Using Model Driven Architecture™
to Manage Metadata

An Architectural Overview of the Meta Object Facility (MOF™)

David Frankel Consulting
df@DavidFrankelConsulting.com
www.DavidFrankelConsulting.com

Adapted from the book Model Driven Architecture: Applying MDA™ to Enterprise Computing, by David Frankel, published by John Wiley & Sons OMG Press

© 2003 David Frankel Consulting
Agenda

- Metadata Fragmentation
- Introduction to MOF
- MOF Isn’t Just for OO Languages
- Abstract Syntax Trees
- Model Driven Metadata Management
- Drill Downs
  - MOF-CORBA Mapping (CMI)
  - MOF-XML Mapping (XMI)
  - MOF-Java Mapping (JMI)
  - Metalevels
  - Using UML Tools to Create MOF Metamodels
- Additional Topics
What is Metadata?

• Originally, metadata meant only “data about data”
  – Database schema are distinct from the data itself

• Metadata now includes
  – UML models
  – Data transformation rules
  – APIs expressed in IDL, MIDL, C#, Java, WSDL, etc.
  – Business process and workflow models
  – Product configuration descriptors and tuning parameters
Metadata Fragmentation

• Example: One enterprise component may have several disparate forms of metadata
  – Platform-independent UML
  – Java interfaces
  – XML descriptors
  – CORBA IDL
  – Object-relational mapping
Ad-Hoc Metadata Integration

- UML
- Java
- IDL
- XML
- Object-Relational Mappings
- Relational Database Schema
- Multi-dimensional Database Schema
## Reflection and Metadata Fragmentation

<table>
<thead>
<tr>
<th></th>
<th>Operation that client invokes</th>
<th>Metadata that the operation returns</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CORBA</strong></td>
<td><code>get_interface</code></td>
<td><code>InterfaceDef</code></td>
</tr>
<tr>
<td><strong>COM</strong></td>
<td><code>GetTypeInfo</code></td>
<td><code>ITypeInfo</code></td>
</tr>
<tr>
<td><strong>Java</strong></td>
<td><code>getDeclaredMethods</code></td>
<td><code>Method</code></td>
</tr>
</tbody>
</table>
Volume and Value

• Volume is large
  – Global 1000 companies have tens of thousands of columns in their data models
  – New kinds of models coming on line

• Value is increasing
  – Metadata drives generators and dynamic execution engines
    • Has been true for some time (e.g. workflow, CORBA, COM) but MDA accelerates trend

• *Increasing amounts of increasingly valuable metadata*
Previous Metadata Integration Attempts

• Late 1980s and early 1990s
• Diagnosis correct: Metadata disparity
• Prescription: Have one grand metamodel
  – One kind of model
• Reason for failure: Different stakeholders have different viewpoints
  – And require different modeling constructs
Agenda

- Metadata Fragmentation
- Introduction to MOF
- MOF Isn’t Just for OO Languages
- Abstract Syntax Trees
- Model Driven Metadata Management
- Drill Downs
  - MOF-CORBA Mapping (CMI)
  - MOF-XML Mapping (XMI)
  - MOF-Java Mapping (JMI)
  - Metalevels
  - Using UML Tools to Create MOF Metamodels
- Additional Topics
MOF Overview

- OMG standard
  - Sister to UML

- The most fundamental MDA language definition mechanism
  - Even UML is defined via MOF
  - UML profiles are the other mechanism

- Supports model-driven metadata management
The Basic Premises

• There will be more than one modeling language
  – For different system aspects and levels of abstraction

• Different languages have different modeling constructs
  – For relational data modeling: table, column, key, etc.
  – For workflow modeling: activity, performer, split, join, etc.
  – For OO class modeling: class, attribute, operation, association, etc.

• A modest degree of commonality is achievable by using one language to define the different languages
  – For example, use same means to describe that…
    • a table owns its columns
    • a class owns its attributes and operations
    • a state machine its transitions
Metamodels

• A metamodel defines a language
• A MOF-compliant metamodel consists of
  – Abstract syntax
    • Expressed formally via MOF metamodeling constructs
  – Semantics
    • Defines the meaning of the abstract syntax
    • Expressed informally (today) via natural language (i.e. English)
Platform Independence

• MOF metamodels are platform independent, meaning independent of...
  – Information formatting technologies such as XML DTD and XML Schema
  – 3GLs and 4GLs such as Java, C++, C#, and Visual Basic
  – Distributed component middleware, such as J2EE, CORBA, and .NET
  – Messaging middleware such as WebSphere MQ Integrator (MQSeries) and MSMQ

• MOF technology mappings
  – XMI: MOF-XML mapping
  – JMI: MOF-Java mapping
  – CMI: MOF-CORBA mapping
Borrowing from UML

- MOF uses UML class modeling constructs
  - Including Object Constraint Language (OCL)
- Uses these constructs as the common means for defining abstract syntax
Fragment of UML Metamodel for Class Modeling

Adapted, with permission, from OMG document formal/01-09-67
Fragment of the CWM Relational Data Metamodel

Adapted, with permission, from OMG document ptc/01-09-04
Fragment of the CORBA Component Metamodel

- **InterfaceDef**: (from Base IDL)
  - +supports 0..*
  - +provides 1
  - +uses 0..*

- **OperationDef**: (from Base IDL)
  - +uses 0..*

- **Provides Def**: +facet 0..*

- **Uses Def**: multiple : boolean
  - +receptacle 0..*

- **ComponentDef**: isBasic : boolean
  - +manages 1
  - +emits 0..*
  - +publishes 0..*
  - +consumes 0..*

- **Emits Def**: 0..*
- **Publishes Def**: 0..*
- **Consumes Def**: 0..*

- **Factory Def**: +factory 0..*
- **Finder Def**: +finder 0..*

- **Home Def**: +home 0..*

- **PrimaryKey Def**: +key 0..1
- **Value Def**: (from Base IDL)
  - +type 1

Adapted, with permission, from OMG document orbos/99-07-02
Fragment of the UML Metamodel for State Charts

Adapted, with permission, from OMG document formal/01-09-67
 Agenda

• Metadata Fragmentation
• Introduction to MOF
• MOF Isn’t Just for OO Languages
• Abstract Syntax Trees
• Model Driven Metadata Management
• Drill Downs
  – MOF-CORBA Mapping (CMI)
  – MOF-XML Mapping (XMI)
  – MOF-Java Mapping (JMI)
  – Metalevels
  – Using UML Tools to Create MOF Metamodels
• Additional Topics
MOF is Not Just for OO Languages

• MOF uses object-oriented modeling to define modeling constructs
• But the modeling constructs it defines need not be object-oriented
Using MOF Subclassing to Define a Metamodel

ModelElement

name : String

Table

+table

1

Column

+column

1..*
Using MOF to Define Subclassing in a Metamodel

**ModelElement**

- **name**: String

```
context Table inv:
  superclass.column->forAll
  (superClassColumn | self.column->includes (superClassColumn))
```

Diagram:

```
+subClass
0..*

+superClass
0..*

Table

+table
1

+column
1..*

Column
```
Agenda

- Metadata Fragmentation
- Introduction to MOF
- MOF Isn’t Just for OO Languages
- Abstract Syntax Trees
- Model Driven Metadata Management
- Drill Downs
  - MOF-CORBA Mapping (CMI)
  - MOF-XML Mapping (XMI)
  - MOF-Java Mapping (JMI)
  - Metalevels
  - Using UML Tools to Create MOF Metamodels
- Additional Topics
Abstract Syntax and Abstract Syntax Trees

Metamodel
  e.g. CWM Relational

Abstract Syntax

Model
  e.g. CWM Relational
  Data Model

Abstract Syntax Tree

A    B  Means B conforms to A
Abstract Syntax Tree for a Specific Relational Data Model

Example 1: CREATE TYPE Person_t AS (name varchar(20), birthyear integer)
CREATE TYPE Emp_t UNDER person_t AS (salary integer)
Agenda

• Metadata Fragmentation
• Introduction to MOF
• MOF Isn’t Just for OO Languages
• Abstract Syntax Trees
• Model Driven Metadata Management
• Drill Downs
  – MOF-CORBA Mapping (CMI)
  – MOF-XML Mapping (XMI)
  – MOF-Java Mapping (JMI)
  – Metalevels
  – Using UML Tools to Create MOF Metamodels
• Additional Topics
Applying the MOF-XML Technology Mapping to CWM

CWM Metamodel

Abstract Syntax

Mapping

Concrete XML-Based Syntax for CWM (XML DTD or Schema)
Applying the MOF-XML Technology Mapping to UML

- UML Metamodel
- Abstract Syntax
- Concrete XML-Based Syntax for UML (XML DTD or Schema)

Mapping
Enforcing Semantics

- Different metamodels defined using the same constructs
  - Such as composite aggregation, invariants, etc.
- Model-driven metadata management tools understand these constructs’ semantics
  - And can enforce them
Metadata Management Scenario
1—Integrated MOF Repository

- MOF CORBA Interfaces
- MOF Java Interfaces (JMI)
- MOF XML (XMI) Documents
- Import/Export

MOF Repository

- UML Models
- Data Models
- Workflow Models
- CCM CORBA Interfaces
- B2Bi Collaboration Descriptions
Metadata Management Scenario
2—Federated MOF Repositories

= MOF CORBA Interfaces
= MOF Java Interfaces (JMI)
= MOF XML (XMI) Documents
= Import/Export
Generic MOF Repository Clients Use the Reflective Interfaces

- CWM Metamodel
- UML Metamodel
- MOF CORBA
- MOF Java
- CWM Metamodel-Specific
- UML Metamodel-Specific
- UML Metamodel-Specific Java Interfaces
- CWM Metamodel-Specific Java Interfaces
- MOF CORBA Reflective Interfaces
- UML-Specific Clients
- MOF Java Reflective Interfaces
- Generic MOF Clients
- CWM-Specific Clients
- <<MOF-CORBA Mapping>>
- <<MOF-Java Mapping>>
- <<call>>
- <<call>>
- <<call>>
Agenda

• Metadata Fragmentation
• Introduction to MOF
• MOF Isn’t Just for OO Languages
• Abstract Syntax Trees
• Model Driven Metadata Management
• Drill Downs
  – MOF-CORBA Mapping (CMI)
  – MOF-XML Mapping (XMI)
  – MOF-Java Mapping (JMI)
  – Metalevels
  – Using UML Tools to Create MOF Metamodels
• Additional Topics
MOF is Not CORBA Based

MOF-IDL Mapping

• Transforms a MOF-compliant metamodel into CORBA IDL
• Also specifies semantics of the IDL the mapping produces
• MOF specification includes this mapping
  – Introduction downplays MOF’s independence from CORBA
Applying the MOF-IDL Mapping Rules to the UML Metamodel

interface Classifier : ClassifierClass, GeneralizableElement, Core::Namespace
{
    FeatureUList feature ();
    void set_feature (in FeatureUList new_value)
    void unset_feature ();
    void add_feature (in Core::Feature new_element);
    void add_feature_before (in Core::Feature new_element,
                             in Core::Feature before_element);
    void modify_feature (in Core::Feature old_element,
                         in Core::Feature new_element);
    void remove_feature (in Core::Feature old_element);
    ...
};
Agenda

- Metadata Fragmentation
- Introduction to MOF
- MOF Isn’t Just for OO Languages
- Abstract Syntax Trees
- Model Driven Metadata Management
- Drill Downs
  - MOF-CORBA Mapping (CMI)
  - MOF-XML Mapping (XMI)
  - MOF-Java Mapping (JMI)
  - Metalevels
  - Using UML Tools to Create MOF Metamodels
- Additional Topics
A Closer Look at XMI

• Return on investment
  – MOF was invented before XML was popular
  – Platform-independence paid off

• XML and XML Schema
  – XMI 2.0 maps MOF to XML schema

• A common misconception about XMI
  – A MOF-XML mapping, not a single DTD for UML models

• XMI Complexity vs. UML complexity
  – Complexity of UML XMI DTD is due to complexity of the UML metamodel
Applying XMI’s MOF-XML Mapping Rules to the UML Metamodel

UML Metamodel

XMI’s MOF-XML Mapping Rules

XML DTD

...<!ELEMENT UML:Classifier.feature (  
  UML:Feature|  
  UML:StructuralFeature|  
  UML:Attribute|  
  UML:BehavioralFeature|  
  UML:Operation|  
  UML:Method|  
  UML:Reception)*>
...

...<!ENTITY % UML:ClassifierFeatures
  '%UML:GeneralizableElementFeatures; |  
  UML:Namespace.ownedElement |  
  UML:Classifier.feature |  
  ...
'">
...

...<!ELEMENT UML:Classifier (%UML:ClassifierFeatures;)*>
...
Preprocessing a Proprietary Representation of a UML Model

Proprietary Representation of a UML Model

Preprocessor

XMI Representation of the UML Model

Main Generator

= Flow
More Complete Separation of Concerns in a Generator

- Proprietary Representation of a UML Model
- Preprocessor
- XMI Representation of the UML Model
- XMI Parser
- Abstract Syntax Tree
- Main Generator

\[ \text{Flow} \]
Agenda

• Metadata Fragmentation
• Introduction to MOF
• MOF Isn’t Just for OO Languages
• Abstract Syntax Trees
• Model Driven Metadata Management
• Drill Downs
  – MOF-CORBA Mapping (CMI)
  – MOF-XML Mapping (XMI)
  – MOF-Java Mapping (JMI)
  – Metalevels
  – Using UML Tools to Create MOF Metamodels
• Additional Topics
A Closer Look at JMI

- Analogous to MOF-CORBA (IDL) mapping
  - For representing MOF metadata as Java objects
- Specified via Sun JSR #40
- Specifies syntax and semantics of generated interfaces
Aren’t XML and JDOM Enough?
Part 1—The MOF Way

• MOF repository enforces
  – Ownership by a table of its columns
  – Rule against table and column having same name
• Generated metadata management code has operations that match the metamodel
  – getName: Returns the name
  – setName: Sets the name
  – getColumn: Returns a Java List of the columns. Uses List because of the
    {ordered} specification in the model. The List can also be used to add,
    modify, and remove columns.
  – refDelete: Deletes the table
Aren’t XML and JDOM Enough?
Part 2—XML Without MOF, Plus JDOM

- Semantically thin
- Does not express ownership by a table of its columns
  - Does not enforce it
- Does not express that column can’t have same name as table
  - Does not enforce it
- Programmers have to write extra code on top of JDOM to enforce these semantics
Agenda

• Metadata Fragmentation
• Introduction to MOF
• MOF Isn’t Just for OO Languages
• Abstract Syntax Trees
• Model Driven Metadata Management
• Drill Downs
  – MOF-CORBA Mapping (CMI)
  – MOF-XML Mapping (XMI)
  – MOF-Java Mapping (JMI)
  – Metalevels
  – Using UML Tools to Create MOF Metamodels
• Additional Topics
## Four Metalevels

<table>
<thead>
<tr>
<th>Description</th>
<th>Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>M3</strong></td>
<td>MOF, i.e. the set of constructs used to define metamodels</td>
</tr>
<tr>
<td><strong>M2</strong></td>
<td>Metamodels, consisting of instances of MOF constructs.</td>
</tr>
<tr>
<td><strong>M1</strong></td>
<td>Models, consisting of instances of M2 metamodel constructs.</td>
</tr>
<tr>
<td><strong>M0</strong></td>
<td>Objects and data, i.e. instances of M1 model constructs</td>
</tr>
</tbody>
</table>
M2 Metamodel Constructs as Instances of M3 Constructs

Instance of MOF Class

Instance of MOF Attribute

ModelElement
  name : String

Instance of MOF Generalization

Table
  +table
  1

Column
  +column
  1..*

Instance of MOF Class

Instance of MOF Association (with composite aggregation)
M1 Data Model Elements as Instances of M2 Data Metamodel Elements

Employee : Table

Address : Column

Number : Column

Name : Column
Level M0

- Employee number A3949 named “Susan Smith” with address “111 Main St. USA.”
  - Instance of Employee (M1)
    - which is an instance of Table (M2)
      - which is an instance of MOF Class (M3).
  - A3949 is an instance of Number (M1)
    - which is an instance of Column (M2)
      - which is an instance of MOF Class (M3)
Does Meta Matter?

• For some purposes, absolute metalevels are arbitrary
  – Only the relative metalevel matters
• For some purposes, absolute metalevels are convenient for discourse
• Special concerns with M1 models
  – Use of deferred constraints
    • Ok for metamodels (M2 models)
    • More problematical for M1 models—”dirty data”
  – Need for lower level models
    • M1 models are an order of magnitude more complex
    • Artifacts generated from M1 PIMs can be hard to manage without platform-specific models (PSMs)
Self-Description
A Fragment of "the" MOF Model's Abstract Syntax

```
Class
    isSingleton : Boolean

Association
    isDerived : Boolean

DataType

Classifier
    +type : 1..1

TypedElement
    0..*
```

Diagram:
- Classifier: +type (1..1) to TypedElement (0..*)
- Class: isSingleton (Boolean)
- Association: isDerived (Boolean)
- DataType
M3, M2, and Abstract Syntax

MOF (M3)  
Abstract Syntax ↔ Metamodel (M2)  
Abstract Syntax Tree

A ← B Means B conforms to A
M3, Self-Description, and Abstract Syntax

MOF (M3) ← Abstract Syntax

MOF (M3) ← Abstract Syntax Tree

A ← B Means B conforms to A
Two XMI Artifacts Per Metamodel

- When a MOF generator transforms a metamodel it can produce *two* kinds of XMI artifacts
  - An XMI document that contains *all* the properties of *all* of the elements of the metamodel, which are M2 elements. This document validates against “the” MOF DTD.
  - A DTD (or Schema) for representing M1 instances of the metamodel’s M2 elements.
Generating Two XMI Artifacts

- XMI document that validates against “The” MOF DTD
  - Includes the invariant rule (instance of MOF::Constraint)
  - “The MOF DTD” encodes instances of the MOF constructs

- XMI DTD (or schema) for encoding instances of Tables and Columns
  - Does not include the invariant rule
  - Reverse engineering does not yield the full metamodel
Fragment of “The” MOF DTD

<!ELEMENT Model:Class (Model:ModelElement.name|
Model:ModelElement.annotation|
Model:ModelElement.container|
Model:ModelElement.constraints|
Model:Namespace.contents|
Model:GeneralizableElement.supertypes|
XMI.extension)*>

<!ATTLIST Model:Class
name CDATA #IMPLIED
annotation CDATA #IMPLIED
isRoot (true|false) #REQUIRED
isLeaf (true|false) #REQUIRED
isAbstract (true|false) #REQUIRED
visibility (public_vis|protected_vis|private_vis) #REQUIRED
isSingleton (true|false) #REQUIRED
container IDREFS #IMPLIED
constraints IDREFS #IMPLIED
contents IDREFS #IMPLIED
supertypes IDREFS #IMPLIED
%XMI.element.att; %XMI.link.att;>
Applying Standard MOF Mappings to "The" MOF Model

MOF-Compliant Metamodel

MOF-CORBA Mapping

MOF-Compliant CORBA APIs

For M1 elements

MOF-Compliant Java APIs

XMI

DTD or Schema

M3 elements

“The” MOF Model (MOF-Compliant)

MOF-CORBA Mapping

MOF-Java Mapping (JMI)

MOF-XML Mapping (XMI)

For M2 elements

MOF-Compliant CORBA APIs

MOF-Compliant Java APIs

XMI

DTD or Schema
A MOF Repository Manages M2 and M3 Similarly to M1

- **MOF Repository**
  - UML Models (M1)
  - Data Models (M1)
  - Meta Models (M2)
  - “The” MOF Model (M3)

Icons:
- **= MOF CORBA Interfaces**
- **= MOF Java Interfaces (JMI)**
- **= MOF XML (XMI) Documents**
- **= Import/Export**
Agenda

• Metadata Fragmentation
• Introduction to MOF
• MOF Isn’t Just for OO Languages
• Abstract Syntax Trees
• Model Driven Metadata Management
• Drill Downs
  – MOF-CORBA Mapping (CMI)
  – MOF-XML Mapping (XMI)
  – MOF-Java Mapping (JMI)
  – Metalevels
  – Using UML Tools to Create MOF Metamodels
• Additional Topics
UML Modeling vs. MOF Metamodeling

• MOF metamodeling is similar to UML class modeling

• Can use UML tools to create MOF metamodel by following some basic rules
  – Don’t use association classes
  – Don’t use qualifiers
  – Don’t use n-ary associations
  – Don’t use dependencies

• UML for Profile for MOF specifies full rules for using UML to define MOF metamodels
Decomposing Association Classes

Association Class

Two Simple Associations

Decomposes to

Subscription
RenewalDate : Date

NewsService
Person

NewsService
Person

Subscription
RenewalDate : Date
Decomposing an N-Ary Association

N-Ary Association *Negotiation*, Associating Three Classes

Binary Associations Only
Agenda

• Metadata Fragmentation
• Introduction to MOF
• MOF Isn’t Just for OO Languages
• Abstract Syntax Trees
• Model Driven Metadata Management
• Drill Downs
  – MOF-CORBA Mapping (CMI)
  – MOF-XML Mapping (XMI)
  – MOF-Java Mapping (JMI)
  – Metalevels
  – Using UML Tools to Create MOF Metamodels
• Additional Topics
Class ApartmentBuilding extends Building
{
    attribute address String;
    ...
}

Class Apartment
{
    ...
}

Association Building_Apartment
{
    Association End aptBuilding type ApartmentBuilding [aggregation_composite] 1..1
    Association End apt           type Apartment           [isOrdered,isNavigable] 1..*
}

...
Using the XML-MOF Reverse Mapping

Non-XMI XML DTD or Schema

XML-MOF Reverse Mapping

MOF-Compliant Metamodel

MOF-CORBA Mapping

MOF-Java Mapping (JMI)

MOF-XML Mapping (XMI)

MOF-Compliant CORBA APIs

MOF-Compliant Java APIs

XMI DTD or Schema
MOF In the Computer Industry

• Enterprise software tooling
  – IBM WebSphere, Eclipse
  – Oracle Data Warehousing
  – Unisys internal application development tools
  – Sun’s MOF-oriented JSRs
  – Third-party tools

• MOF and W3C’s Resource Description Framework (RDF)
  – MOF used for enterprise software tooling
    • But not theoretically limited to it
  – RDF used for describing Web content
    • But not theoretically limited to it
  – Integration possibilities
    • Define a MOF metamodel of RDF
      – Would allow RDF to leverage the MOF technology mappings
    • Define a MOF-RDF mapping
      – Would allow MOF metadata to be expressed as RDF metadata
MOF Weaknesses and Future Directions

• Weaknesses in MOF 1.x
  1. Lack of coverage of graphical notation
  2. Lack of support for versioning
  3. Misalignment with UML
  4. MOF-CORBA mapping problems
  5. Interoperability problems due to immaturity

• Future directions
  – Interoperability testing addresses #5
  – MOF 2.0 #2, 3, 4
Summary

• Basic MOF assumptions
  – Multiple modeling languages
  – Common means of describing the various languages allows metadata integration

• A MOF metamodel is platform independent, and consists of
  – Abstract syntax, in the form of a class model
  – Informal textual descriptions

• Standardized MOF mappings
  – MOF-CORBA (CMI)
  – MOF-XML (XMI)
  – MOF-Java (JMI)

• Class models defining abstract syntax drive model-driven metadata management tools
Need More?