Model Transformations

Richard Wood
richard.m.wood@credit-suisse.com
Model Transformations

- Applied in more than 7 different IT platforms and their projects at Credit Suisse,
- Over the timespan of three years,
- Code generation to increase productivity as main purpose
- Strong diversity in automation requirements
  - Input formats, generated artefacts, model abstraction level and enrichment expectations, etc
- OpenArchitectureWare as generator framework
  - versions 4.2, 4.3 and 4.3.1
- Eclipse 3.3 and 3.4, models based EMF/Ecore
- UML editors and text editors as modeling tools
Concepts and building blocks

- Models
- Generated Artefacts
  - Program Code, Test code, Deployment configuration, etc
- Metamodels (Architecture concept models)
- Readers, Writers (concrete syntax)
  - XML, UML/XMI, Java Beans, etc
- Transformations
  - Xpand (model to text, M2T), Xtend (model to model, M2M)
- Validation
  - Check (oAW), OCL and Java: Eclipse Validation Framework
- Transformation Workflow
  - Model slots as interface between workflow components
Global Services as Enterprise Java Beans

- Java classes for EJB implementation, client and types need to be generated
- Service interface definitions specified as UML class models
- Common data types are specified in separate UML class model
  - Service interface definitions reference common data types
  - Data types are managed independently
- Initial estimates: 50 services and >100 types, >10 Java classes per service
- Changes in common types to be expected, iterations
- Quick and dirty solution, two weeks implementation time
- Disadvantages:
  - code templates difficult to maintain (UML metamodel knowledge required)
  - memory demand too high for batch generation
Global Services as Enterprise Java Beans

- UML Class models
- Common Data Types
- Service Interface
- XMI Reader
- Java files
- Data Type Classes
- EJB Classes
- UML to Java
- M2T
- UML complexity
- tight coupling
- two separate transformations

Produced by: Wood Richard
Date: 16.11.2009 Slide 5
Global Services implemented as Webservices

- WSDL and XSD files need to be generated
- Same input as for EJBs, UML models with references to common data types
- Replaces EJB generator solution
- Independent lifecycles for services and data types, independent versioning
- One WSDL per service plus XSDs for common data types
- Dedicated metamodel introduced for global service concepts

Advantages:
- UML transformation decoupled from code generation
- Batch generation better supported
Global Services implemented as Webservices

- UML Class models
- Common Data Types
- Service Interfaces

- Ecore model
- Service Meta Model (SMM)
  specified in
  - UML to SMM
  - SMM to WSDL and XSD

- XSD for Data Types
- WSDL for Services

- XML files

- No M2T required

Multi Model Reader
Model Validation
M2M
XML Writer
Interface Management System

- System for managing service descriptions and generating platform specific artefacts
  - IDL, PL1 Stubs, WSDL, etc
- Consists of a central database, a web application for queries and modifications, and a set of generators
- Service descriptions are specified and reviewed in a platform independent way
  - service metamodel
- Platform specific models are assembled as intermediary step
Interface Management System

Service Descriptions
Oracle Database
Java Persistence API

Ecore model
Service Meta Model (SMM)

platform specific metamodels
specified in
Java Bean Reader

Service Meta Model (SMM)

platform specific service definitions
and artefacts

SMM to IDL

M2M
Model Validation

M2T
+ IDL Beautifier

SMM to PL1

M2M
Model Validation

M2T
+ PL1 Beautifier

Corba/IDL

PL1

Java to SMM

Java to SMM

Java Bean Reader

M2M
Data Access Layer

- Java classes for EJB 3.0 implementation with JPA usage need to be generated.
- Tasks: operation auditing, data access authorization, queries, CRUD operations, input validation, technical fields, exception handling
- Initial DSL based on XML schema definition for quick prototyping and existing experience with XML
- XSDs impose design restrictions to metamodels
  - tree structure, element references, inheritance, etc
Data Access Layer

- XSD model
  - Data Service Modeling Style
  - model refinement
  - XML model

- Ecore model
  - Data Service Meta Model
  - specified in
  - Ecore model

- Data Service Meta Model
  - M2M
  - M2T
  - test code
    - Java files
  - XSD model
  - service description
    - PDF
  - XML configuration files

- XML Reader
- Model Validation
- M2M
- M2T

Produced by: Wood Richard
Date: 16.11.2009 Slide 11
Project XYZ (multi tier application)

- Large project with the goal to maximize automation during software development. Commitment to model the different layers of the planned solution (presentation, business and data).
- UML as modelling standard within organisation.
  - Technology choices for tooling.
- Models are expressed in a domain specific language (UML + UML Profile)
- Software designers are supported with tailored modelling tools
  - Tooling palette, model validation, collaboration schemes
- UML class models for component definition, UML activity models for business logic
Project XYZ (multi tier application)

UML Class models
UML Activity models

XYZ Modeling Style

Ecore model
XYZ Meta Model

specified in

UML Profile

XMI
Reader
Model Validation
M2M
M2T

web gui
Java files

data access
Java files

business service
Java files

test code
Java files

Spring and other XML config files

XYZ Modeling Style equivalent

Java files

Java files

Java files

Java files
UML Profile generation

- Disadvantage of creating manually a dedicated UML profile in addition to the meta model
- Better to generate the UML profile from the metamodel
  - technical conversion, no semantic changes
- A DSL mapped to UML can use UML elements, i.e. not all concepts need to be expressed as part of a new UML profile.
  - example: NamedElement for elements containing an attribute "name"
- UML Profile elements
  - Stereotypes, properties, inheritance, associations, enumerations, primitive types, constraints
UML Profile generation

- XMI Writer
- M2M
- XMI Reader

- UML
- UML Profile
- XYZ Modeling Style
- Ecore model
- XYZ Meta Model

Mapping specified in

UML model
Lessons learned

- Metamodel requirements for modeling and generating are different
  - dedicated metamodels with an explicit M2M transformation, more flexibility for the modeling style, less complexity and restrictions for the transformations

- No code generation based directly on UML
  - Code templates (M2T) are best created and maintained based on a specific metamodel tailored to the solution domain

- Static content in code templates needs to be independently maintainable
  - Most software engineers prefer to work with M2T than M2M transformations
    => keep M2T transformations simple

- Check for existing workflow components
  - example XMLWriter
Model Transformations

- Contact
  - richard.m.wood@credit-suisse.com

- Questions