Using Model-Driven Development & Eclipse Technology to implement SOA

Tas Frangoullides
Overview

• What is Model-Driven Development (MDD)
• Challenges of SOA
• Our Solution
• Getting There
• Summary and Tips
• Questions
What is...

**MODEL-DRIVEN DEVELOPMENT?**
Model-Driven Development

Model → Generators → Source Code, Data Schemas, Presentation Artefacts
Benefits of MDD

Can target multiple platforms

Raises the level of abstraction

Reduces development times
Implementing Service-Oriented Architecture

**THE CHALLENGES**
Challenges of SOA

• Promoting effective communication of details between business-IT teams
• Ensuring semantic interoperability between enterprise services
• Successful adoption of standards and best-practice
• Retaining independence from platforms and vendors
• Maintaining a good degree of architectural agility
• Getting developers up to speed and helping them adopt technology, best-practice and new platforms
Applying Model-Driven Development and Eclipse Technology

OUR SOLUTION
Enterprise Data

Within an Enterprise Application

Outside an Enterprise Application

Enterprise Web Service maps between the two worlds of data
Common Integration Model

- Representative of an *Enterprise* business domain
- Common vocabulary between systems and people
- Used to constrain the structure of data used in Enterprise Services
- Should be *influenced* by multiple business units
- Should *not* be adopted internally by systems
From domain model to service

- **Domain Model**: Describes business entities and the relationships between them.
- **Message Model**: Describes units of information which may be communicated between systems.
- **Service Model**: Describes SOA service interfaces in a platform-independent manner.

**Constraint**: Use by reference

**Global Website**

**Deployable Web Service**

**Use by reference**

**Message** + **Model**

**Constraint**

**Domain**

**Website**

**Java Bindings**

**XSD**

**Deployable**

**Web Service**
Web Service Generation

Service Model

Eclipse Web Project

- WSDL
- XML Schemas
- Java Binding for XML
- Java Interface & Implementation Skeleton
- Deployment Descriptor
- Ivy Configuration (Dependency Management)
- Ant build.xml (for Jumpstart)
Web Service Generation

- Service Model
- Eclipse Web Project
  - X-Fire Project
    - Services Framework Configuration
    - Spring Configuration
  - Sonic ESB Project
    - Sonic Process
    - Deployment Manager Model Files
Benefits

• Use of models dramatically improved communication across the enterprise
• Agile Architecture
• Platform Independence
• Conformance to standard data model
• Development times for new services significantly improved
EXAMPLE MODELS
Domain Model

- Institution
  - Long Name: String
  - Short Name: String
  - Formation Date: Date
  - Description: String
  - Status

- Instrument

«Domain Model»
CDM
Domain Model

- Institution
  - Long Name: String
  - Short Name: String
  - Formation Date: Date
  - Description String
  - Status
  - Issuer: 0..1

- Instrument
  - Long Name: String
  - Short Name: String
  - Effective Date: Date
  - Status: Instrument Status
  - Current Price
    - Time: DateTime
    - Amount: Money

- Instrument Status
  - Not Yet Issued
  - Issued
  - In Default
  - Due
  - Matured

CDM
Message Model
What do we do with the models?

**THE SOA TOOLKIT**
The SOA Toolkit

- Information Architect
- Enterprise Modellers
- Domain Model
- Service Model
- Business Analyst
- Developers
- Message Model
- Messaging Team

• Analyse & Validate Model
• Manage common data types
• Publish Web Site (Continuous)

• Model Services
• Generate Web Services
• Apply standards and best-practice
• Abstract the platform
• Develop Java Code
• Deploy Services

• Analyse & Validate Model
• Manage common data types
• Publish Web Site (Continuous)

• Create derived models
• Analyse and synchronise changes between domain and message model
• Generate XML Schemas
Getting there

THE STORY
The Story - Overview

POC: UML to XSD

Prototype: Service Project Generator

Release 1.0

Release 2.0

Issues with Sonic ESB

2007

Q4

Q1

Q2

Q3

Q4

Project goes live with generated Web Service

Sonic ESB Introduced

2008

Q1

Q2

Q3

Q4

Release 2.1
The proof-of-concept

FROM UML TO XSD
POC: The Stack

- UML to XSD POC
  - UML API
  - UML Meta-model
  - UML to Ecore Conversion

- Eclipse Modeling Framework
  - Meta-Modelling Language
  - Code Generation
  - XML Persistence
  - XML Schema Generation

- Eclipse
  - Plug-in Architecture
  - UI Framework
POC: UML to XML Schema

Eclipse UML Model → Custom UML2Ecore Converter → Ecore Models → Custom Ecore Schema Builder → XML Schemas
POC: UML to XML Schema

- Customise Data Mapping
- Experiment with XML Choice

- Suppress Ecore annotations
- Standardise Namespace
- Reference XML Data Types
CHOOSING A UML MODELLING TOOL
The Contenders

MagicDraw

Enterprise Architect

Rational Software Modeler
Why RSM?

- Eclipse Based
  - Extensible
    - Plugin Development Environment
  - Integrated
    - Java Development
    - XML Editing
    - Version Control
  - Developer Friendly
- UML API
- Open-Source Foundations
  - EMF
  - UML2
  - GMF
  - TPTP
More Eclipse Frameworks

WEB SERVICE GENERATION
Web Service Generation Stack

- Service Project Generator
  - UML2
  - EMF
  - JDT
  - Eclipse
  - UML Profiles

- Web Tools Platform

- Eclipse
  - UI Framework
  - Resources Framework
  - Builders

- JDT
  - Classpath Handling
    - Java Compiling
  - Templates (JET)
    - Source Code Merging

- Web Projects & Facets
  - WSDL
  - Deployment Descriptors
  - ...
The Generation-Chain Model

**Using MDD to Develop MDD Tools**
The Generation Chain

1. Expand/Apply Patterns
2. UML to Ecore
3. Ecore to XSD
4. Generate Java Bindings
5. Generate Web Project
6. Generate Platform Specific Artefacts
7. Retrieve Dependencies
Generation Chain Meta-Model
Generating an API

Generation Chain Model (UML)

UML Importer

Ecore Model & EMF GenModel

EMF Code Generator

Java API with XMI Persistence

Eclipse Model Editor
Creating and Synchronizing

MESSAGE MODELS
Creating a Derived Model

- Palette is constrained by domain
- Relationships limited to those available in Domain Model
Creating a Derived Model

Message Modeling

Rational Software Modeler

UML2

GMF

EMF

GEF

Eclipse

Graphical Modeling Framework
- Integrated Platform for modelling tools
- Unites GEF and EMF
- Notational Meta-Model
- Model-View Separation

Graphical Editing Framework
- Drawing Canvas
- Palettes & Tools

- Message Palette
- Domain Element Selection Dialog
- Edit Helpers
Synchronisation Problem

Domain Model

Constraint

Message Model
Synchronisation Problem

- Model elements renamed
- Model elements deleted
- Types change
- Packages are restructured
- Traces break
Synchronising Models

- Rational Software Modeler
- BGI Model Analysis
  - Detect and automatically fix broken traces
  - Report items requiring human decision making
- TPTP
- Eclipse
- Test & Performance Tools Platform
  - Static Analysis
    - Reporting
    - Quick Fix Support
Keeping the models pure

THE ‘IDENTIFIABLE’ PROBLEM
The ‘Identifiable’ Problem

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
    targetNamespace= "http://gxml.bglobal.com/Identifiable">
  <complexType name= "Identifier">
    <attribute name="type" type="string"/>
    <attribute name="value" type="string"/>
  </complexType>
</schema>

<Instrument>
  ...
  <Identifier type="SEDOL" value="0263494"/>
  <Identifier type="CUSIP" value="037833100"/>
  ...
</Instrument>
```
The ‘Identifiable’ Problem

Just model it?

✗ Pollutes business model with technical concerns
   ✗ Identifiable is not a business concept (in this case)
The ‘Identifiable’ Problem

Does not change structure of business model

Downstream Implementation can change

<schema xmlns="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://gxml.bglobal.com/Identifiable">
  <complexType name="Identifier">
    <attribute name="type" type="string"/>
    <attribute name="value" type="string"/>
  </complexType>
</schema>
Summary and tips

WRAPPING UP
Benefits of Model-Driven

• Models improve communication
• Agile Architecture
• Platform Independence
• Time-to-market
Benefits of Eclipse Technology

• Eclipse enables a seamlessly integrated solution
• Modeling Projects provide all the scaffolding for Model-Driven Development
• Reduces the gap between the generators and platforms
• UI, Projects, Resources – all taken care of
• Developers are comfortable with Eclipse Plugins
• Keeps getting better
Thinking of giving this a go?

**Top 10 Tips**
Tip #1

Take an iterative and incremental approach
Tip #2

Get buy-in from key stakeholders and groups
Tip #3

Use what is already out there...

...but be wary of in-the-box solutions
Tip #4

Don’t pollute your business models
Tip #5

Have at least one MDD veteran
Tip #6

Version your models along with everything else
Tip #7

Manage your inter-model dependencies and monitor them for leakages
Tip #8

Automate where possible
Tip #9

Work with project teams and reduce their risk...
Tip #10

and if there is failure..

Share It
Questions?

Thank You

tas@model-driven.co.uk