance

Packaged Application

Custom Application

OO Application

Transport

Reliability

Channel

B2B

Bind Resource
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
- Understand Risk, Project Costs, and ROI
- Identify and Manage Projects and Resources

Rational RequisitePro
- Document Business Strategy
- Capture Business Goals, Objectives and Requirements
- Trace Business Objectives, Requirements & Create Business Use Case Outlines

WebSphere Business Modeler
- Create Financial Reports & ROI Estimates
- Create, Simulate & Analyze As-Is Business Model
- Create, Simulate, Analyze and Optimize To-Be Business Model

WebSphere Integration Developer
- Choreograph services using BPEL, WSDL, etc.
- Configure Human Task Manager (including Ad-Hoc) & Client
- Use Business Rules, State Machines, Web Services, Adapters, ESB, etc.

Rational Application Developer
- Implement Services, & expose as Web Services
- Develop Portlets (App UI and Monitor)
- Model Services

Rational Software Architect
- Create & Manage System Requirements and Use Cases
- Trace Requirements & Architect System Use Case Realizations
- Model Services

Rational RequisitePro
- Document Business Strategy
- Capture Business Goals, Objectives and Requirements
- Trace Business Objectives, Requirements & Create Business Use Case Outlines

WebSphere Business Modeler
- Create Financial Reports & ROI Estimates
- Create, Simulate & Analyze As-Is Business Model
- Create, Simulate, Analyze and Optimize To-Be Business Model

WebSphere Integration Developer
- Choreograph services using BPEL, WSDL, etc.
- Configure Human Task Manager (including Ad-Hoc) & Client
- Use Business Rules, State Machines, Web Services, Adapters, ESB, etc.

Rational Application Developer
- Implement Services, & expose as Web Services
- Develop Portlets (App UI and Monitor)
- Model Services

Rational RequisitePro
- Document Business Strategy
- Capture Business Goals, Objectives and Requirements
- Trace Business Objectives, Requirements & Create Business Use Case Outlines

WebSphere Business Modeler
- Create Financial Reports & ROI Estimates
- Create, Simulate & Analyze As-Is Business Model
- Create, Simulate, Analyze and Optimize To-Be Business Model

WebSphere Integration Developer
- Choreograph services using BPEL, WSDL, etc.
- Configure Human Task Manager (including Ad-Hoc) & Client
- Use Business Rules, State Machines, Web Services, Adapters, ESB, etc.

Rational Application Developer
- Implement Services, & expose as Web Services
- Develop Portlets (App UI and Monitor)
- Model Services

Rational RequisitePro
- Document Business Strategy
- Capture Business Goals, Objectives and Requirements
- Trace Business Objectives, Requirements & Create Business Use Case Outlines

WebSphere Business Modeler
- Create Financial Reports & ROI Estimates
- Create, Simulate & Analyze As-Is Business Model
- Create, Simulate, Analyze and Optimize To-Be Business Model

WebSphere Integration Developer
- Choreograph services using BPEL, WSDL, etc.
- Configure Human Task Manager (including Ad-Hoc) & Client
- Use Business Rules, State Machines, Web Services, Adapters, ESB, etc.

Rational Application Developer
- Implement Services, & expose as Web Services
- Develop Portlets (App UI and Monitor)
- Model Services

Details
Service Quality Management

Functional and Performance Testing of Web Services from a common interface

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

Project repositories
ClearCase, ClearQuest, ReqPro databases

Promote
Consume

Analyze

Discover

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

1996
- Objectory Process
- Requirements Test Process
- UI Design
- Data Engineering
- UML 1.1

1997
- Rational Approach
- Performance Testing
- Business Modeling
- Configuration and Change Mgt

1998
- Project Management
- UML 1.3
- RealTime

1999
- Rational Process Workbench
- Major addition of content

2000
- Improved Process for independent testing

2001
- Introduction of RUP Platform providing a configurable process framework

2002
- Major addition of tool mentors

2003
- Terminology changes
- Introduction of RUP Base Concepts
- Key Principles for Business-Driven Development
- Delivery processes

2007
- 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

- **Inception**: Lifecycle Objective Milestone
- **Elaboration**: Lifecycle Architecture Milestone
- **Construction**: Initial Operational Capability Milestone
- **Transition**: Product Release

**Scope and Business Case agreement**

**Architecture baselined**

**Product sufficiently mature for customers**

**Customer acceptance or end of life**
What is Rational Unified Process (RUP)?

**Disciplines**
- Business Modeling
- Requirements
- Analysis & Design
- Implementation
- Test
- Deployment
- Configuration & Change Mgmt
- Project Management
- Environment

**Phases**
- Inception
- Elaboration
- Construction
- Transition

**Iterations**
- Initial
- E1
- E2
- C1
- C2
- CN
- T1
- T2
More detail!

- 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

Lessons learnt from previous project and iteration

Corporate guidelines on compliance

Develop and manage **Processes** for performing projects

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

Task: Detail a Use Case

Discipline: Requirements

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.

Relationships

Steps

Review and Refine the Scenarios

Detail the Flow of Events

You should already have an 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.

Storyboard 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 [signal 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 [signal happens], the use case terminates.”
- How does the use case interact with actors? To minimize any risk of misunderstanding say exactly what
Process Example

[Diagram of software development process]

[Table of activities and processes]

[Diagram of software development process]

[Diagram of software development process]
Tools - Authoring, configuring and viewing capabilities
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: 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

**Top-down Analysis**
- Domain decomposition
- Functional Area Analysis
- Process Decomposition
- Variation-Oriented Analysis

**Align Services with Business Goals**
- Goal Service Modeling
- Business directions, metrics, KPIs

**Bottom-up Analysis**
- Existing Asset Analysis
- Industry models as common framework for mapping legacy systems functionality

- CBM Maps aligned with industry models
- Industry process models
- Business classification / externalization of rules

Identification
Specification
Realization
Process decomposition helps identify candidate services

- 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”).

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.
Process Decomposition work products

Process Decomposition

Domain Decomposition

Process Decomposition

Functional Area Analysis

Variation-Oriented Analysis

Process Definition

Use Case Model

Service Model

Legend

Key Work Product

Secondary Work Product
Detailed process analysis

- Customize Control Flow
- Define Business Concepts And Information Flows
- Identify Automated Activities
- Assign Roles

Analysis Process Model (APM)

Process models

Process Definition
Service Analysis with RSM/RSA

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

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

Analysis Process Model (APM)

Business Object Model (BOM) – Boundary of type ClaimFolder
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

<table>
<thead>
<tr>
<th>Service Specification</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identify Non-Functional Requirements</td>
<td>Use non-functional requirements to specify the desired quality of service.</td>
</tr>
<tr>
<td>Identify Dependencies</td>
<td>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.</td>
</tr>
<tr>
<td>Identify Services Composition and Flow</td>
<td>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.</td>
</tr>
<tr>
<td>Apply Service Litmus Test to Make Exposure Decisions</td>
<td>Use Service Litmus Test to make service exposure decisions – “From my candidate services, which ones should be exposed?”</td>
</tr>
<tr>
<td>Specify Service Messages</td>
<td>Identify and specify the format and content of input and output messages of a service.</td>
</tr>
<tr>
<td>Document State Management Decisions</td>
<td>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?”</td>
</tr>
</tbody>
</table>

**Service Specification** defines the dependencies, composition, exposure decisions, messages, quality of service constraints and decisions regarding the management of state within a service.
Service Litmus Test

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?

<table>
<thead>
<tr>
<th>Business Goals</th>
<th>Candidate Services</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>S0</td>
</tr>
<tr>
<td></td>
<td>S1</td>
</tr>
<tr>
<td></td>
<td>S2</td>
</tr>
<tr>
<td></td>
<td>S3</td>
</tr>
</tbody>
</table>

**Service Alignment**

- Composability
- Externalized Service Description
- Redundancy Elimination
Apply service Litmus test

Define services within IDM based on BOM use cases

- Define services within IDM based on BOM use cases

Business Object Model (BOM)

- claim folder input : ClaimFolder
- loss event input : LossEvent
- recorded loss event : LossEvent
- recorded claim folder : ClaimFolder

Interface Design Model (IDM)

- Record Claim Details
- Operation parameters based on Use Case Inputs and Outputs
Specify services messages

Service Model

Specify service messages

Interface Design Model (IDM)
Service Realization: Services May be Implemented in Many Ways

- **Build**
  - New Component Functionality
  - (“Roll your own”)

- **Buy**
  - Integrate with third party product

- **Integrate**
  - Wrapping a legacy system’s function

- **Subscribe**

- **Transform**
  - Legacy to enable functionality exposure for this service to reuse

- **Outsource**
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
RUP SOMA – Service Infrastructure definition

- 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
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

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