MDA

A Case Study

Espalier Technologies

17 September 2009

Joe Pullen, joe.pullen@espalier.com

Agenda

- Introduction
- MDA fundamentals
- AndroMDA
- Case Study
- Demonstration

Introduction

- Who am I
  - Middleware specialist (engineering)
  - Security (PKI)
  - Performance (Clustering)
  - WebLogic, JBoss
- Why MDA
  - Project exposure
  - Project development: use new tools but at lower cost

MDA Fundamentals

- MDA provides an approach for, and enables tools to be provided for
  - specifying a system independently of the platform
  - specifying platforms and choosing a particular platform for the system, and transforming the system specification into one for a particular platform.
- The three main goals of MDA through the use of architectural separation of concerns are:
  - portability
  - interoperability
  - reusability

MDA Terminology

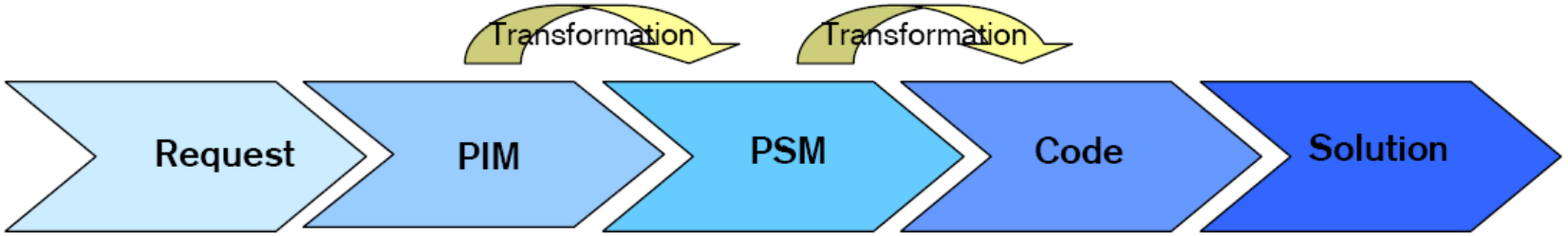
- Basic concepts of MDA are
  - its model-driven
  - it supports transformations
  - provides different views
- Models
  - Computation Independent Model (CIM)
  - Platform Independent Model (PIM)
  - Platform Specific Model (PSM)

MDA: Platform Independent Model

- With MDA the initial goal is the creation of a Platform Independent Model (PIM), this model is a one-or-one mapping of a Mental Model (MM) into a more formal language such as UML.
- Advantages
  - It is a very straightforward translation process
  - The developer/architect keeps focus on the business logic, not the underlying technology
  - The PIM can be reused later, it is not bound to any existing platform
  - The PIM is a useful medium to communicate ideas to others

MDA: Platform Specific Model

- The next step is to have a way to transform the PIM into program code, the MDA way of doing this is to gradually refine the model into a more Platform Specific Model (PSM) and transform this model into the code that normally would be written manually.



MDA Experience

- OptimalJ (2002)
  - Global Commodity Trading
  - Common Reference Project (Entity, JMS)
  - Global Accounting System (Entity, Web tier)
  - WebLogic 6 - 8.1, JBoss, Oracle 9i
  - Altura libraries, hard to integrate, expensive templates
- ArcStyler (2005)
  - International Bank
  - Service (Business) Facades
  - WebLogic 8.1, CORBA
- AndroMDA (2007)
  - International Bank
  - Reporting Service, Payments Monitoring Solution
  - WebLogic 9.2, Oracle 10g (RAC)

AndroMDA

- AndroMDA is an MDA framework for transforming models (usually UML models stored in XMI produced from case-tools) using a combination of plug-ins (cartridge and translation libraries) and to produce custom components.
- A very important thing to know about AndroMDA is that the transformation process is controlled using plug-ins called cartridges and each cartridge is very easy to tune.
- AndroMDA was developed to help eliminate boring and repetitive tasks, while at the same time allowing a model to really communicate what the system is doing: it is not meant be a Silver Bullet solution

See more at http://www.andromda.org/

AndroMDA: J2EE

- AndroMDA is mostly used by developers working with J2EE technologies
- Out-of-the-box AndroMDA can setup a new J2EE project from scratch, in which code is generated from a UML model
- Can be used to generate code for Hibernate, EJB, Spring, Web Services, Struts, JSF, Seam and more
- The generated code is automatically integrated into the build process using Maven
- Allows to keep focus on business logic
- Integrates out of container testing (Spring or micro-container)

See more at http://www.andromda.org/

Why MDA and AndroMDA

- It eliminates the need to write redundant code
- Project models reflect the actual code
- Projects are documented/diagrammed in a standard platform independent way making it much easier/faster to adapt to ever rapidly changing technologies
- Artifacts (models) that allow much easier communication with the business
- AndroMDA is not commercially marketed but the product speaks for itself:
  - community driven
  - visionary but pragmatic
  - open and modular
  - documented and release driven

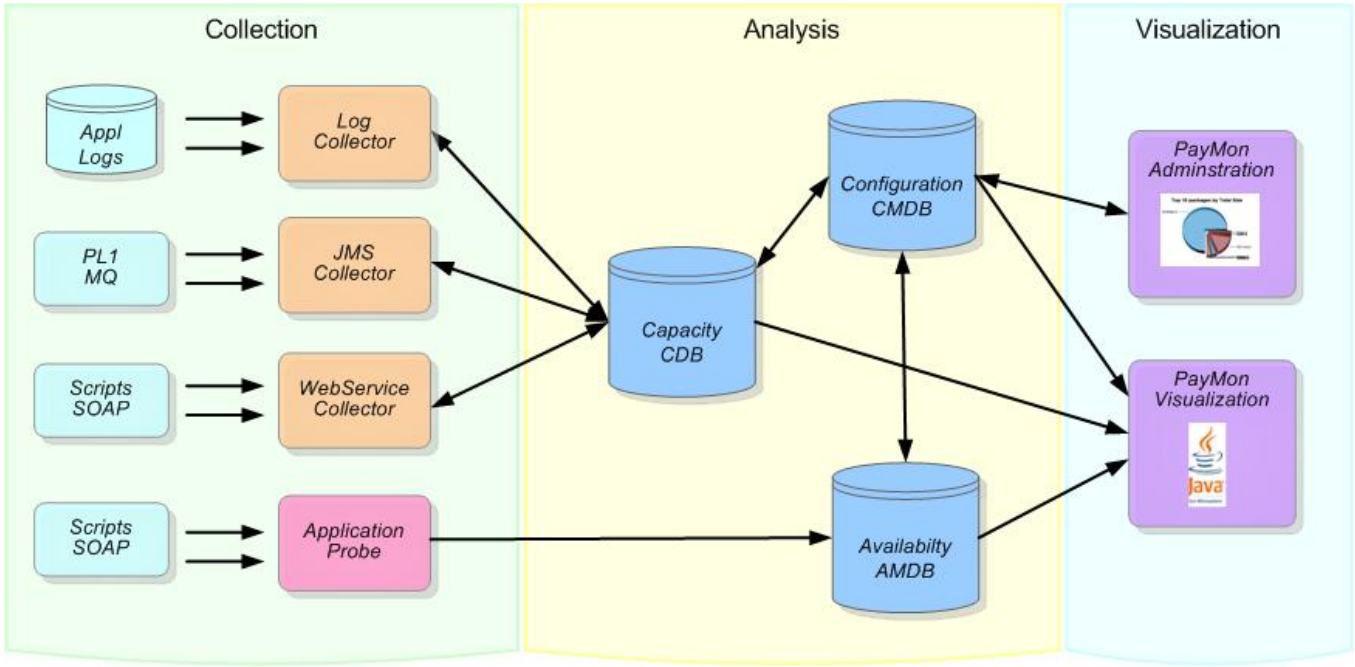
Challenges using AndroMDA

- Steep initial learning curve
  - learning how to model to generate the correct artifacts
  - learning the maven build lifecycle
  - good initial documentation for examples, lacking more advanced samples
  - complexity of combining open source tool set (Hibernate performance and mappings)
- Changing between different cartridges is not as transparent as expected
  - experience needed to discover best practices and pragmatic solutions (EJB3, Hibernate, HQL and OCL)
- XMI interoperability (MagicDraw, RSM) and dependency on custom profiles

Case Study: Project Details

- Banking environment
  - 1 analyst
  - 1 project management
  - 2 developers
  - 2 off shore developers
- Scope
  - Phase one: prototype, analytical log processing, metrics data model and BIRT reporting
  - Phase two: SLA management, alerting, complex pattern matching

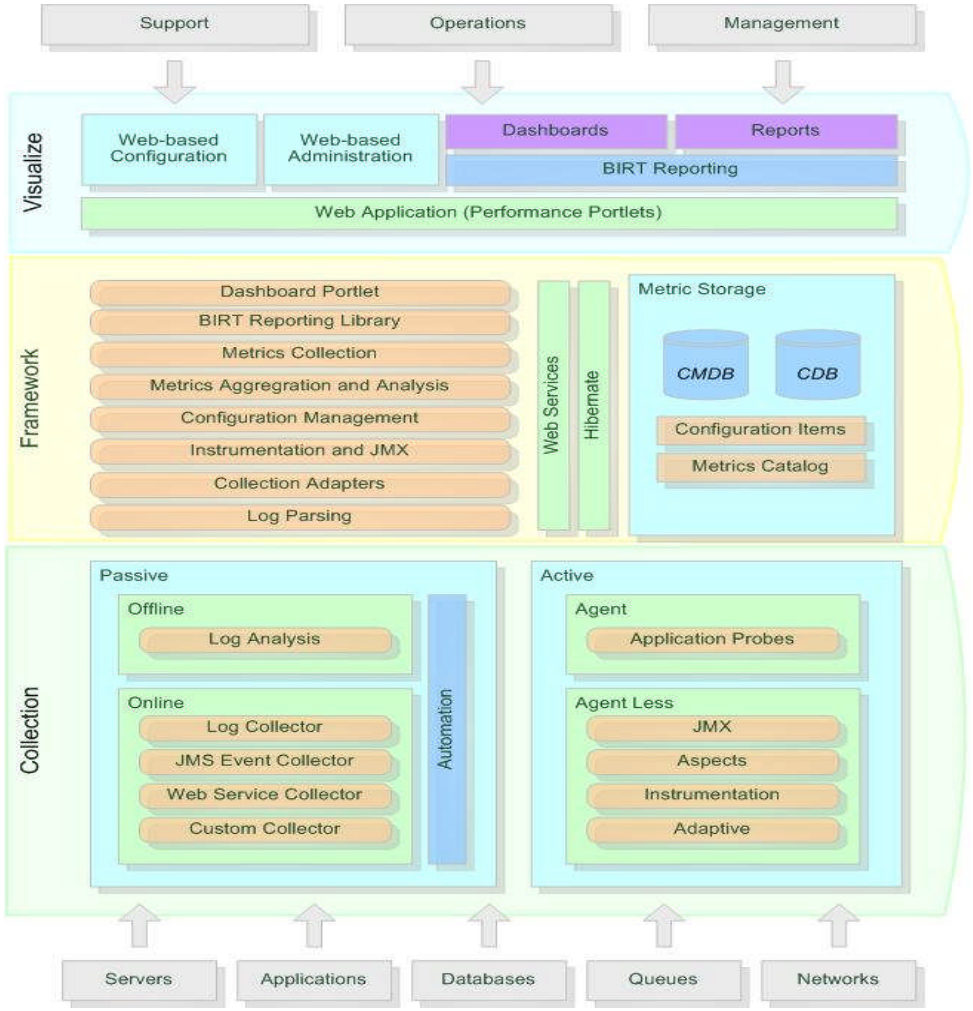
Case Study: Payments Monitoring



- Custom collectors
  - High throughput
- BIRT Reporting
- Cockpit
  - Dashboards (JSF)

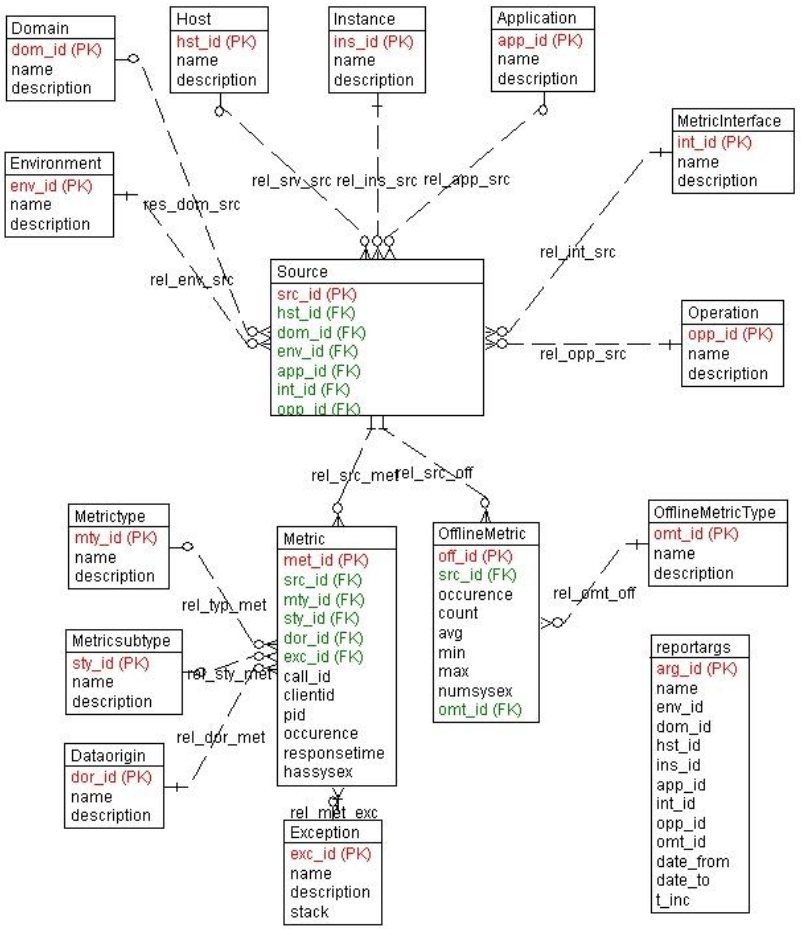
- Services
  - Reference Services
  - Reporting Services
  - Alerting Services

Case Study: Requirements



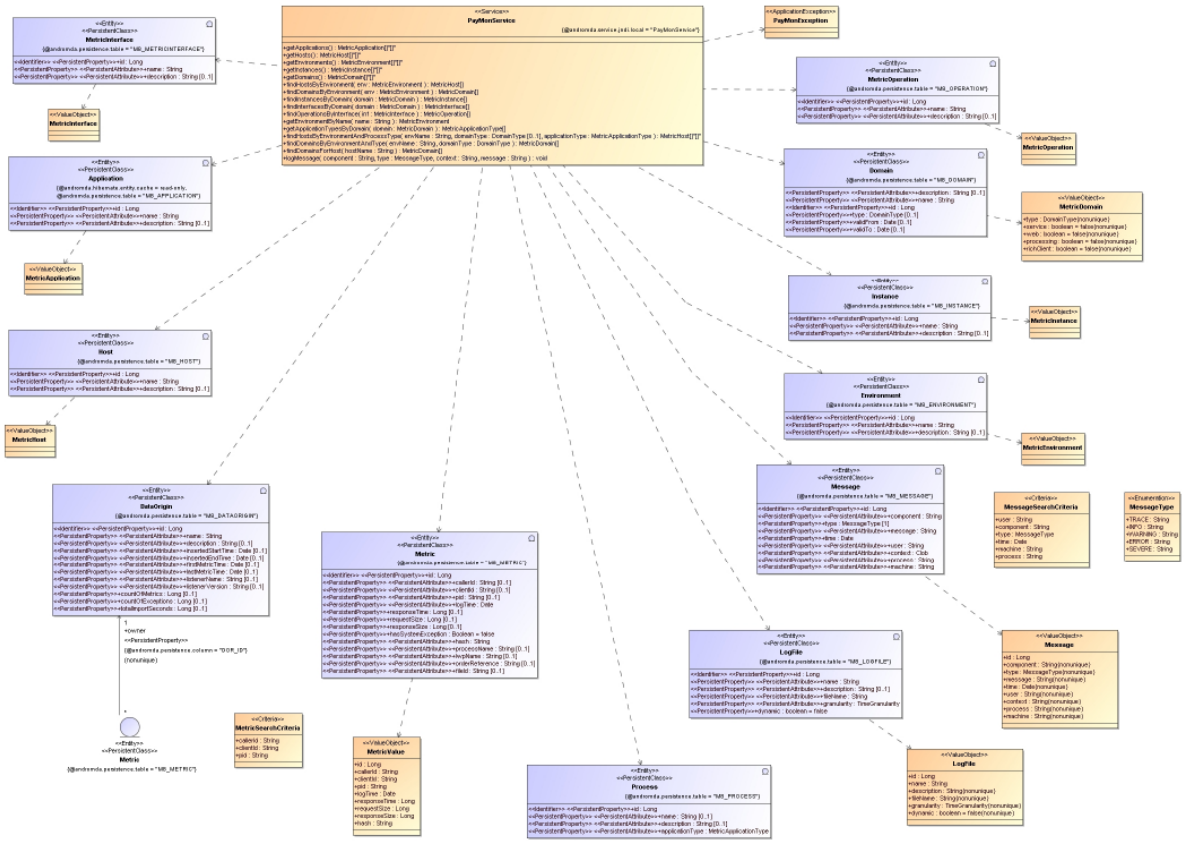
- Requirements
  - Open Source
  - Hibernate support
  - Web Services support
  - Java Server Faces Support
- Flexible build for integration into various target platforms
- Support for various modeling tools (UML)

Case Study: Payments Monitoring



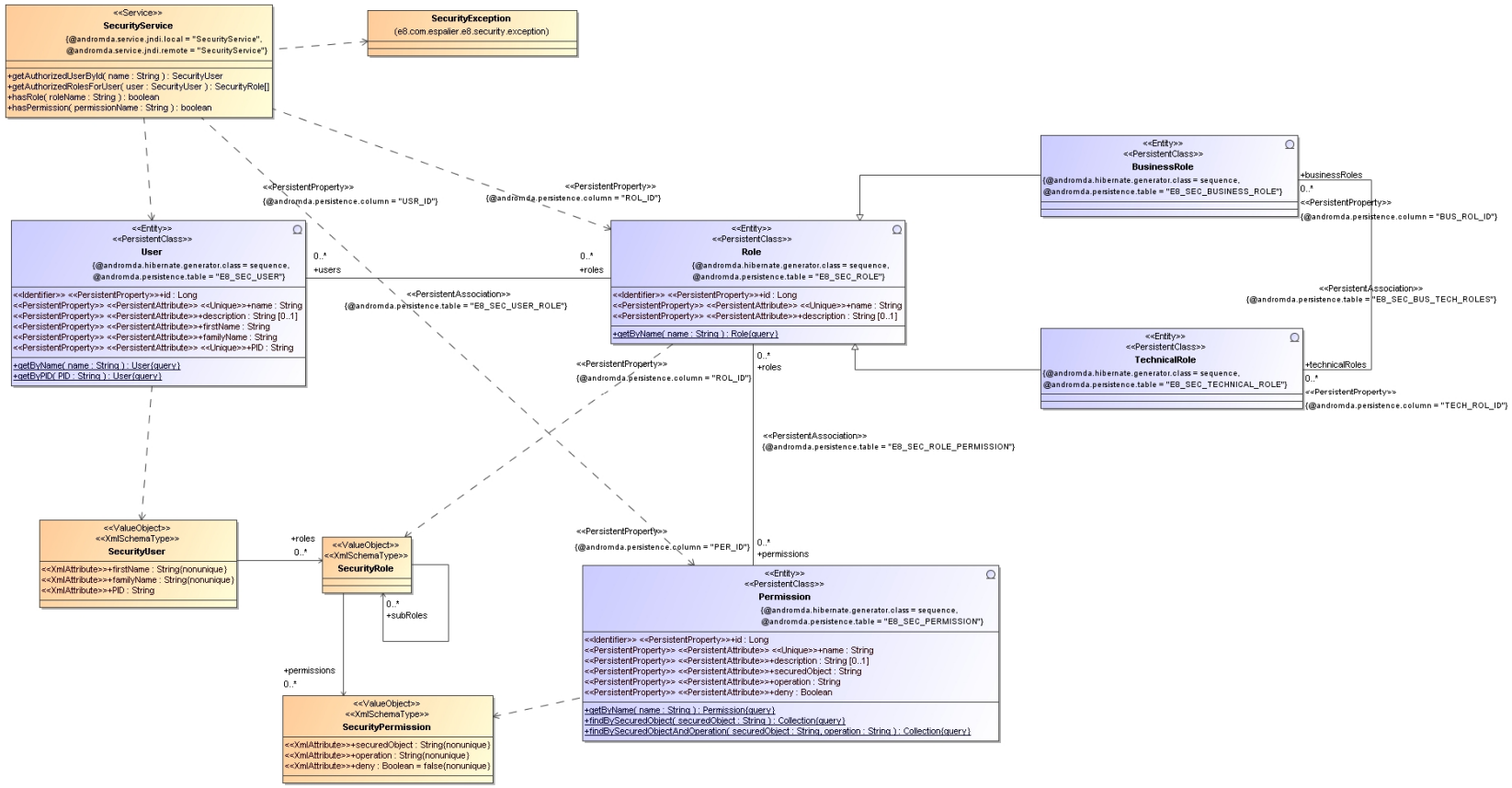
- Built on existing proven database model
- Managed evolution of model using MagicDraw
- Support distributed development (Off Shoring)

Case Study: Payments Monitoring



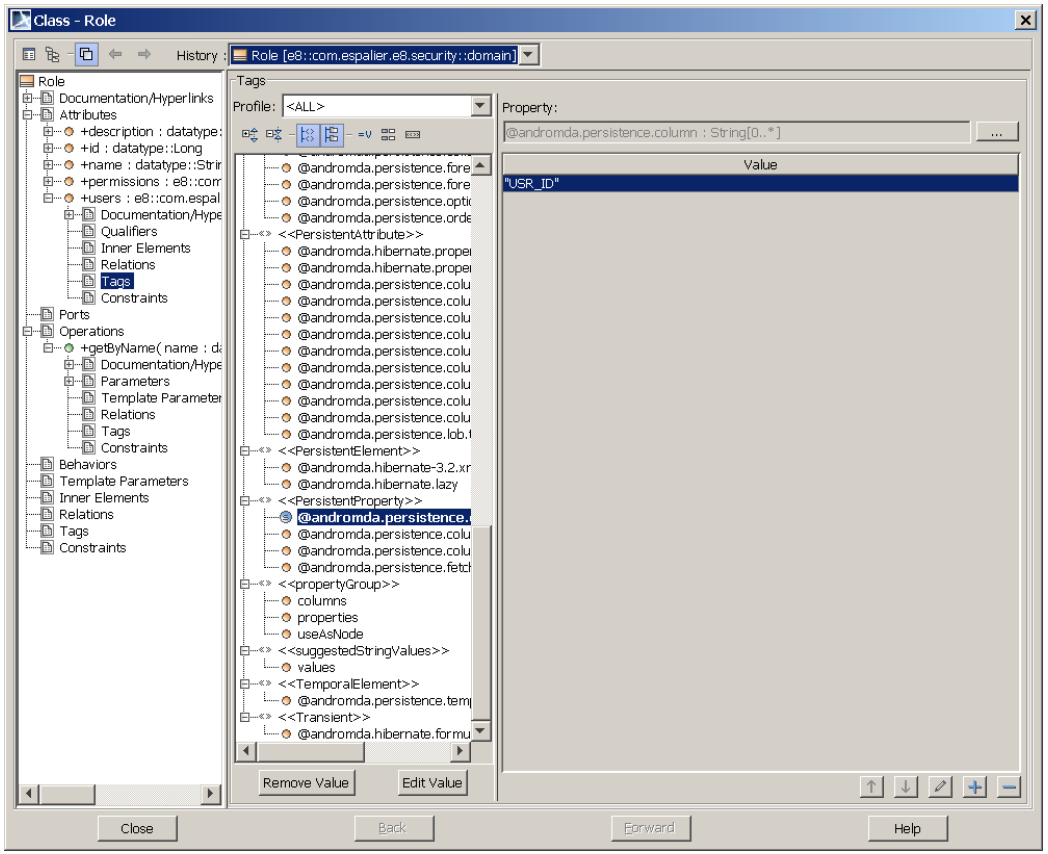
- 500-600 generated artifacts
  - services
  - value objects
  - entities (DAO)
- 100% generated ORM layer
- 90% service code generated
- code for business implementation in services
- JSF and web tier coded manually

Demonstration



- Generic Authorization Model

Demonstration



- Excellent support for ORM mapping using Hibernate
- Tags for managing artifact generation (services, value objects, entities)
- Optimized round trip engineering
- Integrated support for JUnit using Spring cartridge
- Improved quality through documentation and testing

Espalier Technologies provides services built on the foundation of the proven expertise of it's consultants in delivering effective solutions for its clients.

Espalier Technologies
Mühlegasse 11
8001 Zürich

Phone: +41 44 251 61 00
Fax: +41 44 252 54 47

Email: info@espalier.com

IT Solutions for a complex world™

www.espalier.com