INF5120: Lecture #3 (F3) 5 February 2007

UML Profiles and Domain-Specific Languages (DSLs)

Brian Elvesæter, SINTEF ICT
brian.elvesater@sintef.no

ATHENA Model-Driven Interoperability (MDI) Framework

- MDA & Interoperability
- Metamodelling
- UML Profiles & DSLs
- Model Transformations
- Method Engineering

Reusable MDI Assets
- Method chunks
- Tools and services
- Models and metamodels
- Model transformations
- DSLs and UML profiles
- Reference examples
Outline

• Metamodelling (revisited)
• UML profiles
• Domain-specific languages (DSLs)
• Eclipse technologies
  – Eclipse Modeling Framework (EMF)
  – Eclipse Graphical Modeling Framework (GMF)
• Example
  – Metamodels and UML profiles for SOA and Web services
• References

Metamodelling (revisited)
Three stages in the evolution of modelling techniques at the OMG.

1. (a) UML to aModel
2. (b) aModel to MOF
3. (c) MOF to other specifications (UML, SPEM, etc.)

- UML
- MOF
- aModel
- UML for CORBA
- Action language
- Workflow, etc.
- Common Warehouse Metadata
Egyptian architecture

- **M₀**: "the real world"
- **M₁**: model
- **M₂**: metamodel
- **M₃**: meta-metamodel (The MOF)

The MOF
- The UML meta-model and other MM's
- Some UML models and other M's
- Various usages of these models

Illustration

- **M₁**: UML model
- **M₂**: Java program
- **M₃**: Java MM UML MM

Class
- Interface
- Package
- Presentation
- Participant

Association
- Extends
- listens

Presentation
- UML model
The three modelling levels

- **M^3 level**: the MOF MMM
- **M^2 level**: the UPM MM (SPEM) → the UML MM → the CWM MM
- **M^1 level**: a UML model m → another UML model m'
- **M^0 level**: a particular use of m → another use of m

Model -> Metamodel

- entity → meta-entity relationship
- model → meta-model relationship

UML MM

- Class
- Attribute

UML model

- Client
- Name: String
Metamodel -> Meta-metamodel

MOF Model (M3)
A statemachine consists of a number of states. States can be start states, stop states and “normal” states.

A transition connects two states. States know their outgoing and incoming transitions.

We also support composite states that themselves contain sub state machines.

A state machine is itself a composite state.

A state has actions. Actions can either be entry or exit actions.
UML profiles

• They allow us to adapt the UML language to the needs of the analysts or the application domain
• Allows designers to model using application domain concepts.
• There are three extension mechanisms:
  – Stereotypes
  – Restrictions
  – Tagged values
### Stereotype

- **Extends** the vocabulary of UML with **new construction elements** derived from **existing** UML but specific to a problem domain
- Can have associated **restrictions and tagged values**
- Possibility of assigning an **icon** for a better graphical representation

```
<<Database>>
DB Partners
```

```
<<Interface>>
SQLQuery
- Query()
```

### Restriction

- Is a semantical **condition** represented by a **textual expression**
- Imposes some kind of condition or **requisite on the element** to which it is applied
- **OCL** – Object Constraint Language

```
(SQLQuery does not have attributes, only operations)
```

```
<<Interface>>
SQLQuery
+ Query()
```
Tagged value

- Is a **property** associated to a **model element**
- Used to store **information** about the element
  - Management information, documentation, coding parameters, ...
- Generally, the **tools** store this information but it is **not shown in the diagrams**

---

Metamodels and profiles

- **MOF**
- **UML**
  - **Real-time**
  - **For J2EE**
- **Migration oriented process Meta-model**
- **Workflow Meta-model**
- **Workflow model**
- **Migration model**
- **real-time model**
Classification

Living Being

Four Legged Object

Celebrity

Movie Star

Animal

Dog

Collie

Lassie

Classification dimensions

Ontological classification (domain types)

Animal

Dog

Collie

Celebrity

Movie Star

Linguistic classification (representation form)

Model Element

Object

Instance

Four Legged Object
Kinds of metamodels

- Two kinds of information of a set of models are modelled in metamodels
  - Form (linguistic aspects)
    - OMG is predominantly occupied with this
  - Content (ontological aspects)

Linguistic metamodelling

- L3 (called M3 in OMG) – Class
- L2 (called M2 in OMG) – Class to Object
- L1 (called M1 in OMG) – Collie to Lassie
- L0 (called M0 in OMG) – instance-of
Ontological metamodelling

Domain-specific languages (DSLs)
UML – one size fits all?

- While the OMG MDA promotes UML as the visual “universal” glue suitable for modelling everything, we are also seeing a trend towards development and co-existence of several domain-specific modelling languages, e.g. supported by the Microsoft Domain-Specific Language (DSL) tools (http://lab.msdn.microsoft.com/teamsystem/workshop/dsltools/default.aspx).
- Such approaches are now also being discussed in various OMG forums.
- UML is seen as a “general-purpose” language while DSLs may be more expressive for most purposes.
- A model-driven framework needs to acknowledge the existence of different models and views expressed in different modelling languages.
- The MDA technologies can help us to align these models through a common metamodelling language on which model transformations and model mappings can be defined.

Software factory

- The Software Factories Web site (http://www.softwarefactories.com/) defines the term Software Factory in the following way:
- “A Software Factory is a software product line that configures extensible development tools like Visual Studio Team System with packaged content like DSLs, patterns, frameworks and guidance, based on recipes for building specific kinds of applications. For example, we might set up a Software Factory for thin client Customer Relationship Management (CRM) applications using the .NET framework, C#, the Microsoft Business Framework, Microsoft SQL Server, and the Microsoft Host Integration Server. Equipped with this factory, we could rapidly punch out an endless variety of CRM applications, each containing unique features based on the unique requirements of specific customers. Better yet, we could use this factory to create an ecosystem, by making it available to third parties, who could extend it to rapidly build CRM applications incorporating their value added extensions.”
UML and DSLs

- The issue of the role of UML is often stated in overly simplistic terms: MDD advocates the use of UML for all domain modelling while the Software Factories approach advocates that UML never used.
- This is an incorrect statement of the positions of both camps.
  - While the MDD approach treats UML, with customization, as the modelling language of choice for most application modelling, it also acknowledges the value of custom languages in certain specialized circumstances.
  - This is the purpose of the OMG Meta-Object Facility (MOF) standard that plays an important role in MDD. UML itself is defined using MOF and there are MOF definitions of many other languages.
  - The MDD approach acknowledges the value of non-UML DSLs as a technique to be applied judiciously.
  - Further, the Software Factories approach does not reject UML entirely. It suggests that you use UML for developing sketches and documentation, where DSLs should be used for developing models from which code is generated.

Advantages of using UML profiles

- UML is an open standard modelling language for which there are many available books and training courses.
- UML profiles provide a lightweight approach that is easily implemented using readily available UML tooling.
- Models with UML profiles applied can be read by all UML tools even if they do not have any knowledge of the profile.
- Basing all DSLs on UML creates a set of related languages that share common concepts.
- UML can be used for high-level architectural models as well as detailed models from which code can be generated.
Disadvantages of using UML profiles

- UML profiles only permit a limited amount of customization.
  - It is not possible to introduce new modelling concepts that cannot be expressed by extending existing UML elements.
- The use of UML does require familiarity with modelling concepts.

Eclipse Technologies: Eclipse Modeling Framework (EMF)
Eclipse Modelling Framework (EMF 2.2.0)

- Java Framework and code generation facility
- Evolved implementation of MOF specifications
- Unifying of Java, XML and parts of UML
- Standard serialization in form of XMI
- Uses Ecore to represent EMF models
  - MOF-like core meta model
  - Ecore is also an EMF model and therefore its own meta-model

Making the metamodel(EClass)
Making the metamodel (Generalization)

Eclipse Technologies: Eclipse Graphical Modeling Framework (GMF)
Graphical Modelling Framework (GMF)

- The Eclipse Graphical Modeling Framework (GMF) provides a generative component and runtime infrastructure for developing graphical editors based on EMF and GEF.

- GMF v.1.0 released June 28, 2006

GMF overview

- GMF uses and depends on EMF and GEF
  - EMF:
    - Core facility for defining models
    - generates java code for manipulating and persisting model instances
  - GEF:
    - Manipulates object models
- Simplifies combination of EMF and GEF
- Generates GEF editors for manipulating EMF models
Editor Generation Overview

Create MIF Project

Domain Model (*, acres)

Develop Graphical Definition

Create Generator Model

Adjust Generation Parameters

Generate Diagram plug-in

The Editor
Example #1: Metamodels and UML profiles for service-oriented architectures (SOAs) and Web services

Metamodel and profile development

- Understanding of SOA concepts and interoperability issues
  - Initial (interoperability) requirements
  - SOA concepts
  - Partitioning of the metamodel into structures
  - Architectural style for developing interoperable software systems
  - Document the metamodel in RSM (.uml2) and develop it in EMF (.ecore)

1. Metamodel scope, concepts, style

2. UML profile
   - Map metamodel concepts to UML

3. Test
   - Evaluate in user scenarios

4. Feedback
   - Add, remove, and modify concepts

Feedback
- Iterative development
- Modifications according to test and evaluation

Define how SOA models should be specified using UML
- Map metamodel concepts to UML types
- Define UML stereotypes and tagged values
- Define new visual symbols
- Define model views
- Define validation rules
- Implement UML profile in RSM

Evaluate and test UML profile in user scenarios
- Four ATHENA use cases (photos)
**Characteristics for metamodel**

- **Suited for target roles**
  - Support domain concepts and scenarios of target roles
  - Ease-of-use and understandable for business modeller (use terms)
  - Support precise details and correctness for solution architect
- **Avoid unnecessary complexity**
  - Keep it simple stupid (KISS)
  - Number of elements and associations
  - Type and navigation of associations
- **Make it modular**
  - Provide core with extensions
  - Define and illustrate possible subsets ("dialects") that support scenarios
  - Consider integration and extension points
- **Suited for implementation**
  - EMF representation
  - Transformation from/to UML profile
  - Transformation to PSM

---

**PIM4SOA objectives**

- **Platform independent model for specifying service-oriented architectures**
  - Represent SOA solutions in a platform independent way
  - Integrate and define mappings to Web services, agents, peer-to-peer (P2P) and Grid execution platforms.
  - Bridging the gap between the enterprise layer and the technical layer
  - Establishing relationships between layers through model-based transformations
  - Two-way transformations supporting both
    - model-driven development (MDD); and
    - architecture-driven modernisation (ADM)
PIM4SOA requirements

Depending on the source of requirements

- From the enterprise or business viewpoint
  - Process, Organisation, Product and System (POPS) dimensions
  - Mapping enterprise and business model elements to PIM4SOA

- From the platform point of view
  - What are the necessary PSM elements to be represented at PIM level?
  - How do we identify these elements?
  - We need identify overlapping elements amongst platforms
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)
PIM4SOA metamodel description

- **Service Oriented Metamodel**
  - has the objective of describing service architectures as proposed by the W3C
  - represent the functionalities provided by a system or a set of systems to achieve a shared goal

- **Information Oriented Metamodel**
  - starting point are the UML constructs used in “plain vanilla” class modelling
  - based on EDOC as well as on the Class related parts of the UML metamodel

- **Process Oriented Metamodel**
- **Non-functional oriented metamodel**

Information metamodel
PIM datatypes

- Requirements
  - Base types: There must be a small set of base types that represents the basic needs for identifying types for model properties.
  - Constructed types: It must be possible to construct more complex data types from simple ones (base types).
  - Platform independence: The types for PIM modelling should be independent of any target platform or language and not assume any specific representation for a type.
  - Constraints on types: Constraints for types should be supported.
  - Types as parameters – what are their semantics?

UML extensions for PIM4SOA modelling

- UML extensions for Service Modelling
- UML Extensions for Process Modelling
- UML Profile for Information Modelling
- UML Extensions for QoS Modelling
### UML profile for XSD (1)

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>UML construct</th>
<th>Tagged value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;&lt;any&gt;&gt;</td>
<td>Class, Property</td>
<td></td>
<td>The stereotyped class or attribute will be replaced by an 'any' or 'anyAttribute' element. The tagged values are copied into the corresponding attributes of the generated element.</td>
</tr>
<tr>
<td>namespace</td>
<td>Class</td>
<td></td>
<td>As defined in XML Schema specification.</td>
</tr>
<tr>
<td>processContents</td>
<td>Class</td>
<td></td>
<td>As defined in XML Schema specification.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>values=&quot;strict</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>default=&quot;strict&quot;</td>
<td></td>
</tr>
<tr>
<td>&lt;&lt;attribute&gt;&gt;</td>
<td>Property</td>
<td></td>
<td>Assigned to UML attribute or association end. Indicates item is to be generated as an attribute within complexType and not as an element.</td>
</tr>
<tr>
<td>default</td>
<td></td>
<td></td>
<td>As defined in XML Schema specification.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>fixed</td>
<td>As defined in XML Schema specification.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>form</td>
<td>Overrides the attributeFormDefault for this schema.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>values=&quot;qualified</td>
</tr>
<tr>
<td></td>
<td></td>
<td>use</td>
<td>As defined in XML Schema specification.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>values=&quot;prohibited</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>default=&quot;optional&quot;</td>
</tr>
<tr>
<td>&lt;&lt;choice&gt;&gt;</td>
<td>Class</td>
<td></td>
<td>Elements marked with this stereotype represent a Choice model group contained within a complexType definition.</td>
</tr>
<tr>
<td>&lt;&lt;complexType&gt;&gt;</td>
<td>Class</td>
<td></td>
<td>ComplexType definition generated in XML Schema.</td>
</tr>
<tr>
<td>memberNames</td>
<td></td>
<td></td>
<td>Overrides the package-level default for naming complexType definitions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>values=&quot;qualified</td>
</tr>
</tbody>
</table>

### PIM4SOA → platform specific models

- **PIM**
- **PIM4SOA Metamodel**
- **Web Services Metamodel**
- **Agent Metamodel (AgentMM)**
- **P2P Metamodel**
- **Grid Metamodel**
Web services architecture metamodel

UML profile for Web services

- Web Service Basic Profile
  - This plugin implements the base UML modelling support for how to specify and develop Web services.
  - This plugin can be extended with other specific Web service technology profiles (e.g. XSD, WSDL and BPEL) each covering different parts of the Web service stack.

- UML Profile for XSD
  - This plugin implements UML modelling support for XML schema definitions

- UML Library for XSD
  - This plugin defines XSD types to be used when modelling XML schema definitions in UML.

- UML Profile for WSDL
  - This plugin implements UML modelling support for WSDL.

- UML Profile for BPEL
  - This plugin implements UML modelling support for BPEL.
XSD metamodel

XSD metamodel (simplified)
### UML profile for XSD (1)

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>UML construct</th>
<th>Tagged value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;&lt;any&gt;&gt;</td>
<td>Class, Property</td>
<td>The stereotyped class or attribute will be replaced by an 'any' or 'anyAttribute' element. The tagged values are copied into the corresponding attributes of the generated element.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>namespace</td>
<td>As defined in XML Schema specification.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>processContents</td>
<td>As defined in XML Schema specification.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• values=&quot;skip</td>
<td>lax</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• default=&quot;strict&quot;</td>
<td></td>
</tr>
<tr>
<td>&lt;&lt;attribute&gt;&gt;</td>
<td>Property</td>
<td>Assigned to UML attribute or association end. Indicates item is to be generated as an attribute within complexType and not as an element.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>default</td>
<td>As defined in XML Schema specification.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>fixed</td>
<td>As defined in XML Schema specification.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>form, use</td>
<td>Overrides the attributeFormDefault for this schema.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• values=&quot;qualified</td>
<td>unqualified&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• default=&quot;optional&quot;</td>
<td></td>
</tr>
<tr>
<td>&lt;&lt;choice&gt;&gt;</td>
<td>Class</td>
<td>Elements marked with this stereotype represent a Choice model group contained within a complexType definition.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>memberNames</td>
<td>Overrides the package-level default for naming complexType definitions.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• values=&quot;qualified</td>
<td>unqualified&quot;</td>
</tr>
</tbody>
</table>

### UML profile for XSD (2)

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>UML construct</th>
<th>Tagged value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;&lt;element&gt;&gt;</td>
<td>Property</td>
<td>Determined whether this element may contain mixed element and character content.</td>
<td></td>
</tr>
<tr>
<td>mixed</td>
<td></td>
<td>• values=&quot;true</td>
<td>false&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• default=&quot;false&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>modelGroup</td>
<td>Overrides the package-level default model group.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• values=&quot;all</td>
<td>sequence</td>
</tr>
<tr>
<td></td>
<td>anonymousRole</td>
<td>The class type will be directly embedded within the complexType definition. Omit attribute or role type wrapper.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• values=&quot;true</td>
<td>false&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• default=&quot;false&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>anonymousType</td>
<td>The class type will be anonymous for XML documents generated by the schema.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• values=&quot;true</td>
<td>false&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• default=&quot;false&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>form</td>
<td>Overrides the elementFormDefault for this schema.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• values=&quot;qualified</td>
<td>unqualified&quot;</td>
</tr>
<tr>
<td></td>
<td>position</td>
<td>If assigned, indicates position in the sequence model group.</td>
<td></td>
</tr>
</tbody>
</table>
|           |                | A facet is a single defining aspect of a value space. Generally speaking, each facet characterizes a value space along independent...
References

http://www.mm.informatik.tu-darmstadt.de/staff/kuehne/publications/papers/mdlafoundation.pdf

http://albini.xactium.com/web/index.php?option=com_repositor&Itemid=54&func=select&id=1
