Consolidation of the Generator Infrastructure
MDGEN – Model Driven Generation

Date: October 16th, 2012
Produced by: Mario Lovisi / Serano Colameo
Version: 1.0
Agenda

- Initial Situation / Goal of the Project (Mario Lovisi)
- The Multi Channel Platform (MCP) (Mario Lovisi)
- IcmDsl & MDGEN Features (Serano Colameo)
- Q&A (Mario Lovisi / Serano Colameo)
## Initial Situation

<table>
<thead>
<tr>
<th>Generator</th>
<th>Artifacts</th>
<th>Model</th>
<th>Technology</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOOLS GenCAL</td>
<td>EJB 2.x conform Java Code for Standard, FN and MCP Services</td>
<td>Java Code</td>
<td>Velocity and Recoder Framework</td>
</tr>
<tr>
<td>JBSGEN</td>
<td>EJB 2.x conform Java Code for FN Classic Services</td>
<td>XML / DTD</td>
<td>oAW 3.x (Xpand only)</td>
</tr>
<tr>
<td>Data Service Framework</td>
<td>EJB 3.x conform Java Code and Persistence Layer stuff for FN</td>
<td>XML / XSD</td>
<td>oAW 4.x (Xpand and Xtend 1.x)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
## Initial Situation

<table>
<thead>
<tr>
<th>Generator</th>
<th>Artifacts</th>
<th>Model</th>
<th>Technology</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOOLS GenCAL</td>
<td>EJB 2.x conform Java Code for Standard, FN and MCP Services</td>
<td>Java Code</td>
<td>Velocity and Recoder Framework</td>
</tr>
<tr>
<td>JBSGEN</td>
<td>EJB 2.x conform Java Code for FN Classic Services</td>
<td>XML / DTD</td>
<td>oAW 3.x (Xpand only)</td>
</tr>
<tr>
<td>Data Service Framework</td>
<td>EJB 3.x conform Java Code and Persistence Layer stuff for FN</td>
<td>XML / XSD</td>
<td>oAW 4.x (Xpand and Xtend 1.x)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Goal of the Project

- Unify existing Generator Technologies in one Technology (Xtext)
- To be able to include existing code (Java) and models (e.g. IDL)
- Import existing interface definitions (IDL) into the new DSL (ICM)
- Generate for MCP more artifacts (Façade Implementation, Data Mappers, Transfer Object Handler, Commands etc.) as before
- Allow a “model-driven” migration (e.g. EJB2 => EJB3, JAP etc.)
What is the Multi Channel Platform?

MCP is a pragmatic Cooperation Model for Business to increase effectiveness and synergies by:

- aligning business needs across channels
- aligning new IT initiatives concerning channel application within business
- reusing existing business functionality
- ensure consistency of functionality and data across channels
- an Integration Platform for IT to increase efficiency by sharing functionality, expertise and resources.
MCP Façade – Technology Architecture Concept

A façade is a design pattern, that provides a consistent interface for client applications and contains comprehensive logic to be reused:
Agenda

- Initial Situation / Goal of the Project  (Mario Lovisi)
- The Multi Channel Platform (MCP)  (Mario Lovisi)
- IcmDsl & MDGEN Features  (Serano Colameo)
- Q&A  (Mario Lovisi / Serano Colameo)
1st Step: Analysis, Separation of Concerns

Middlewares

|         | XICB | CAL | DSF | ...
|---------|------|-----|-----|-----

Targets

- FN Service Host
- FN Service DB
- FN Security
- GCRM
- MCP Facade
- Standard Svc Host
- Test Cases

Cross Cutting Concern

EJB 2.1 = ▲
EJB 3 = □, □ (Future)

Technische Subdomain 1

Metamodell 1

Technische Subdomain 2

Metamodell 2

Technische Subdomain 3

Metamodell 3
2nd Step: Define Runtime Usage of the MDGEN Tools

MDGEN Generator (ant task)

Features:
+ EJB3
+ JUnit4
- XML
...

JDK and JAP is automatically recognized by the Generator!

Java Artefacts including tests

Config files

EJB xml files

features:
+EJB3
+JUnit4
-XML
...

Eclipse: JDT, EMF

Developer

build.xml (ant, qmb)

build properties

execute

generator properties
3rd Step: Design DSL and Generator Architecture

- Interface Component Model as a Domain Specific Language – IcmDsl
- Built-in and extendable Type System with Mapping Functionalities
- Java is directly supported as Model (Jvm-Model) in IcmDsl
- IDL can be Imported (transformed) into IcmDsl (model-2-model & model-2-text)
- Other Artifacts can be imported as well (we stopped with WSDL)

Refer existing Java code

ICM

Source Models (existing artifacts)

Service/Component

Mapping Modell

Persistence Modell

Realized = red
Disabled = green

Refer existing Java code

ICM

Source Models (existing artifacts)

Service/Component

Mapping Modell

Persistence Modell

Realized = red
Disabled = green
inheritance and versioning concept

types and constant bind java types

definition of exceptions

mapping of existing data structures

Use of Annotation

definition of components
Generic Java Generator Architecture

IcmDsl Files

Platform Independent to
Platform Specific Model

m2t – transformation

m2m – transformation

m2t – transformation

Model 2 Text Templates

xtend

Java Meta Model – JMM

association

JVM Model

Java

Model 2 Model

xtend

m2t – transformation

m2m – transformation
Generated Artifacts

- Service Layer: Interface, Implementation, Delegator, Helper Classes
- Transport Layer: Transfer Objects, Transfer Object Handler, Command Classes
- Business Layer: Business Objects, Data Mapper Classes
- Data Layer: XJCB Services (Java Corba Bridge Service Artifacts)
- JUnit Tests for each Service Operation
- Deployment Descriptors
- Configuration Files (XML)
- ...

All needed Service Artifacts are automatically generated
Mapping of Business and Transfer Objects can be also generated
Only complex Business Logic must be manually implemented

Use of Generation Gap Pattern to allow manual enhancement
Migration of Existing MCP Façade Services
Change Technology Stack: EJB2.1 => EJB3, JAP6 => JAP7

Features:
+EJB3
+SkeletonOnly

Generator Switches

JDK and JAP is automatically recognized by the Generator!
Migration of Legacy Services (Java Corba Bridge – JCB)

Source

We migrated over 300 existing JCB services by just importing and regenerating all existing artifacts in one step!
Mapping of Tier Layer Data Structures

- Enterprise Architectures consists of many Layers...

Mapping of Data Structures
Mapping and Binding of existing Models and Artifacts

Mapping of IcmDsl Types (class, structure etc.)

Mapping of Java types (classes)

Binding of IDL Types/Interfaces
Mapping (i) of Model and/or Java Types in Combination

IcmDsl:

```java
structure MappedStructure {
    mapping of type A {
        map a1;
        // ...
    }
    mapping of java A {
        map a2;
        // ...
    }
    mapping from type A to type B {
        map a1 to b1;
        // ...
    }
    mapping from type B to java A {
        map b1 to a1;
        // ...
    }
    mapping from java A to type B {
        map a1 to b1;
        // ...
    }
    mapping from java A to java B {
        map a1 to b1;
        // ...
    }
}
```

Generated Java Code:

```java
class MappedStructure {
    String a1;
    int b1;

    public void populate(A a) {
        a1 = a.a1;
    }

    public void populate(B b) {
        b1 = b.b1;
    }

    public B toB(A a) {
        B result = new B();
        result.b1 = Integer.valueOf(a.a1).intValue();
        return result;
    }

    public A toA(B b) {
        A result = new A();
        result.a1 = String.valueOf(b.b1);
        return result;
    }

    // ...
}
```
Mapping (ii) of Model and/or Java Types

IcmDsl:

```
model Mapping_Ex1 {
    type jvm java.sql.Date SqlDate;

    structure A {
        string a1;
        integer a2;
        double a3;
        SqlDate a4;
        long a5;
    }

    structure B {
        string b1;

        mapping of type A {
            map a1 to value;
            map a2 to type float;
            map a3 to java ^java.util.Calendar myDate;
            map a4 to java ^java.util.Calendar myDate;
        }
    }
}
```

Generated Java Code:

```
class A {
    String a1;
    int a2;
    double a3;
    java.util.Date a4;
    long a5;
}

class B {
    String b1;

    public void populate(A a) {
        a1 = a.a1;
        value = a.a2;
        a3 = Double.valueOf(a.a3).floatValue();
        myDate = java.util.Calendar.getInstance();
        myDate.setTime(a.a4);
    }
}
```
Mapping (iii) Type Selection with the "."-Dot-Notation

IcmDsl:

```java
import java.util.ArrayList;
import java.util.List;

class C {
    String c1;

    public String getC1() {
        return c1;
    }

    public void setC1(String c1) {
        this.c1 = c1;
    }

    public List<B> populate(List<A> input) {
        List<B> out = new ArrayList<B>();
        for (A a : input) {
            B b = new B();
            b.setB1(a.getB1());
            b.setB2(a.getB2());
            out.add(b);
        }
        return out;
    }
}
```

Mapping of Types on Field definitions using the dot ("." ) notation. Cardinality is of course also supported...
Mapping of Service Structures to Business Objects

```java
... businessObject CifDetails
@MappedBy(GetCifsShortDataMapper_1_0)
    mapping of java CifShortStruc_T
        map address;
        map birthDate to type date;
        map buId;
        map cifNo;
        map cifUwi;
        map creditOfficer;
        map deceasedDate to type date;
        map incorpDate to type date;
        map nationalityCd to nationalityCode;
        map domicileCd to domicileCode;
        map customerLanguageCd to customerLanguageCode;
        map customerStateCd to customerStateCode;
        map custTypeCd to customerTypeCode;
        map custSegCd to customerSegmentCode;
        map maritalStateCd to maritalStateCode;
        map occupationCd to occupationCode;
        map economicSector;
        map relationManager;
        map customerOffice;
        map phoneNoP to phoneNumberPrivate;
        map phoneNoE to phoneNumberBusiness;
        map shortAddress to shortAddress;
    }

dataMapper
```

---

Produced by: Serano Colameo
itemis Switzerland
Date: 18/10/2012 Slide 24
Mapping of Business Objects to Transfer Objects

"User data Input Transfer Object"

```java
transferObject UserData INTO {
    "This mapping is based on the CifDetails BO"
    mapping of type CifDetails {
        select cifNo, buId, address, cifUwi // only these fields
    }
}
```

Mapping of Business Objects fields to Transfer Objects allows no type conversion

"User data Output Transfer Object"

```java
transferObject UserData TO {
    mapping of type CifDetails; // map all fields
}
```

No «select» keyword means map all fields
Mappings are managed by Data Mappers

```java
/** * Data Mapper for Cif Details */

@Incomplete
dataMapper of service CIFS_GetDetailForCifs_1_0::getCifsShortDataMapper_1_0 { as GetCifsShortDataMapper_1_0 {
    structure UserDataDMTO {
        mapping of java:InputStruk_T {
            map cifUwiStruk.cifNoIntMec.cifNo;
            map cifUwiStruk.cifNoIntMec.builId;
        }
        in UserDataDMTO[*] userDataDMTOs;
        out CifDetails[*];
    }
} in UserDataDMTO[*] userDataDMTOs;

"Business object holding the customer information details like number, date, code etc."

businessObject CifDetails {
    Remark[*] remarks;
}

@MappedBy(GetCifsShortDataMapper_1_0)
mapping of java:CifShort_T {
    map cifShort.cifNo;
    map cifShort.builId;
    map cifShort.birthDate to type date;
    map cifShort.deceasedDate to type date;
    map cifShort.incorpDate to type date incorporationDate;
    map cifShort.nationalityCd to nationalityCode;
    map cifShort.domicileCd to domicileCode;
    map cifShort.customerLanguageCd to customerLanguageCode;
}
```
MCP – Location Specific Customization (I)

Reusing of façade API and business logic but with different backends!

```java
model CustomizationFacadeEx1 {
    businessObject AccountBO {
        string build;
        string cifNo;
        string firstName;
        string lastName;
    }

    facade Customer version 1.0 {
        transferObject CustomerAccountTO {
            mapping of type AccountBO
        }

        CustomerAccountTO_1_0 getAccountInfo(string firstName, string lastName);
    }
}
```
MCP – Location Specific Customization (II)

Customization of Data Mapper Definitions by extending the model and tagging them

```java
model Customer 3 {
    criteria BusinessUnits {
        CH include "0011";
        SI include "0090";
    }

    @ConstraintOn(tagName="CH", businessUnit=BusinessUnits.CH)
dataMapper of service CUPA BD_GetAgreementsOfCifs_1_0 as GetDepositNosWithCifNosDataMapper_1_0 {
        @Incomplete mapping from java GetAgreementsOfCifs_1_0Out to type CifRelatedData[*] {
            @Incomplete
            map _sysEx to remarks;
        }

        in string[*] cifNumbers, string theBusinessUnit;
        out CifRelatedData[*];
    }

    @MappedBy(GetDepositNosWithCifNosDataMapper_1_0)
    businessObject CifRelatedData {
        string cifNumber;
        string[*] depositNumbers;
        Remark[*] remarks;
    }
}

model Customer 3SI extends Customer 3 {
    @ConstraintOn(tagName="SI", businessUnit=BusinessUnits.SI)
dataMapper of service CUPA BD_GetAgreementsOfCifs_1_0 as GetAgreementsOfCifs_1_0DataMapper_1_0 {
        @Incomplete mapping from java GetAgreementsOfCifs_1_0Out to type CifRelatedData[*] {
            @Incomplete
            map _sysEx to remarks;
        }

        in string[*] cifNumbers, string theBusinessUnit;
        out CifRelatedData[*];
    }
}
```
Questions