Using **Model-Driven Development** & **Eclipse Technology** to implement SOA

Tas Frangoullides

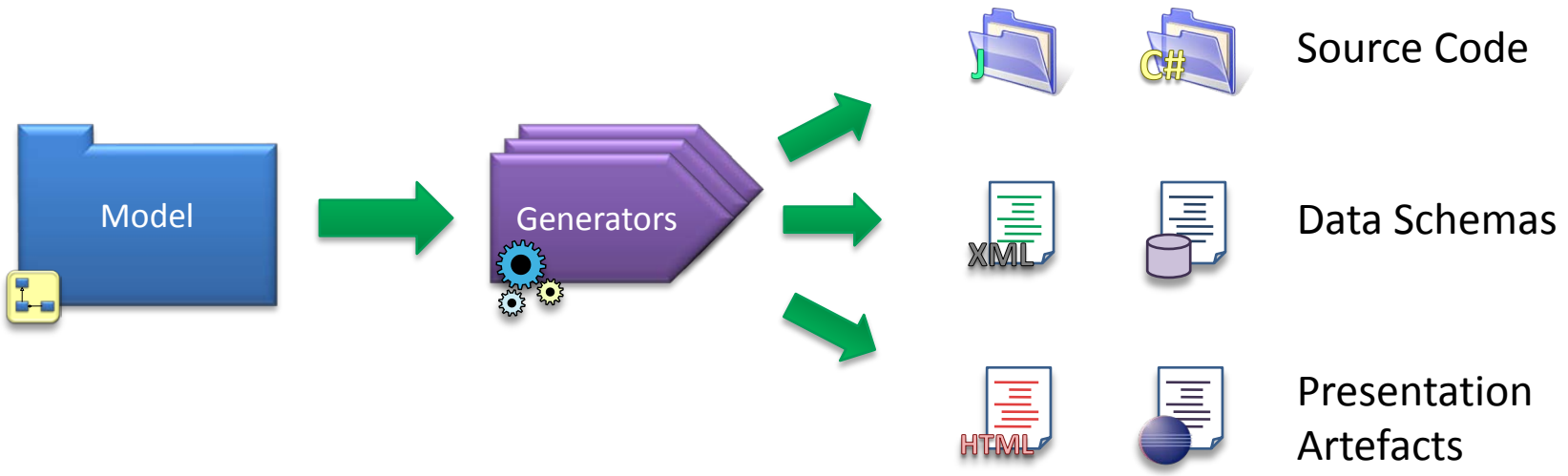
Overview

- 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



Benefits of MDD

Raises the level of abstraction

Can target multiple platforms

Reduces development times

Implementing Service-Oriented Architecture

THE CHALLENGES

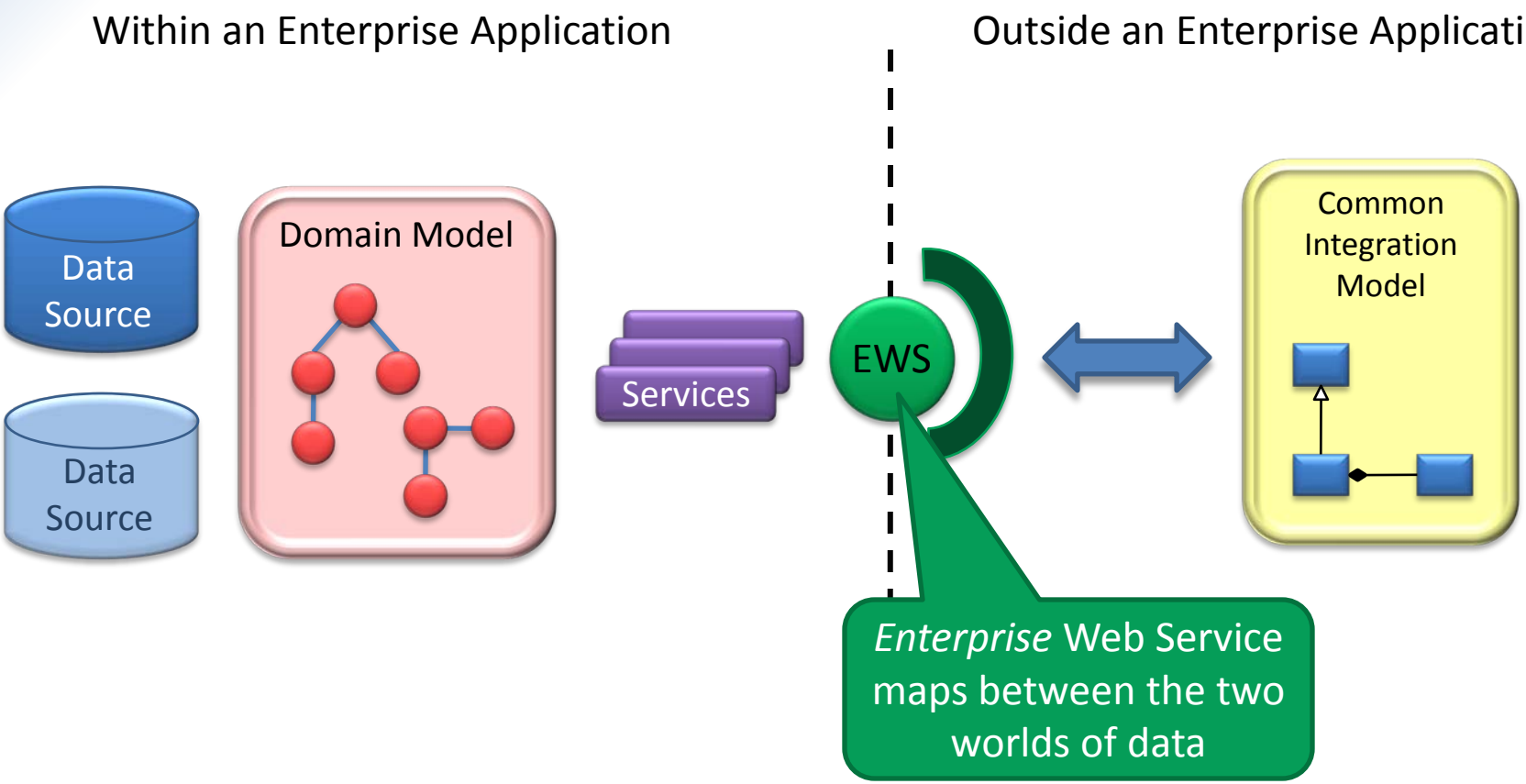
Challenges of SOA

- 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

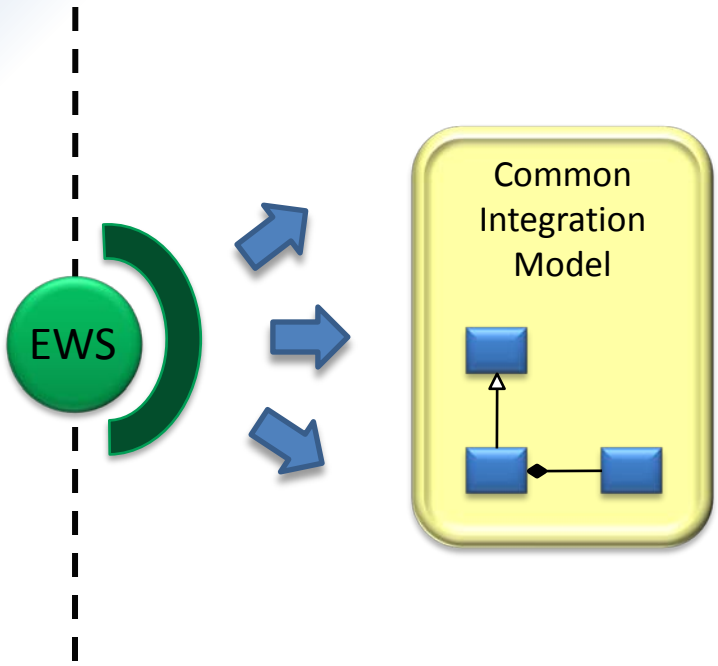
Applying Model-Driven Development and Eclipse Technology

Our Solution

Enterprise Data



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

Describes business entities and the relationships between them

Domain Model

Constraint

Describes units of information which may be communicated between systems

Message Model

Use by reference

Describes SOA service interfaces in a platform-independent manner

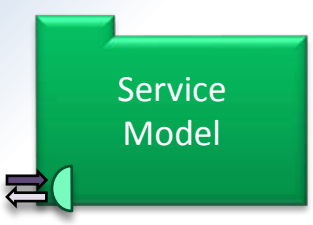
Service Model

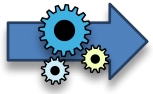
Global Website

XSD + Java Bindings

Deployable Web Service

Web Service Generation





Eclipse
Web Project

WSDL

XML Schemas

Java Binding for XML

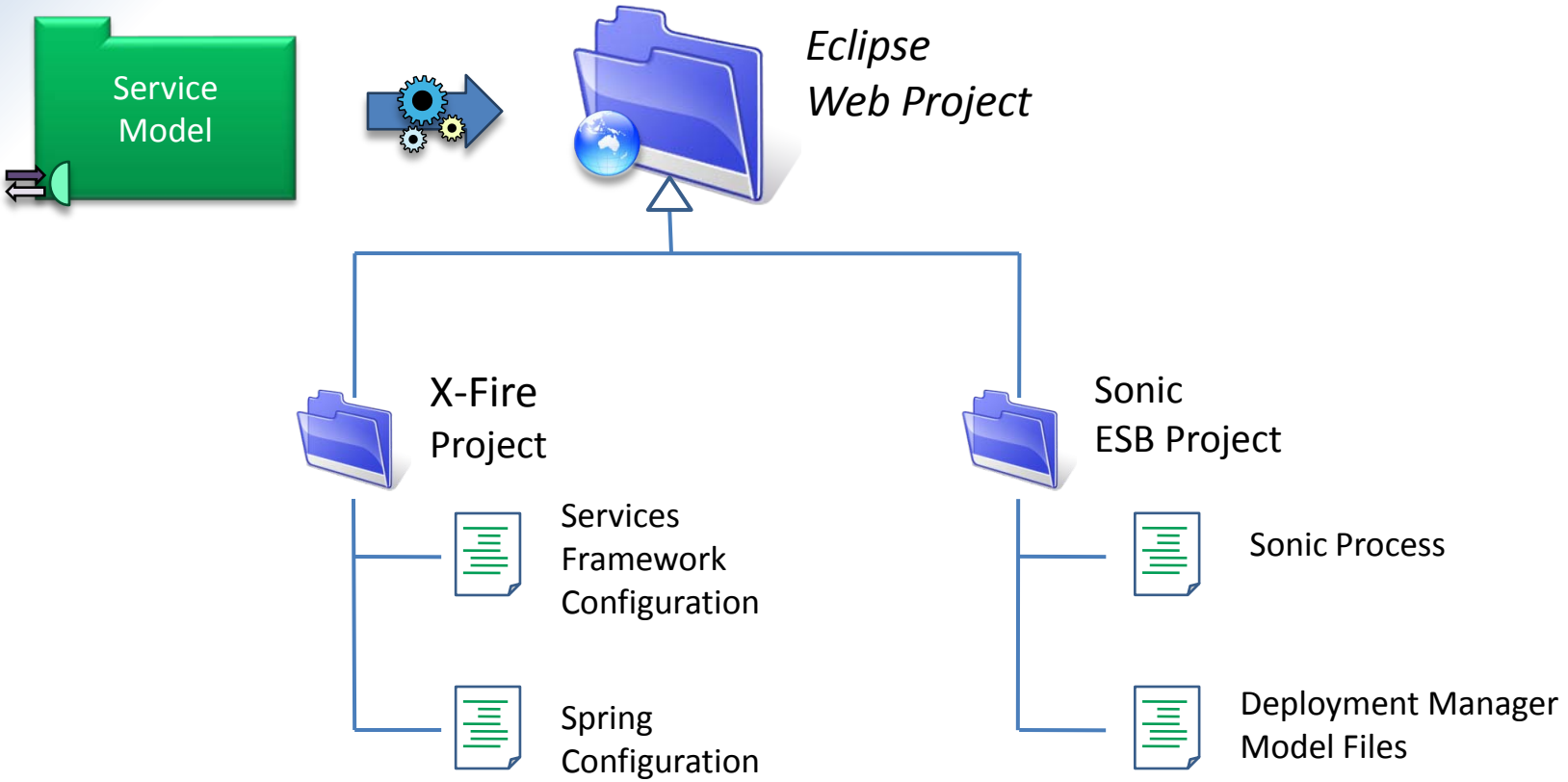
Java Interface & Implementation Skeleton

Deployment Descriptor

Ivy Configuration (Dependency Management)

Ant build.xml (for Jumpstart)

Web Service Generation

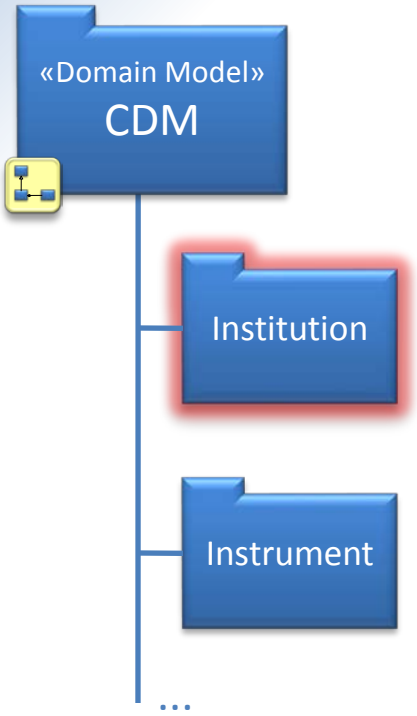


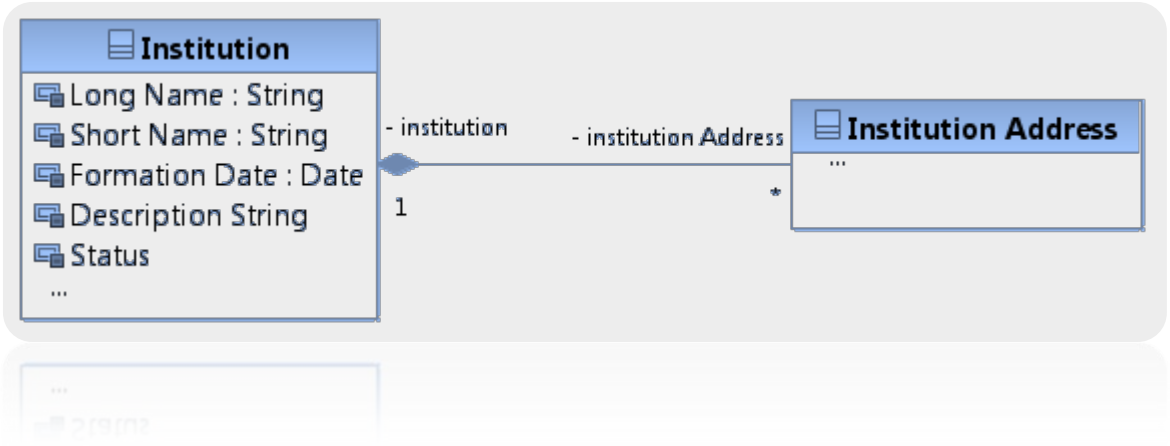
Benefits

- 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

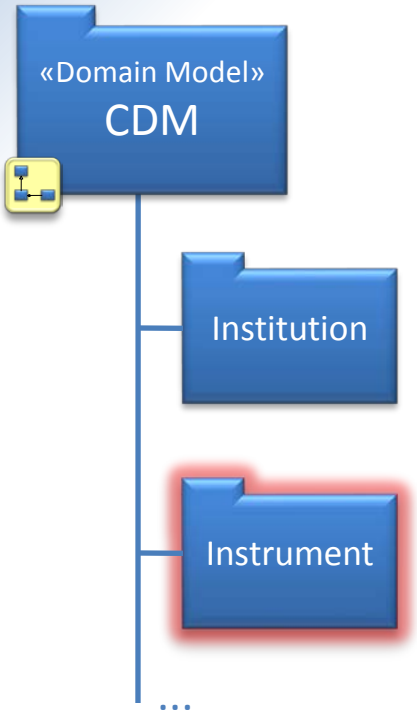
EXAMPLE MODELS

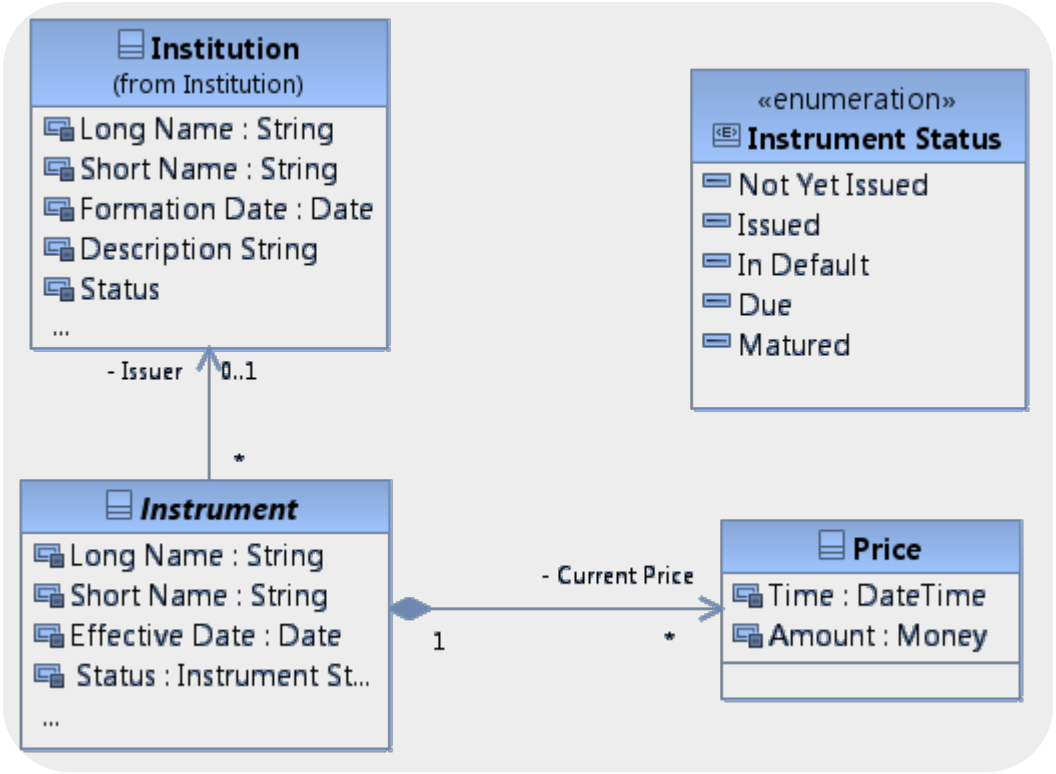
Domain Model



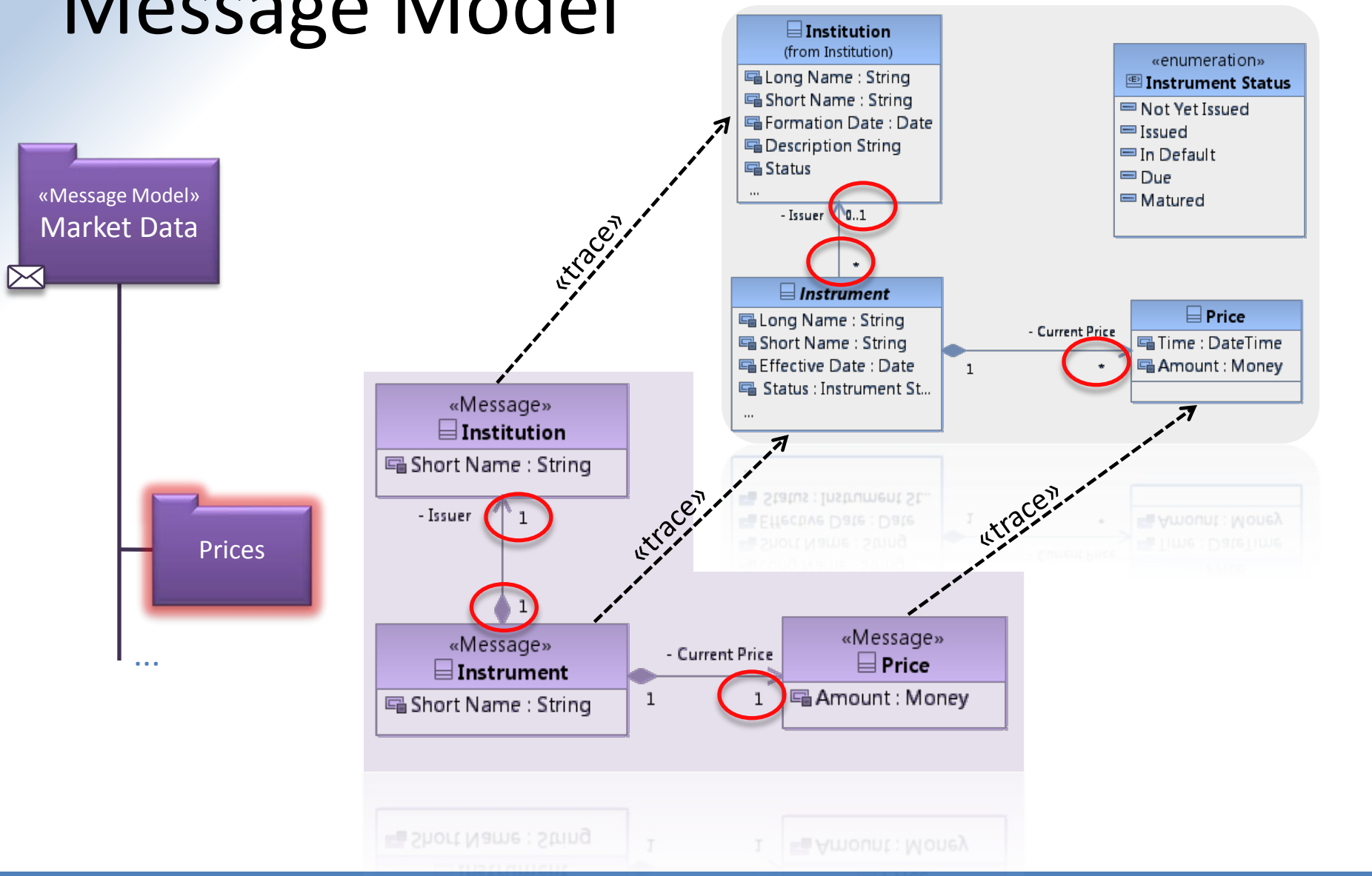


Domain Model

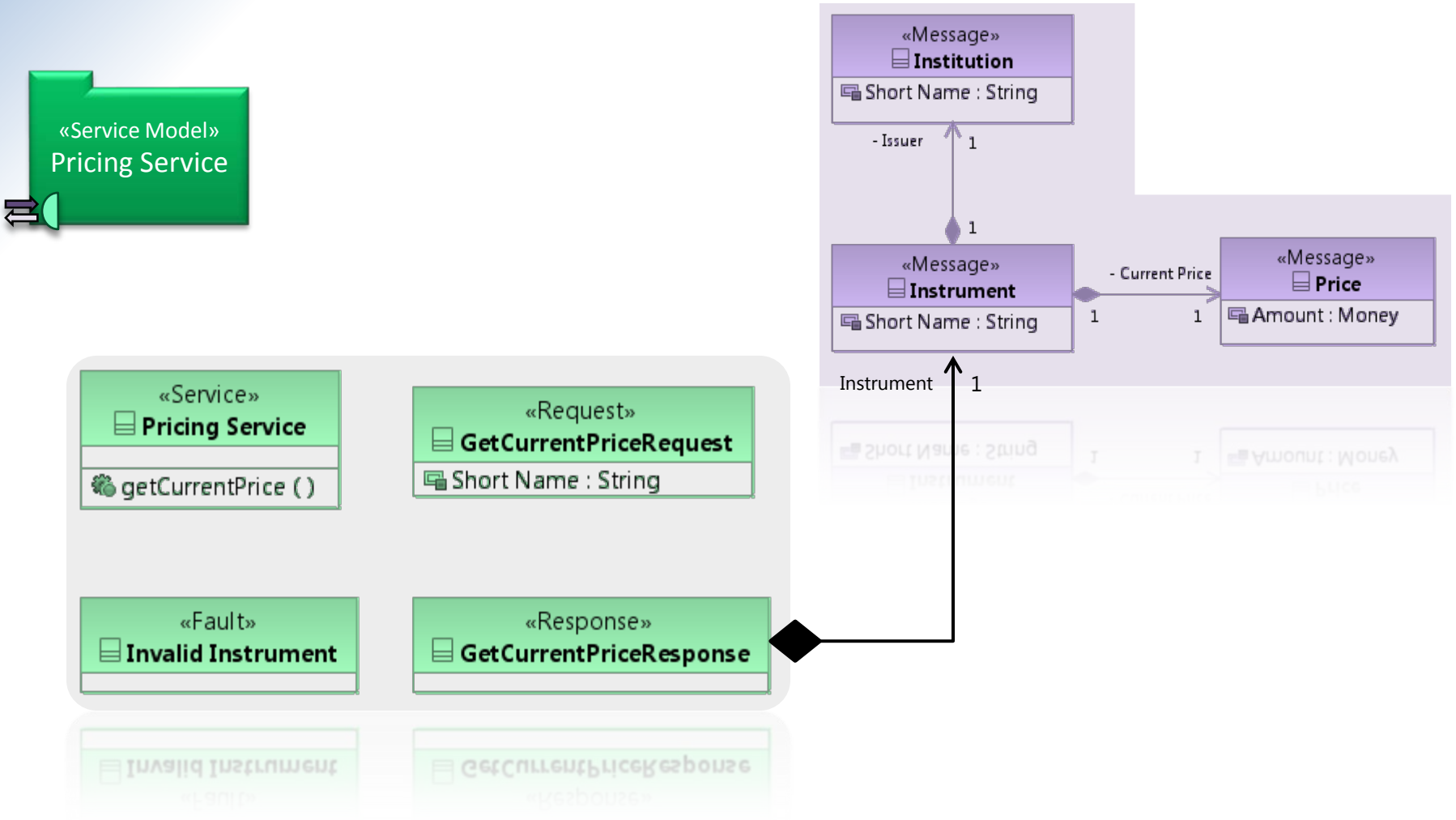




Message Model



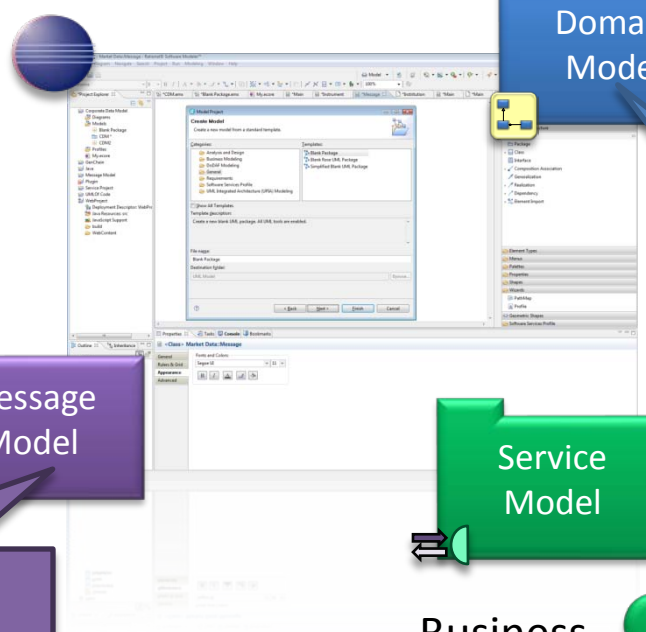
Service Model



What do we do with the models?

THE SOA TOOLKIT

The SOA Toolkit



Information Architect

Enterprise Modellers

Domain Model

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

Messaging Team

Message Model

- Create derived models
- Analyse and synchronise changes between domain and message model
- Generate XML Schemas

Service Model

Business Analyst

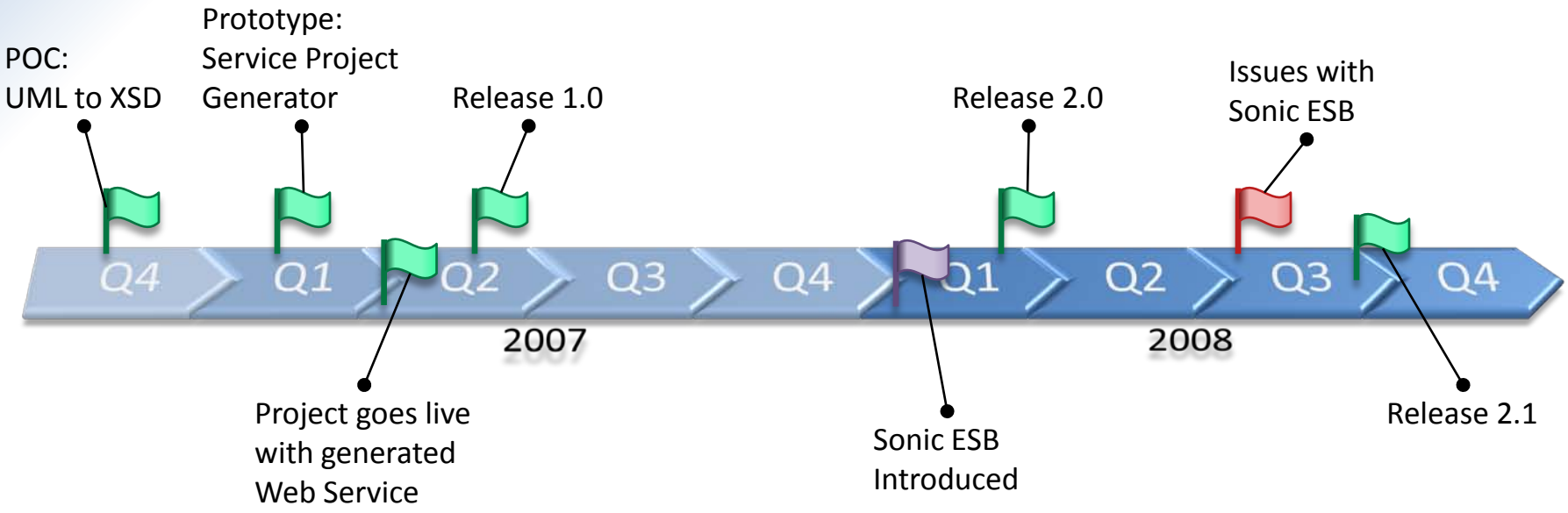
- Model Services
- Generate Web Services
- Apply standards and best-practice
- Abstract the platform
- Develop Java Code
- Deploy Services

Developers

Getting there

THE STORY

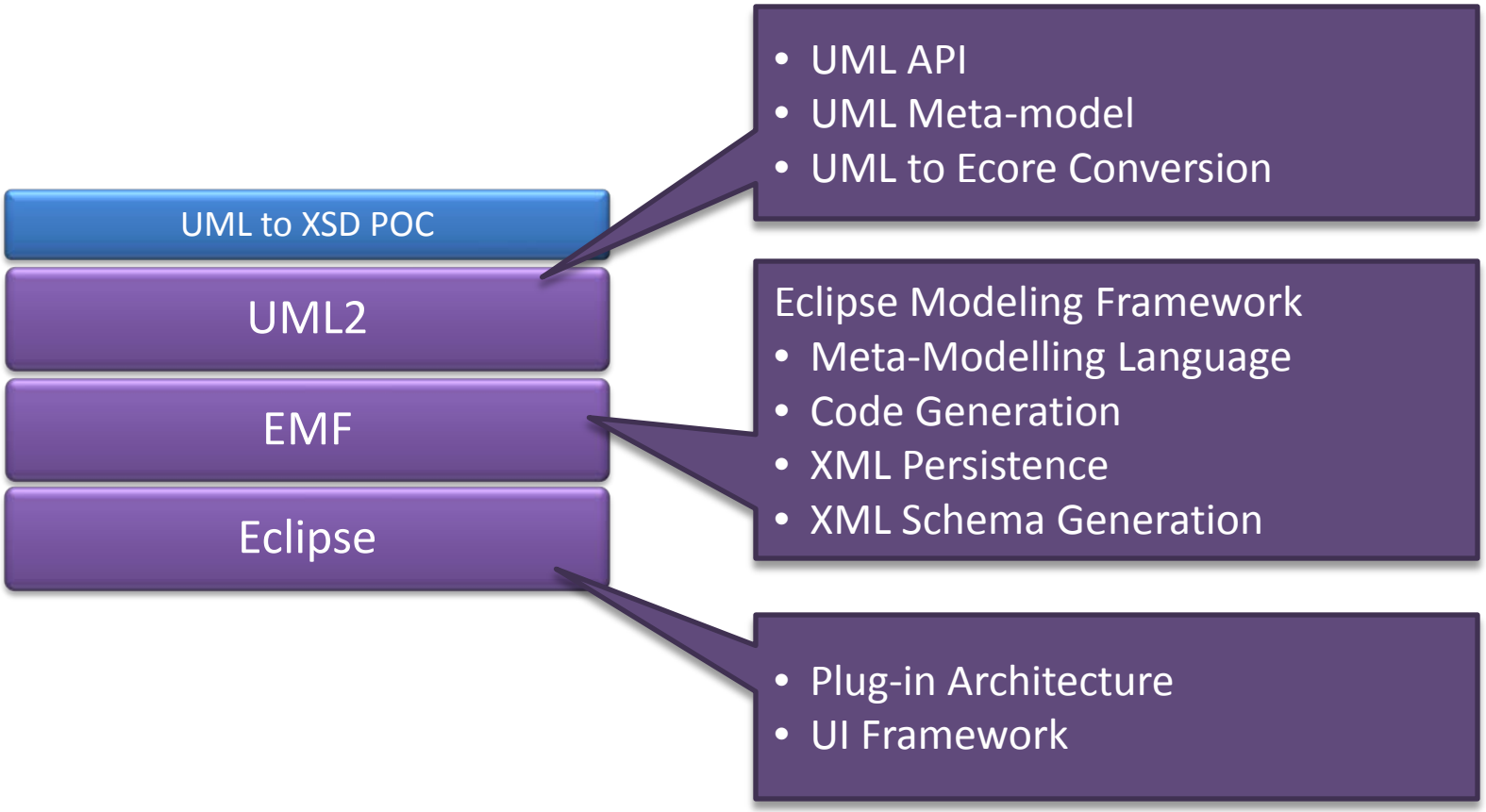
The Story - Overview



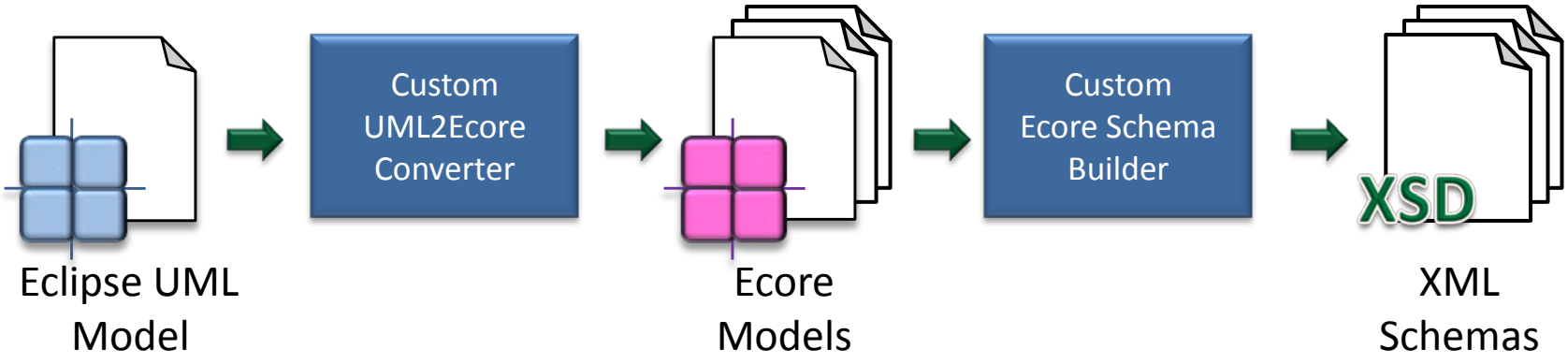
The proof-of-concept

From UML to XSD

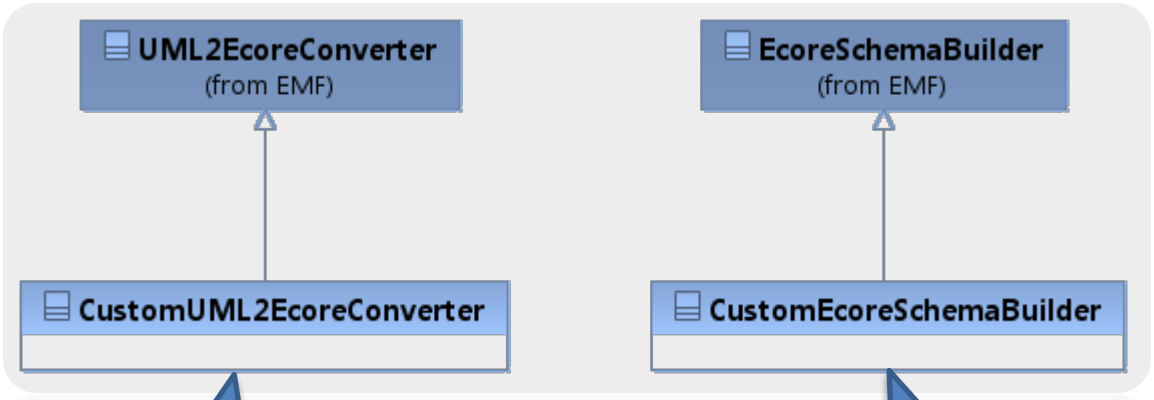
POC: The Stack



POC: UML to XML Schema



POC: UML to XML Schema

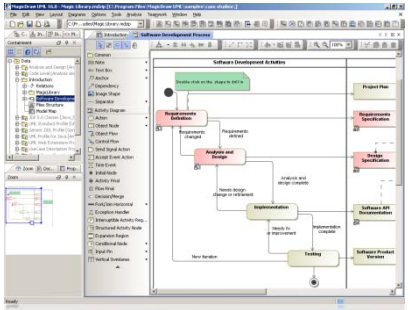


- Customise Data Mapping
- Experiment with XML Choice

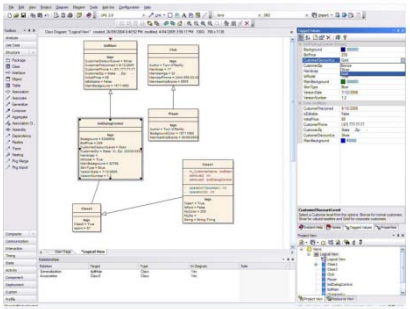
- Suppress Ecore annotations
- Standardise Namespace
- Reference XML Data Types

CHOOSING A UML MODELLING TOOL

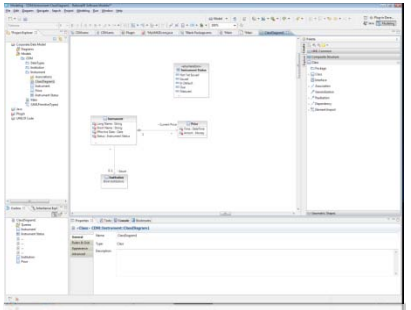
The Contenders



MagicDraw



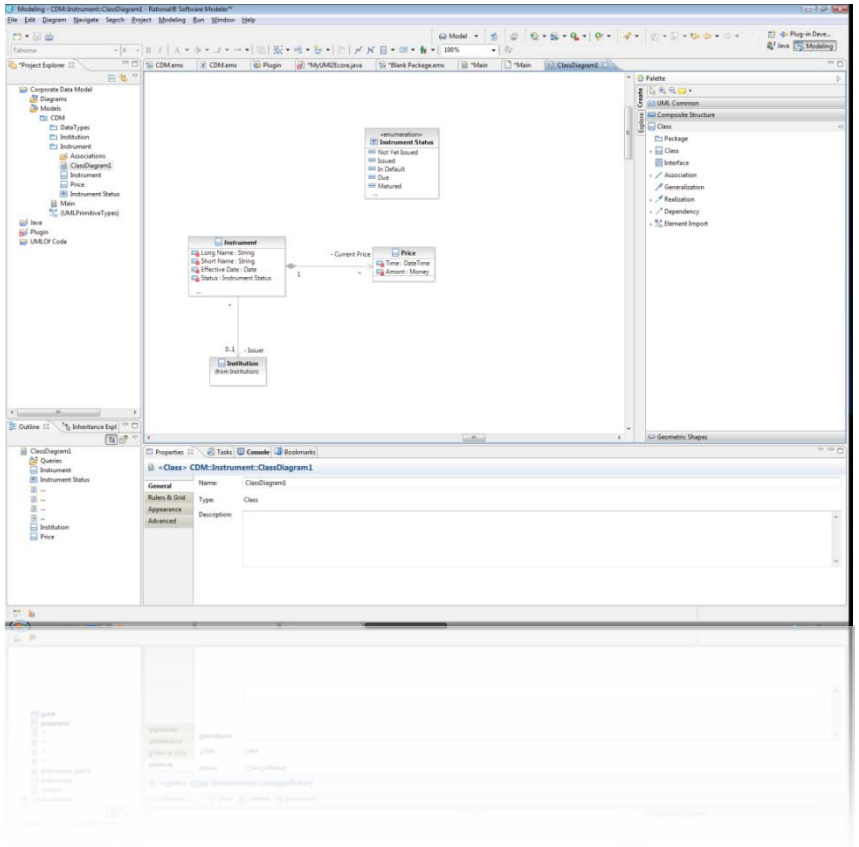
Enterprise Architect



Rational
Software Modeler

Why RSM?

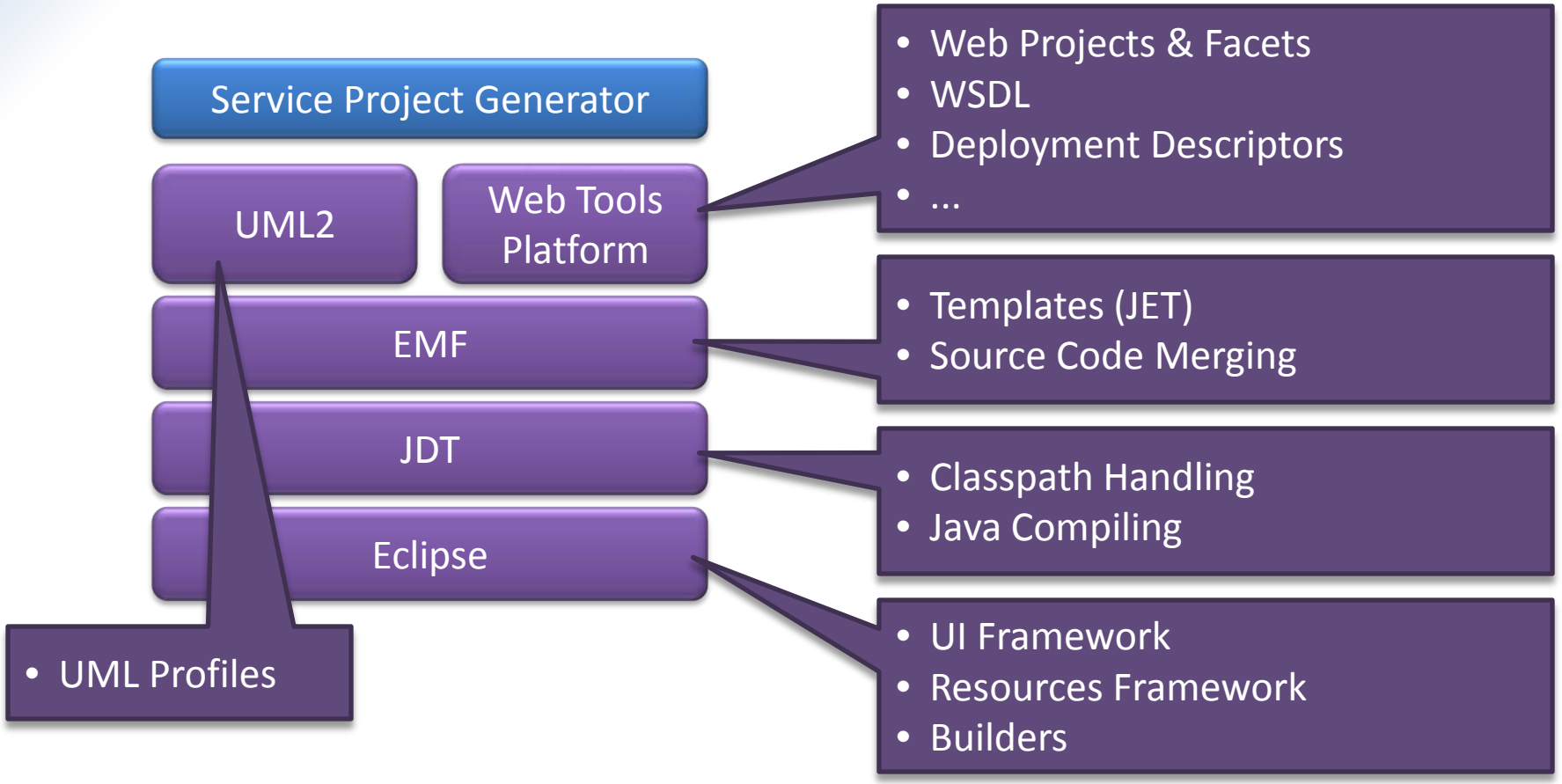
- Eclipse Based
  - Extensible
    - Plugin Development Environment
  - Integrated
    - Java Development
    - XML Editing
    - Version Control
  - Developer Friendly
- UML API
- Open-Source Foundations
  - EMF
  - UML2
  - GMF
  - TPTP



More Eclipse Frameworks

Web Service Generation

Web Service Generation Stack

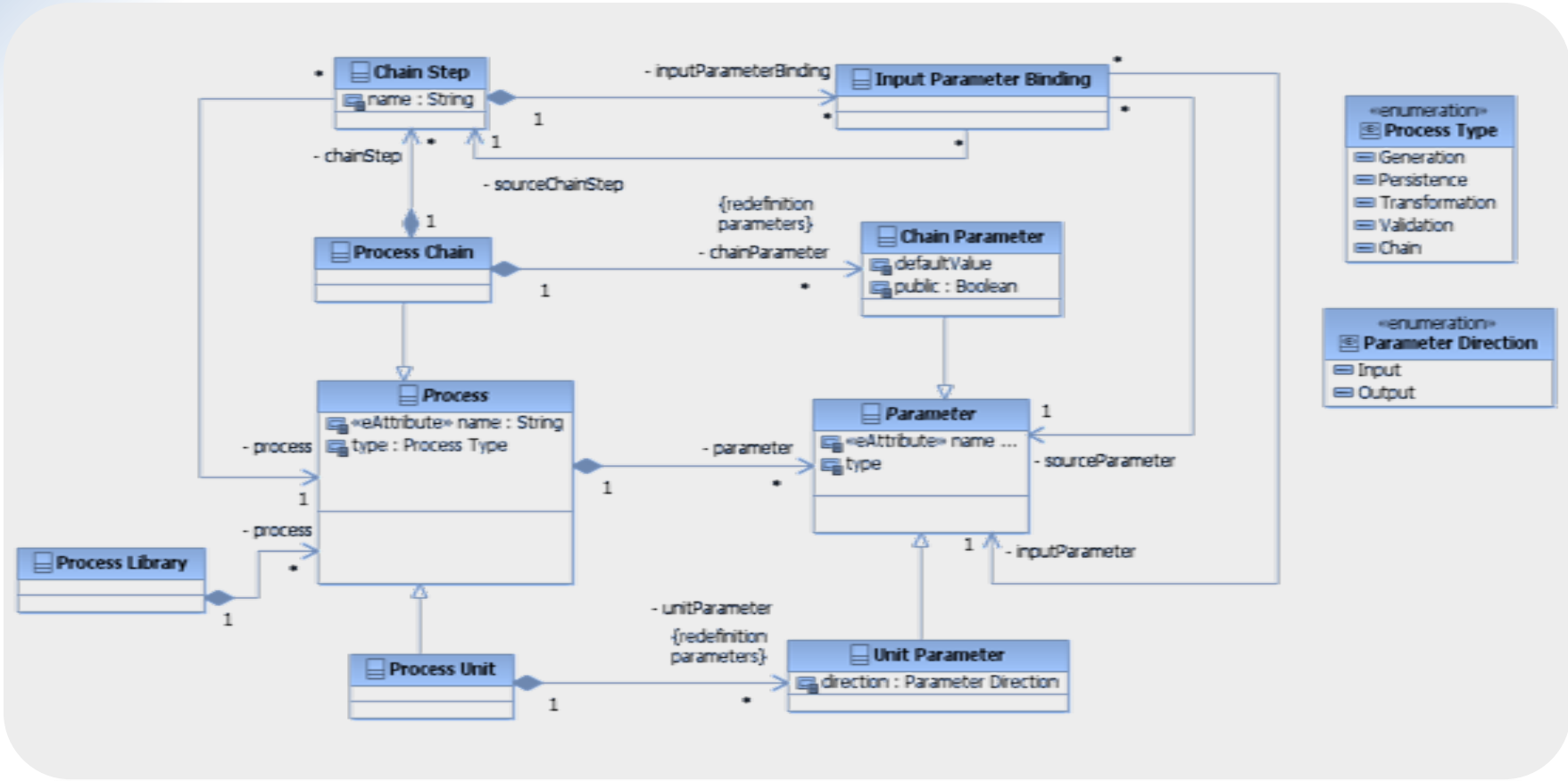


The Generation-Chain Model

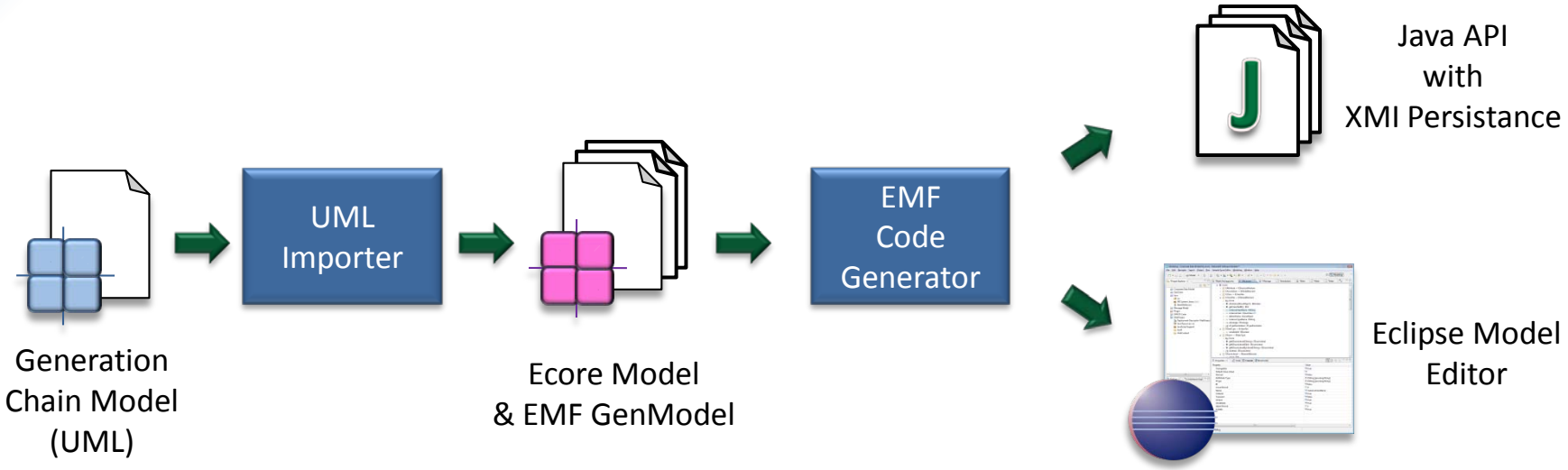
USING MDD TO DEVELOP MDD TOOLS

The Generation Chain

Generation Chain Meta-Model



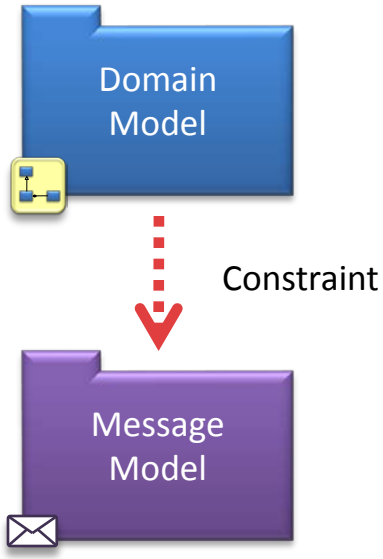
Generating an API



Creating and Synchronizing

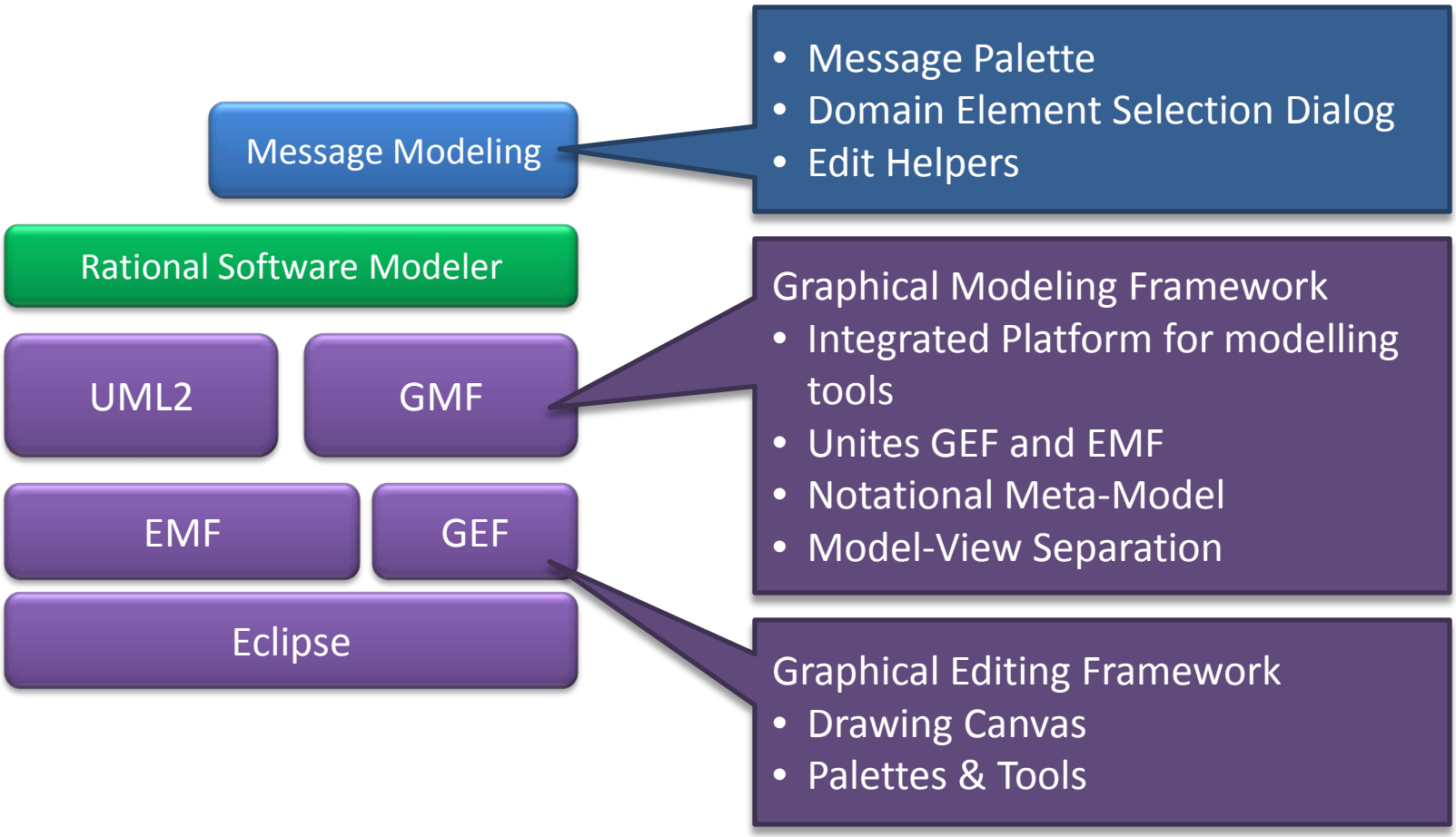
Message Models

Creating a Derived Model

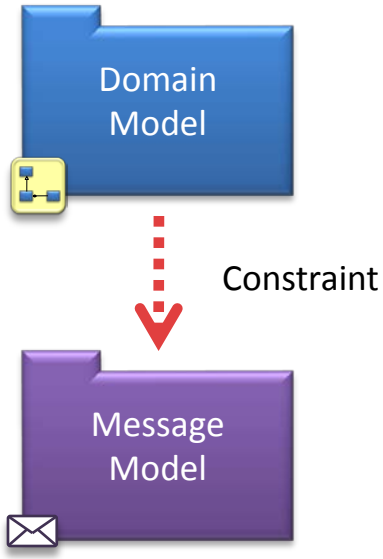


- Palette is constrained by domain
- Relationships limited to those available in Domain Model

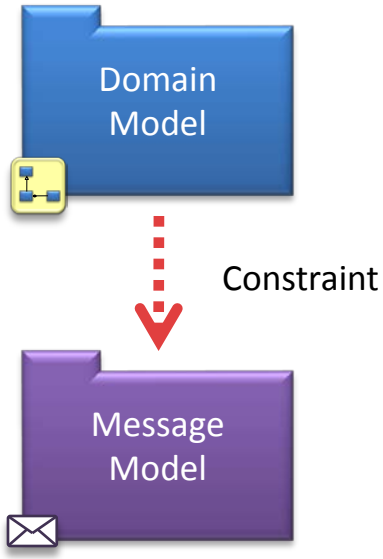
Creating a Derived Model



Synchronisation Problem

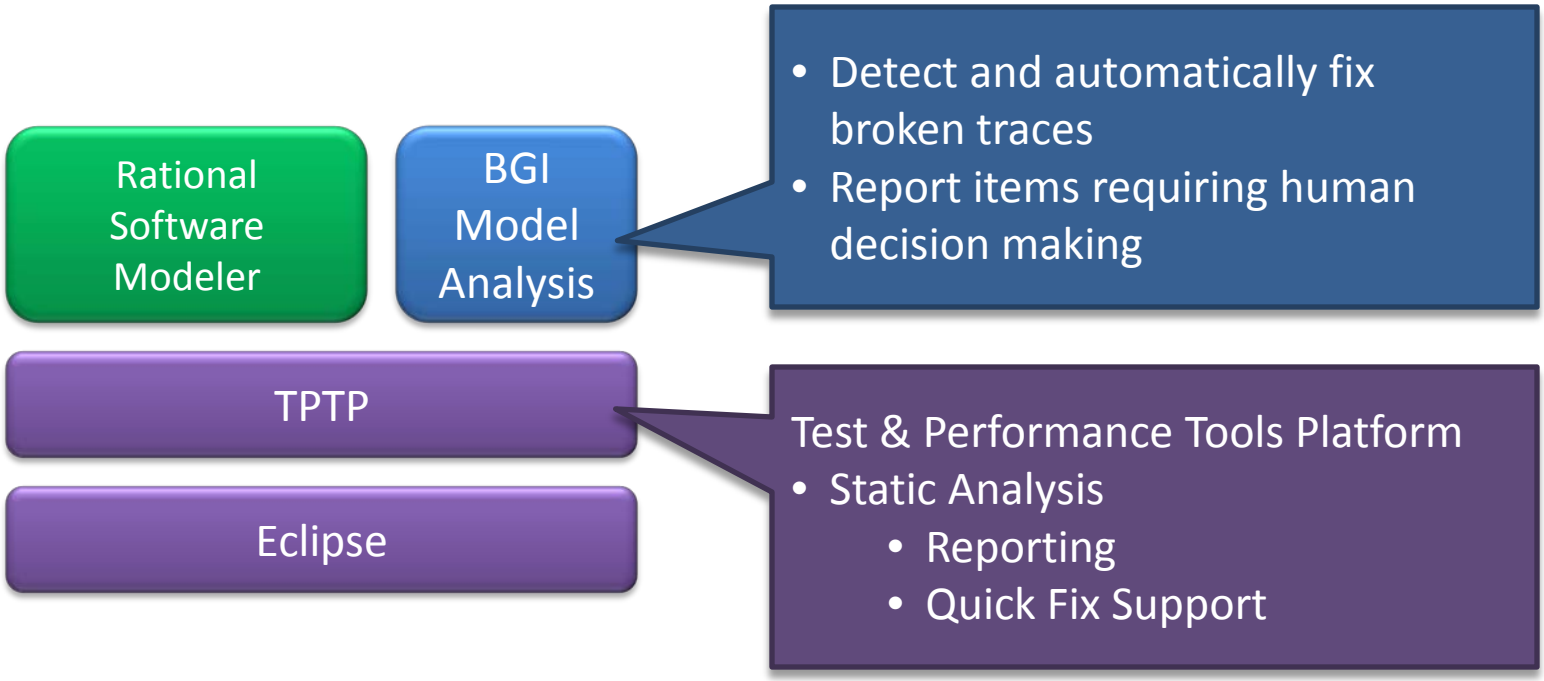


Synchronisation Problem



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

Synchronising Models



Keeping the models pure

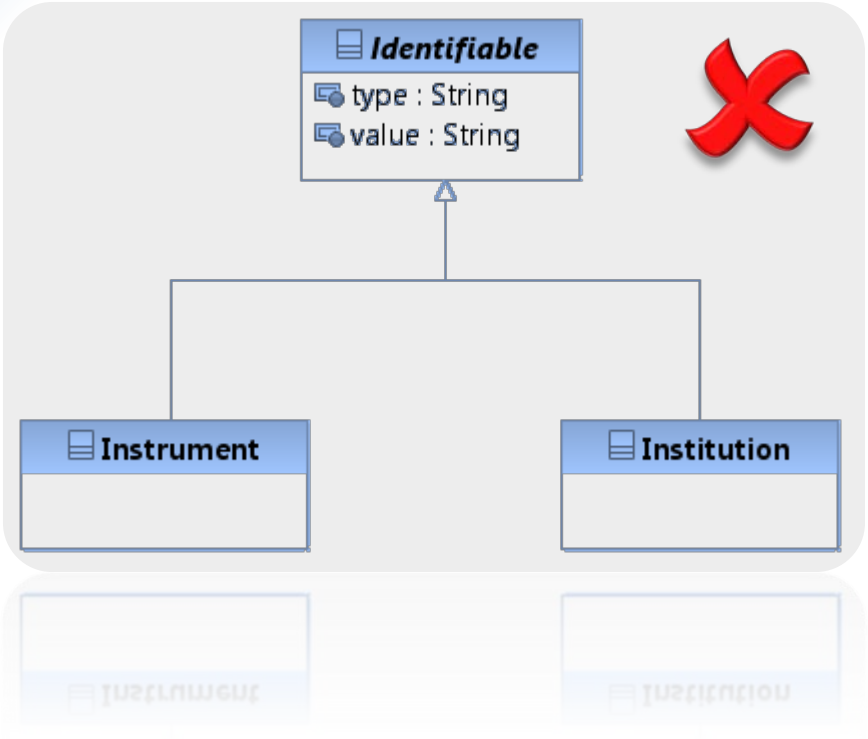
THE 'IDENTIFIABLE' PROBLEM

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>
```

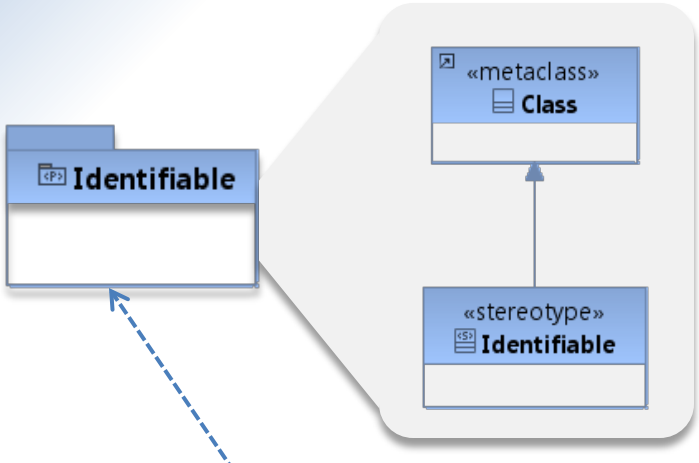
The ‘Identifiable’ Problem



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>
```

«apply»

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

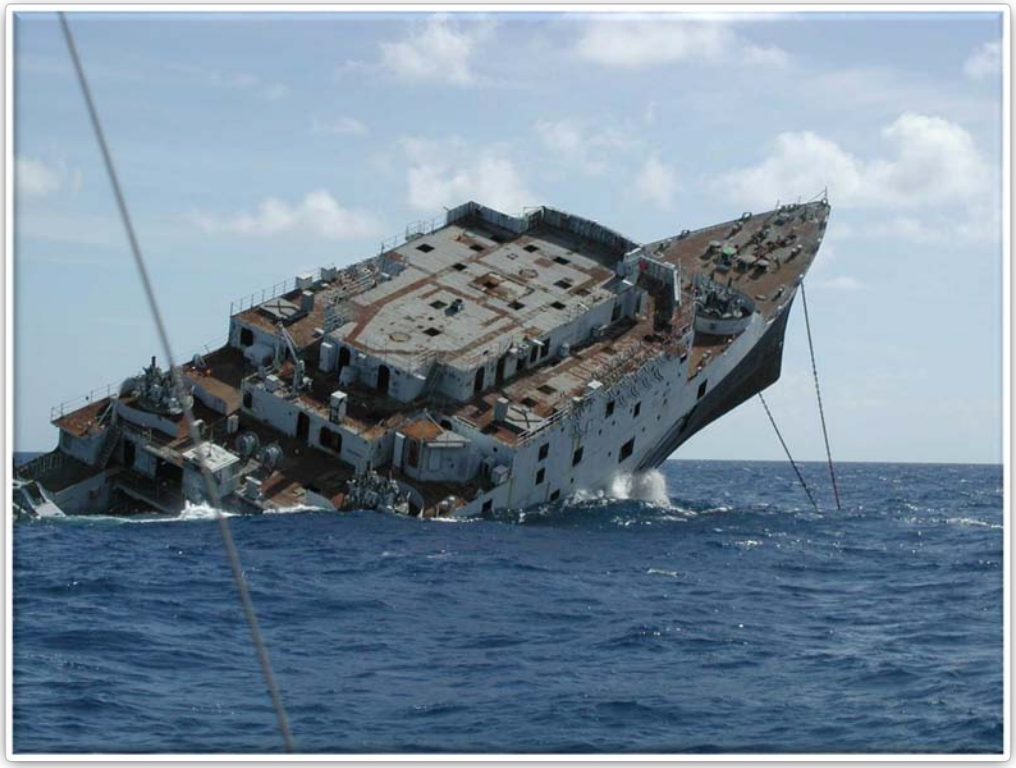
Have at least one MDD veteran

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