MOF Model to Text Transformation - MOFScript

The present courseware has been elaborated in the context of the MODELWARE European IST FP6 project (http://www.modelware-ist.org/).
Outline

• Background: Model to text transformation
• Introducing the MOFScript language
• MOFScript and the MOF
• MOFScript and the relation to QVT
• Details of the MOFScript language
• Example1: From UML class diagram to Java
• Example2: From UML activity models to BPEL (to difficult?)
• MOFScript advanced features
• The MOFScript tool

Background: Model to text transformation
The MOF Model to Text Transformation RFP

- The Model to Text Transformations RFP (ad/2004-04-07) asks for proposals that define a language that provides capabilities for generating text output from MOF models, e.g. in order to support code generation, documentation, etc. Important in this regard, is capabilities to perform string manipulation.

- Proposals shall define a solution for generating text from MOF 2.0 models.
- Proposals shall reuse existing OMG language specifications (e.g. such as QVT capabilities)
- Proposals shall be able to support complex text transformation mappings.

MOF Script - background

- Usability
  - Ease of use: Writing and understanding
  - Few constructs
- End user recognizability
  - Similar to programming and scripting languages
- Sequential execution semantics
  - Rules are called explicitly
    - Might also support pattern matching execution
  - Contents of rules is executed sequentially
- Compatibility
  - Alignment with QVT
Introducing the MOFScript language

What is MOFScript?

- The MOFScript tool is an implementation of the MOFScript model to text transformation language
- An Eclipse plug-in (Feature)
- Developed by SINTEF ICT in the EU-supported MODELWARE project
- Ongoing standardization process within OMG
  - OMG RFP MOF Model to Text Transformation process
  - MOFScript tool and language is part of this process
  - Competing proposal: ...
MOFScript in action

MOF MODELS

UML
RDBMS
BPMN

MOFScript

Lexical output

Program code
XML
Documentation

MOFScript language features [1]

• Rules
  • Rules are invoked explicitly.

• Output of text
  • Standard print mechanism (println)
  • Escaped output

• Files
  • Generation of named files

• Collections and Iterators
  • Collection types (List, Hashtable/Dictionary)
  • Iterators: Loops on collections (or Strings/Integers)
MOFScript language features [2]

- String handling
  - Functions for String manipulation

- Types
  - String, Boolean, Integer, Real, List, Dictionary/Hashtable

- Variables and constants
  - Global or local, explicitly or implicitly typed.

- Utility functions
  - Utility & system functions: time(), date(), getenv(), setenv(), indent(), tab(), space(), position(), count()

- Side effects
  - Variables can be modified
  - MOF Script has side effects, in the sense that variables can be declared and manipulated during the lifetime of a transformation. It does not, however affect the source models.

Basic QVT extensions
TextTransformation

OperationalTransformation

MappingOperation

MappingBody

TextTransformation

TextMappingRule

TextMappingBody

MappingBody

MappingSection

OclExpression

TextMappingBody

TextMappingSection

MappingBody

MappingSection

OclExpression

MappingBody

MappingSection

OclExpression

TextMappingBody

TextMappingSection

MappingBody

MappingSection

OclExpression

TextMappingBody

TextMappingSection

MappingBody

MappingSection

OclExpression

TextMappingBody

TextMappingSection

MappingBody

MappingSection

OclExpression

TextMappingBody

TextMappingSection
Variables and constants

- VariableDeclaration
- VariableInitExp
- OclExpression

Output Expressions

- OutputExpression
- OutputDeviceExpression
- FileDeviceExpression
- StdOutDeviceExpression
- EscapedOutputExpression
Iterator extensions

- QVT defines a range of iterators and block expressions
- QVT also defines block expressions which are blocks executed in sequence
  - E.g. while

```
self.ownedMembers->forEach (p:uml.Package | p.stereotype = "System") |
  <% // the package declaration
      package org.mofscript.%>p.name <%
  %>
  if (self.name = "Person") |
    <% This is the person type %>
  else |
    <% This is not a person type %>
```

- ForEachExp and If also as block statements

Function libraries

- String library
  - Various string manipulation functions, such as:
    - Size, substring, substringBefore|After, toLower, toUpper, indexOf, trim, endsWith, startsWith, replace, equals, equalsIgnoreCase,

- Collection library
  - Standard collection operations...
    - Hashtable: put, get, clear, size, isEmpty
    - List: add, size, clear, isEmpty

- Utility library
  - Various utility functions, such as
    - Time, date, Indent, unindent, newline(count: int), tab(count : int), position()
      (numbered position within an iterator), ...

- XML library
  - Functions to support XML manipulation
Details of the MOFScript language

Textual syntax

- Textmodule
- Rules
- Iterators
- Files
- Escaped output

```
textmodule UML2Java (in uml:uml2)
access library variousJavaStuff
("uml2java_include.m2t")

umlPackage::mapPackage () { self.ownedMember->forEach(c:uml.Class)
  <%
  <class name="%> c.name <% "/>
  %>
}

file f2 (c.name + ".java")
<% package %> c.ownerPackage.name <% %>
f2.println ("public class" + c.name);

<% public class %> c.name <% extends Serializable {
%>
```
Textual syntax

- If expression
- Printing
- Collections, Vars

```java
if (self.name == "Person") {
    // This is the person type
} else {
    // This is not a person type
}

stdout.println("\t Printing to standard out...")
dclFile.println("Print to file " + c.name)
print("\n\n print to declared context")
<% escaped output print …
… %> c.name <% … %>
```

```java
var packageNames:List
var packageIdList:Hashtable
self.ownedMember->forEach(p:uml.Package) {
    packageNames.add(p.name)
    packageIdList.put(p.id, p.name)
}
if (packageIdList.size() > 0) {
    // Listing the package names that does not start with 'S'
    packageIdList->forEach(s:String | not(s.startsWith("S")) {
        // Package: s
    }
}
```

```java
uml.Package::interfacePackages () {
    if (self.getStereotype() == "Service"){
        file (rootdir + self.name.toLower() + ".wsdl")
        self.wsdlHeader()
        self.wsdlTypes()
        self.ownedMember->forEach(i:uml.Interface) {
            i.wsdlMessages()
            i.wsdlPortType()
            i.wsdlBindings()
            i.wsdlService()
        }
        self.wsdlFooter()
    }
}
```

```java
uml.TypedElement::getType () : String {
    if (self.type.name.equalsIgnoreCase("string"))
        result = "xsd:string"
    else
        result = self.type.name
}
```
MOFScript extends QVT Mappings

MOFScript a transformation language

- Language for writing model to text transformations
- Rules / Operations are called explicit (Procedural language)
- Partly based on the current QVT specification (keeps it within the family)
- Transforms input models to output text files
  - Input: UML, ecore or your own based on it
- Supports EMF meta-models and models as input
  - UML2, Ecore, etc..
General syntax of a mapping operation

Fra siste QVTMerge-spec:
The general syntax for the body of a mapping operation is:

```
mapping <dirkind0> X::mappingname
(<dirkind1> p1:P1, <dirkind2> p2:P2) : r1:R1, r2:R2
  when { ... }
  where { ... }
  {
    init { ... }
    population { ... }
    end { ... }
  }
```

Example 1: From UML class diagram to Java
Example (1)

textmodule UML2Java (in m1:UML)

    property package_name = "org.sintef.no"
    property package_dir = "org/sintef/no/"
    property ext = ".java"
    property author = "Jon Oldevik"

    /*
     * Entrypoint
     */
    main () {
        m1.ownedMember->forEach(p:uml.Package)
            p.mapPackage()
    }

    /*
     * mapPackage
     */
    void uml.Package::mapPackage () {
        p.ownedMember->forEach(c:uml.Class)
            c.mapClass()
    }

Example (2)

void uml.Class::mapClass () {

    file f1 (name=self.name, ext="txt", dir=package_dir, type="text")
    file f2 ("Test.dilldall")
    use file f3 (package_dir + self.name + ext)
    self.standardClassImport ()
    self.standardClassHeaderComment ()
    %
    public class % c.name <% extends Serializable {
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
    %>
Example (3)

// classPrivateAttribute
void uml.Property::classPrivateAttribute (uml.Class owner) {

    private String _%>
    self.name.toLowerCase() <% ; %> <% // Visibility:%>
    self.visibility <%
    // Type : ... %> self.type
    println(" // This was the private attributes 

    

    }

MOFScript advanced features
Change management

- Proposed metamodel for handling:
  - Links from a model toward code blocks
  - Traces from a model to text blocks
  - Controlled by user
    ```
    unprotect {
        <!-- User writes code here -->
    }
    ```
  - The resulting code will be generated with protected blocks
    - Identifying the start and end the section
    - Code is protected by user-defined tags
    - E.g. comment tags

Traceability model
What is MOFScript?

- An Eclipse plug-in (Feature)
- Developed at Sintef ICT
MOFScript architecture

MOFScript a model to text tool

- Provides the means of:
  - Editing, compiling and executing the transformations
  - Syntax high-lightening
  - Content assist
GUI overview

![GUI Image]

Tool cont.

- Editor
  - Syntax highlighting
  - Context sensitive / content assist
- Outline viewer
  - View the
- Preference pages
  - Setting global preferences
- File properties
  - Setting preferences for a single transformation file
- Actions
  - Compile, execute, convert to model,...
References

- OMG MOF Model to Text Transformation RFP
- MOFScript submission
- MOFScript tool
  - [http://www.modelbased.net/mofscript](http://www.modelbased.net/mofscript)

Final merged OMG MOF2Text proposal expected during 2006, (April 2006)
Submission Structure

- Hybrid Declarative/Imperative Architecture
- Common trace mechanism
  - Stored relationships between transformed model elements
- Two level declarative model

MOF & OCL Model Extension
MDI – Model Driven Interoperability

Context for metamodels – for Enterprise level and Systems

Mål for resten av kurset: Metamodellering for Enterprise (POP*) og System (PIM4SOA) modeller, som vist her.

www.athena-ip.org

ATHENA Fact Sheet

- One of the largest IT research programmes funded by the European Commission (Budget: 26.5 Mio€)
- Duration: 3 Years Overall Effort: 170 PY
- Strategic Objective 2.3.1.9: Networked Businesses and Governments
- Main goal of ATHENA: achieve impact in industry
- Approach
  - Holistic, technology based
  - Guided by user requirements
  - Complemented by business and community building activities.
ATHENA – Consortium

**Industry Partners**
- CR FIAT (I)
- EADS CCR (F)
- INTRACOM (GR)
- AIDIMA (SP)

**Technology Providers**
- SAP (D)
- IBM (UK)
- SIEMENS (D)
- TXT (I)
- Gruppo FORMULA (I)
- ITREC (F)
- TROUX (NO)

**Academic / Research Partners**
- FHG/IPK (D)
- DFKI (D)
- SINTEF (NO)
- ESI (SP)
- LEKS/IASI-CNR (I)
- Univ. BORDEAUX I (FR)
- UNINOVA (PT)
- IC FOCUS (UK)
- INSEAD (FR)
- Univ. St. Gallen (CH)

---

**Vision & Objective of ATHENA**

**Vision:**
*By 2010, enterprises will be able to seamlessly interoperate with others*

**Objective:**
*Contribution to enabling enterprises to seamlessly interoperate with others*
ATHENA Interoperability areas – The ATHENA railroad

ATHENA Interoperability areas

Additional Dimensions:
AIF: Conceptual, Technical, Applicative
MDA: PIM, PSM (Enterprise models– CIM)
ATHENA Interoperability Framework

Conceptual Framework:
- Concepts, meta-models, languages & model relations

Technical Framework:
- Development Framework & Execution Environment

Applicative Framework:
- Methodology, Standards & Domain Profiles

Logical Repository

Networked Businesses

Enterprise A
- Business
- Knowledge
- ICT Systems

Enterprise B
- Business
- Knowledge
- ICT Systems
Conceptual Framework

Enterprise System A (MDI Abstraction)

- Computational Independent Model (CIM)
- Platform Independent Model (PIM)
- Platform Specific Model (PSM)
- Execution Platform A
- Ontologies

Enterprise System B (MDI Abstraction)

- Computational Independent Model (CIM)
- Platform Independent Model (PIM)
- Platform Specific Model (PSM)
- Execution Platform B
- Ontologies

Reference Ontology

Semantic Annotation

Model Interoperability

Horizontal Interoperability

Vertical Interoperability

Reference Ontology

Semantic Annotation

Model Interoperability

Horizontal Interoperability

Vertical Interoperability

Method chunks for SOA and Web Service Interoperability

OWL Ontology

annotated with

OWL-S Document

annotated with

XSD Document

annotated with

WSDL Document

annotated with

BPEL Document

annotated with

BDI Document

annotated with

WS-* Document

annotated with

ATHENA Service Execution Infrastructure
Collaborative Enterprise Modelling (A1)
POP* Metamodel

OMG Model Driven Architecture (MDA)

Microsoft DSL
Domain Specific Languages
Potential Components of Enterprise Modeling

Enterprise Model

- Strategic Plan
- Value Chain
- Business Cycles
- Financial Plan
- Motivation
- Locations
- Resources
- Competencies
- Processes
- Organization
- Rules
- Vocabulary
- Financial Plan
- Motivation
- Business Cycles
- Value Chain
- Strategic Plan

Current Work
Potential Development

Business Motivation Model

- Processes
- Rules
- Means
  - Mission
  - Strategy
  - Tactic
  - Guidance: Policies & Rules
- Organization Unit
- Ends
  - Vision
  - Goal
  - Objective
- Influencers
  - Internal
  - External
- Assessment
  - Strengths
  - Weaknesses
  - Opportunities
  - Threats
- Impact Value
  - Risk
  - Reward
Organization Structure Metamodel

- Organization unit
- Position
- Authority
- Responsibility
- Relationships
- Locations
- Resource ownership
- Organization rules
- Modeling vs. runtime

Business Process Definition Metamodel (BPDM)

- XPDL Model
  - Transformation
- BPMN Model
  - Transformation
- BPEL Model
  - Transformation
- Proprietary Model
  - Transformation
- BPDM Model
Business Process Runtime Interfaces

Platform Independent Model for runtime business process systems

Process Request

Business Process

Work lists (people)

Activity  Activity  Activity

Collaborating Business Process

Event Publication

Invoked Business Process

Enterprise model coordination

ARIS  MO²GO  Rational Software Modeler POP* Profile  Metis  GRAI

Web Client

Web Server

Metis Team 3.6 Repository

MPCE Web Services (Standard WSDL/SOAP)

EKA Services

© ATHENA Consortium 2005

© ATHENA Consortium 2005
Cross-Organisational Business Processes (A2)
CBP Modelling Method

Business Level Processes

Private Process
View Process
CBP

Technical Level Processes

Private Process
View Process
CBP

Transformation semi-automated with manual steps

Transformation automated

Execution Level Processes

Private Process
View Process

Mapping with tool support

Abstract Technology Model

Synchron. request
Deferred Synch request
Message
Streaming
Event - publish & subscribe
Naming service
Trading service

Interaction/Pres services
User services (application/process)
Shared Business Services
Data services & Legacy systems
Integration service

XML
Workflow service
System/Use Mgmt
Server Components
Concurrency service
Transaction service
Persistence service

Multi Media, QoS

User Interface
Document model
Web interaction
Mapping from PIM4SOA to specific platforms (A5)

PIM4SOA Metamodels

- Four interconnected metamodels
- Focus so far on service, information, process
A6 PIM4SOA architecture & transformations overview

PIM4SOA addresses four system aspects

Metamodel for (software) services

- Services are an abstraction and an encapsulation of the functionality provided by an autonomous entity. Service architectures are composed of functions provided by a system or a set of systems to achieve a shared goal.
- Web Services Architecture as proposed by W3C (W3C 2004)
- UML Profile for Enterprise Distributed Object Computing (OMG 2002)

Metamodel for (automated software) processes

- Processes describe sequencing of work in terms of actions, control flows, information flows, interactions, protocols, etc.
- Business Process Definition Metamodel (BPDM) (IBM et al. 2004)
- UML Profile for Enterprise Distributed Object Computing (OMG 2002)

Metamodel for information

- Information is related to the messages or structures exchanged, processed and stored by software systems or software components.
- Structural constructs for class modelling in UML 2.0 (OMG 2003)
- UML Profile for Enterprise Distributed Object Computing (OMG 2002)

Metamodel for quality of service (QoS)

- Extra-functional qualities that can be applied to services, information and processes.
- UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms (OMG 2004)
**WSDL 1.1 metamodel**

- **Service**: A container for data type definitions
- **Port**: A single endpoint defined as a combination of a binding and a network address
- **Port Type**: A concrete protocol and data format specification for a particular port type
- **Operation**: An abstract set of operations supported by one or more endpoints
- **Message**: An abstract, typed definition of the data being communicated

**Definitions**

- **Interface**:
  - Name: wsdls_NCName
  - Target namespace: wsdls_anyURI
- **Operation**:
  - Name: wsdls_NCName
  - Target namespace: wsdls_anyURI
  - Message exchange pattern: wsdls_anyURI
  - Style: wsdls_anyURI
  - Safety: wsdls_boolean
- **Message Reference**:
  - Message label: wsdls_NCName
  - Direction: wsdls_token
  - Message content model: wsdls_token
- **Interface Operation**:
  - Interface Reference: wsdls_NCName
- **Binding**:
  - Name: wsdls_NCName
  - Target namespace: wsdls_anyURI
  - Type: wsdls_anyURI
  - Interface Reference: wsdls_NCName
- **Service**:
  - Name: wsdls_NCName
  - Target namespace: wsdls_anyURI
  - Interfaces: 0..*
- **Property**:
  - Name: wsdls_anyURI
  - Required: wsdls_boolean
- **Feature**:
  - Name: wsdls_anyURI
  - Required: wsdls_boolean

**WSDL 2.0 metamodel**

- **Definitions**
- **Interface**:
  - Name: wsdls_NCName
  - Target namespace: wsdls_anyURI
- **Operation**:
  - Reference: wsdls_NCName
- **Binding**:
  - Message Reference: wsdls_NCName
  - Fault Reference: wsdls_NCName
- **Service**:
  - Name: wsdls_NCName
  - Target namespace: wsdls_anyURI
  - Interfaces: 0..*
  - Faults: 0..*
  - Operations: 0..*
- **Property**:
  - Name: wsdls_anyURI
  - Required: wsdls_boolean
- **Feature**:
  - Name: wsdls_anyURI
  - Required: wsdls_boolean