Model-driven and Service-oriented Development using Eclipse, J2EE, and Web Services: Standards, Tools, Assets & Patterns

Sridhar Iyengar
IBM Distinguished Engineer
Chief Technical Strategist
Rational Software

Kyle Brown
Senior Technical Staff Member
IBM Software Services for WebSphere
Outline

- Where we are with our Application Development Tooling and MDA?
  - Key Trends
  - Eclipse and IBM strategies
- What are the key challenges facing our customers with SOA and MDA?
- How do we meet the challenges of SOA and MDA?
  - Tooling and runtime directions
  - Assets and Patterns
Some Key Trends in Application Development - 1

- The rise of open source (eclipse) and open standards (W3C, OASIS and OMG) based tools
- From individual tools to a suite of tools – ‘End-end application lifecycle management’
- From multiple tools platform (one for each vendor) to a unified Eclipse open tools platform with support for rich data/metadata interchange
- Enterprise customers moving to integrated tool suites gradually – still many gaps to close
Some Key Trends in Application Development - 2

- The concept of end-end Application Life Cycle Management taking hold
- Evolution of Software Development as a Business Process
  - IBM, Borland and now Microsoft are showing increased focus on Software Development as a process’
  - ‘Software Development as a Business Process’ is now highlighted by IBM, Borland and others
- Vendors announce ‘new & improved’ platforms – with tools customized by role – yet integrated
  - Eclipse Open Source & Open standards based
    - Borland SDP - Software Delivery Platform (2005)
  - Microsoft Visual Studio Team System
- Tools for each user role
  - Business Analyst, IT Architect, Data Architect, Developer, Tester, Manager, Executive etc
Model Driven Development & Deployment

Design/Build

- Business Modeling (BPD,UML)
- IT Modeling (UML, SQL, XSD)
- J2EE/Web Services Development Wrapping Orchestration (J2EE)

Run/Manage

- Deployment J2EE App Svr Web Services
- Management Component Mgmt App Mgmt

Specific metadata Models

- Traceability Links and Transformations
- Serve up models, Components, processes On Demand

Requires a Models, Metadata, Mappings enabled Integration Platform

© 2005 by IBM Corporation
How is the IBM tools platform evolving?

- IBM began the move to the use of EMF as a tools integration metadata platform in 2001
  - We had a couple of iterations as we started with early versions of the OMG MOF 1.3 spec
  - Several optimizations led to creation of EMF and the Ecore model
  - Over time the Ecore and MOF2.0 models have converged
    - The MOF2::EMOF compliance point is now supported by EMF (Import EMOF models to drive EMF generations)
- As we move forward with our tools we are learning more about how to improve the Eclipse tools platform
  - becoming more model driven!
  - In addition to EMF support, we are beginning to add RAS support to desktop tools, then repositories
- There is a way to go on improving the model-driven nature of Eclipse
  - We expect this will iterate as the UML2 and anticipated ‘Graphical Modeling Framework’ (GMF) projects move forward
IBM Modeling Platform Architecture: A simple view
Application Life Cycle Integration Platform

Language Tooling (J2EE, Web Services, Deployment)
Data Tools (RDBMS, XML...)
MDD Tools (Object, Data Modeling, Code generators...)
Domain Specific Tools/Apps...

End to End Application Lifecycle Tooling (Eclipse.org member value add tools)

Eclipse Tools Integration platform (Models, APIs, XML formats...)

Web Services (XSD...)
J2EE (EJB, JSP...)
MDD/MDA (UML2, GMF...)
Testing TPTP
ETC.

EMF
GEF
JDT/CDT
RCP
ETC.

Eclipse Core

Code/Artifact Repositories, Management Tools (Eclipse.org member value add tools)

© 2005 by IBM Corporation
Where does this leave us?

- We have a good story for development of standard Java and J2EE applications through the lifecycle using MDA approaches
  - Integration and interoperability with other vendors and with the free Eclipse tools will only get better over time
- But this is only the easiest part of our problem…
  - We still have to address the issues brought on by SOA
60 to 80 percent of the average company's IT budget is spent on maintaining existing applications.¹

- Overwhelming complexity
- Relentless time-to-market pressure
- Geographically distributed teams

¹“Optimizing Enterprise IT” (Intelligent Enterprise, 2/24/2005)
So what is Service-Oriented Architecture?

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.
The world of SOA

Composite Application
A series of integrated services that support a business process built on an SOA
Business Driven Development …Roadmap

Rational RequisitePro
- Understand Risk, Project Costs, and ROI
- Identify and Manage Projects and Resources

WebSphere Business Modeler (4Q05)
- Create, Simulate & Analyze As-Is Business Model
- Create Observation Model with KPIs & export to Monitor
- Create, Simulate, Analyze and Optimize To-Be Business Model
- Create Financial Reports & ROI Estimates

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 Data Architect (4Q05)
- Model Relational Database Schemas
- Data Mapping

Model & Implement Services, & expose as Web Services
- Trace Requirements & Create System Use Case Realizations
- Develop Portlets (App UI and Monitor)

Rational Software Architect
- Architect
- Java Developer
- Portal Developer

IBM Rational Team Unifying Platform
- Business Operations Analyst
- Monitor
- Deploy/Run
- WebSphere Process Server
- WebSphere Portal
- WebSphere Business Monitor

Run-time Statistics
Observation Model

BPEL UML WSDL EAR
Business Driven Development…Reality

Customer Purchase Process

- #1. Poorly shared business context
- #2. Elaboration / Refinement upwards not possible
- #3. Poor Traceability/Notification
- #4. Too many tertiary UIs

Determine Applicant Eligibility Service Elaboration

- Business Goals
- Business Use Cases
- Business Opportunities
SOA Programming Model

- Programming Model Simplification
  - Complexity in the programming model shows through as complexity in development, debug, test, systems administration and monitoring

- Service Data Objects (SDO) - Universal data model for data access (pass data to & from applications)
  - On a standards path JSR 235 championed by IBM and BEA

- Service Component Architecture (SCA) – Component model for WebSphere
  - Required and Offered interfaces (Java interfaces or WSDL interfaces)
  - Component and Subsystem wiring

- BPEL4WS – Scripting of components, workflow
SCA Service-Oriented Component Model

- Significant simplification over J2EE
- Abstracts stateless session EJBs, web services, POJOs, BPEL4WS processes, database access, JCA access, etc.
- Separates “business logic” from “infrastructure logic”
- Covers both usage and development of services/components
- A uniform model for application programmers and for tools
  - Enables advanced, domain-specific, task-oriented tools
- Uniform programming model for “programming in the small” (application development) and “programming in the large” (application integration)
  - Integrates with and provides programming model for the Enterprise Service Bus
- Simple declarative model for importing services that are implemented in other service technologies (e.g., EJB, Web Service, JCA, etc.)
- Uniform client programming model for accessing services whether an SCA service or an imported service.
## SDO Data Abstraction

<table>
<thead>
<tr>
<th>Model</th>
<th>Model</th>
<th>API</th>
<th>Data Source</th>
<th>MetaData API</th>
<th>Query Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDBC Rowset</td>
<td>Connected</td>
<td>Dynamic</td>
<td>Relational</td>
<td>Relational</td>
<td>SQL</td>
</tr>
<tr>
<td>JDBC Cached Rowset</td>
<td>Disconnected</td>
<td>Dynamic</td>
<td>Relational</td>
<td>Relational</td>
<td>SQL</td>
</tr>
<tr>
<td>Entity EJB</td>
<td>Connected</td>
<td>Static</td>
<td>Relational</td>
<td>Java Introspection</td>
<td>EJBQL</td>
</tr>
<tr>
<td>JDO</td>
<td>Connected</td>
<td>Static</td>
<td>Relational, Object</td>
<td>Java Introspection</td>
<td>JDOQL</td>
</tr>
<tr>
<td>JCA</td>
<td>Disconnected</td>
<td>Dynamic</td>
<td>Record-based</td>
<td>Undefined</td>
<td>Undefined</td>
</tr>
<tr>
<td>DOM and SAX</td>
<td>Disconnected</td>
<td>Dynamic</td>
<td>XML</td>
<td>XML InfoSet</td>
<td>XPath, XQuery</td>
</tr>
<tr>
<td>JAXB</td>
<td>Disconnected</td>
<td>Static</td>
<td>XML</td>
<td>Java Introspection</td>
<td>N/A</td>
</tr>
<tr>
<td>JAX-RPC</td>
<td>Disconnected</td>
<td>Static</td>
<td>XML</td>
<td>Java Introspection</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**SDO**
- Common API
  - Disconnected support
  - Relationship integrity
  - Generated and dynamic interfaces
- Validation
- Rich meta-data
- XML and XML schema
- XPath navigation
- SCA data mediators
Component = “Service A”

Implementation

References: Define service requirement

Java Interface
WSDL Port Type

“BPEL”
“Java”
“Adaptive Entity”
“Business Rule”
“Human Task”
“Selector”
“Mediation”

Parameters SDO
Service B Reference
Service C Reference

Interface
SCA – Module Structure

Exports: Provide services externally

Imports: Use external services

Module: Unit of deployment

Wires: Resolve references

Service Export

Standalone Reference

Service Import

Service A
Impl = “BPEL”

Service B
Impl = “Java”

Non-SCA Artifact

JSP
XSD
WSDL
Java Class
......

binding = “Service”

binding = “Web Service”

binding = “J2C”

binding = “JMS”

binding = “...”

binding = “S-SEJB”

binding = “Data Mediator”

binding = “J2C”

binding = “JMS”

binding = “...”

Exports:
Provide services externally

Imports:
Use external services

Module: Unit of deployment

Wires: Resolve references
Tomorrow’s Development Tooling Goals

- Simplification
- Better assist the “business-oriented” developer
- Shift to SOA and a simplified programming model
Service Component Tooling

- Abstract the operation creation task for a service component.
- Links interface operations with implementation.
- Easily set references to other service components and then use it in the coding of the implementation.
- Service implementation can be in many different flavors:
  - Java
  - BPEL
  - DataAccess (specialized)
- Supported service interfaces will be Java and WSDL.
- Easily set Quality of Services (e.g., security, transactions, etc.).
- Support publish and discovery of service components as reusable assets.
System Creation

Leverage Enterprise topology patterns for creating new systems.
Deployment Modeling

- Captures system structure and application requirements.
- Captures topology and associated supported specifications.
- Captures the logical structure of your data center nodes or servers.
- Allow a mapping of the system structures to the logical servers.
- Constraint validation checking between application structures and logical server supported specifications.
- Portable to Tivoli provisioning tools to further customize.
So where are we?

- Our development tools are becoming SCA-aware
  - We are also addressing issues of deployment modeling in the SOA IF
- But this is still only part of the problem...
  - We can build applications the way WE want, but is this the way our CUSTOMERS want their applications?
Patterns and Assets

- What are patterns and why do we care?
  - Patterns, recipes, templates & taxonomy
- The Rational Patterns & Repository Architecture
  - The Reusable Asset Specification (RAS) & Repository
  - Authoring & utilizing patterns/templates in RSA
- A call to action
Motivation for IBM focus on patterns & assets

- Addressing the most difficult issues
  - Capturing and tooling best practices found in the field
  - Capturing common customer architectural decisions
  - Enabling our consultants and customers to show us the important problems

- Consumability
  - Complexity of products
    - Testing, outside-in design, ease of use mitigate some of this complexity
  - Complexity of customer problems
    - Recipes, patterns, and templates mitigate some of this complexity

- Moving forward with parallel delivery strategies
  - Base tooling/runtimes
  - SOA IF
  - Rich patterns portal on dW
What’s An Asset?

- When tooling for assets, need a consistent representation on which to build

- An Asset is
  - a collection of Artifacts
  - which provide a solution to a problem
    - for a given context
    - with rules for usage
    - and variability points

- What are Artifacts?
  - Workproducts from the **software** process
    - Requirements, Models, Source code, Tests, and so on…
  - Workproducts from the **solution** process
    - Pricing, terms and conditions, hardware, and so on…

- Kinds of assets
  - Components, patterns, web services, frameworks, templates, …
Reusable Asset Specification (RAS)

- Provides a standard way to describe and package assets
- Reduces the friction on development transactions
  - Thru standard, consistent packaging
- RAS provides the organization and structure of information and artifacts for an asset
- RAS enables tooling to automate many asset workflows
- RAS is used to package many kinds of assets including components, services, and pattern implementations, …
What’s A Pattern?

- A pattern is a kind of asset
- A pattern describes a solution to a recurring problem in a context
- Patterns ease the sharing of proven solutions

**Concepts**

Pattern **describes** a solution to a recurring problem for a context

**Pattern Specification**

SOA Requester Side Cache
Pattern Specification

- Link To Pattern Spec Website
- Requester Side Cache Pattern Specification Document

**Example**

SOA Requester Side Cache
Pattern Specification

**Pattern Implementation**

Automation
Component/
Service/
RSA Transform, Pattern/
Template/ …

Pattern Implementer develops
the pattern using the Pattern
Specification and puts it into a
RAS repository

**Implemented as**

- RSA Transform
- Pattern Template
- Component/Service/
- RSA Transform, Pattern/
- Template/ …
Patterns At Many Levels Across Many IBM Teams

- **Business**
  - Business Model/Process Patterns
  - Cross-Industry Business Patterns

- **Operational Architecture**
  - Patterns for e-Business (P4eb)
  - Process integration patterns (P4eb)
  - Data integration patterns (P4eb)
  - SOA & Web Services patterns
  - Enterprise integration patterns
  - System management patterns (P4eb)
  - Information integration patterns (P4eb)

- **Application Architecture**
  - Cross-industry business application patterns (P4eb)
  - LWP templates
  - SOA design patterns
  - Design patterns (ISSW, ODSD)
  - Data model & Grid patterns
  - Legacy transformation patterns

- **Deployment**
  - Data management patterns
  - Application & infrastructure deployment patterns
  - Testing patterns
  - Hardware patterns

**Systematic: top-down approach to building systems**

**Opportunistic: start at any point**
Authoring And Implementing A Pattern

- There are multiple skill levels to target these proven solutions for development time
- When deciding which approach to use, the pattern implementer evaluates the
  - skills of the target user
  - interactive vs. batch style
  - nature of the problem to be solved
- Although this puts greater effort on the implementers, it improves the consumability for the target development users

![Diagram of SOA Requester Side Cache Pattern Implementation](image)

The pattern implementation outlined above could produce the same, runtime service or component. In this case Service X is created.

The Design Pattern Toolkit (DPTK), on alphaWorks, is being merged with JET as JET2 in RSA.
Recipes reference the assets (services, components, RSA patterns, …) to use and the sequence to use them to solve larger-grained problems
- A recipe is a special kind of asset which composes other assets

Recipes improve consumability by reducing the level of effort to
- Find the assets which solve a particular problem
- Figure out how to make several assets work together to solve a particular problem
Tools and Repository For Pattern Implementations

Author Pattern Implementations & Recipes

Rich Client

- Analyst
- Architect
- Developer
- Tester
- Deployment Manager

Thin Client

Asset Portal

Browse, Reuse Solutions by
- Role
- Industry
- ...

Browse, Repository Service

Manage Assets

Repository Service

RAS Asset Repository

- SOA Integ Frmwk
- Business Pattern Implementations
- Application Arch Pattern Impls
- Operational Arch Patterns Impls
- Deployment Pattern Impls

Recipes

- Recipe 1
- Recipe 2
- Recipe 3

EMF/RAS

Financial Services

Retail

Insurance

- Business Delegate
- Session Facade
- Data Access Object
- Message Facade

Project Manager

Executive

Rational Team Unifying Platform

Rational Software Modeler

WebSphere Business Integration Modeler & Monitor

Rational Software Architect

Rational Web App Developer

Rational Performance Tester

Tivoli Configuration Manager

Tivoli Monitoring

Repository Service

Customer Extensions

3rd Party ISV Tools

Factory Patterns

Create/Reuse

Browse

© 2005 by IBM Corporation
Pattern, Recipe, RAS Tools: Next Steps

- The asset repository should support a spectrum from lightweight to enterprise-level capabilities
  - Team-level Repository
    - Rational has developed a RAS repository J2EE service deployable on WAS for customers and partners
    - This service is also hosted on developerWorks
  - Service Oriented Asset Repository (SOAR)
    - This repository is being developed for IBM services teams
  - Enterprise-level Repository
    - This is supported through partners such as Flashline, and LogicLibrary

RAS and other components are moving to a common component layer, allowing multiple SWG products to implement patterns.
Building Pattern Community

- Pattern Solutions main page on developerWorks
  - Help drive RSA sales
  - Create market buzz
- Pattern Enablement activities:
  - Working with Rational Media Relations on press articles
  - DeveloperRelations, DevX
  - Low Touch publicly available RAS repository to promote ease of asset sharing
- Amazon.com style review
  - aka “junk yard” approach
  - Vote your most favorite pattern
Patterns : How do you get involved?

- Learn to use existing frameworks for RSA and how to reuse existing patterns on Developerworks
  - Contribute to the development of new frameworks and tools (e.g. JET II)
- Author your own pattern solutions for your domain of expertise
  - Work with services to find and implement patterns
  - Publish these on Developerworks
- Participate in the SWGAB Patterns Workgroups
  - Implementation, Specification Governance
- Help IBM be more competitive by expanding the reach of your expertise
Summary

- **We’ve seen:**
  - How our MDA and Development tools have evolved hand-in-hand with the Eclipse foundation
  - Where we are and where we are going to address the challenges posed by SOA
  - Where we need to go to address the issues raised by asset-based development
Trademarks

- **OMG™, MDA™, UML™, MOF™, and XMI™** are trademarks of the Object Management Group
- **IBM™** is a trademark of the IBM Corporation
- **Java™ and J2EE™** are trademarks of SUN Microsystems
- **XML™** is a trademark of W3C
- All other trademarks belong to their respective organizations