Introduction to OMG’s Model Driven Architecture

September 2001

Written and Presented by
Jon Siegel, Ph.D.
Vice President, Technology Transfer
Object Management Group
siegel@omg.org
781-444-0404
From Design to Deployment

Modeling Environment:
- Data Model
- Appl Model
- Repository

Applications
- Domain Frameworks:
  - Mfg: PDM
  - Fin: Acctg

Distributed Infrastructure:
- Platforms, Languages, Networks, Protocols, Middlewares, Messaging

Support for All your Business Computing
From Design to Deployment

Modeling Environment:

- Data Model
- Appl Model
- Repository

Support for All your Business Computing
OMG Modeling Support

- **MOF: Meta-Object Facility**
  - Integrated Repository
  - Standard MetaModel
- **Unified Modeling Language UML 1.3**
  - World Standard for A&D
  - Representation for Structure, Dynamics, Deployment
- **XMI: XML Metadata Interchange**
  - Model & MetaModel Interchange
  - XML-Based Format, including DTDs
- **CWM: Common Warehouse Metamodel**
  - Data Warehousing Integration
  - Record, Table formats; Data Loading & Transformation
OMG Metadata before MOF

• An Example: Three places to store Metadata about Objects in your System:
  – Naming Service
  – Trader Service
  – Interface Repository

• But no explicit Metadata Architecture

• MOF defines modeling primitives
  – MOF::Class (MetaClass)
  – MOF::Attribute (MetaAttribute)
4-Layer Metamodel

- Describes Modeling Concepts in a Domain
- Arrow denotes “instance-of” dependency

User Objects Layer (M0)

Model Layer (M1)

Metamodel Layer (M2)

Meta-MetaModel Layer (M3)

<PG Shares> 56 3/8 [User Data]

StockShare AskPrice [Domain Model, CBO, …]

UML::Class Attribute, Operation [UML, CWM, …]

MOF::Class MOF::Attribute [MOF]
What is the MOF?

- The MOF defines an abstract model called a meta-metamodel
- The MOF specification defines a standard distributed repository:
  - That is, a set of modeling constructs and IDL interfaces to define and manipulate a set of interoperable metamodels
- With UML and XMI, an integral part of a complete suite of modeling tools
The MOF Defines...

- **CLASSES**, with Attributes and Operations at both Object and Class level
  - Attributes represent Metadata
  - Operations represent functions on metadata

- **ASSOCIATIONS** support binary links between class instances
  - AssociationEnds specify Ordering or Aggregation semantics, Cardinality

- **PACKAGES** are collections of related Classes and Associations

- **DATATYPES** represent non-object types as Parameters or Attributes

- **CONSTRAINTS** associate semantic restrictions with other elements
What is this good for?

- Every development environment is built on a meta-model:
  - Languages like C++, Java, Smalltalk, etc.
  - Environments like CORBA, COM, CICS, etc.
- You need to consider this when you pick a modeling tool
  - Specialized tools have only limited use
  - Generalized tools may not constrain to models implementable in your development environment
- You may already have, or need, multiple modeling tools
  - Use XMI to transfer models among them, mapping from one meta-model to another
  - Store your models in the standard MOF repository regardless of their tool of origin, or meta-model
- Interfaces for **reflective** and **introspective** functions let objects or applications examine their meta-data
  - Take advantage of modeling to design and implement more flexible, powerful applications
XMI: XML Metadata Interchange

- XMI Integrates 3 key industry standards:
  - XML; MOF; UML
- The XMI Specification consists of:
  - Set of XML DTD/Schema production rules for transforming MOF-based metamodels into XML DTDs/Schemas
  - Set of XML document production rules for encoding and decoding MOF-based metadata
  - Design principles for generating XMI-compliant DTDs & Schemas
  - Design principles for generating XMI-compliant XML documents
  - Concrete XML DTDs & Schemas for MOF (to exchange metamodels) and UML (to exchange models)
XMI: a Pair of Parallel Mappings

- Between MOF Metamodels and XML DTDs or Schemas, and
- MOF Metadata and XML Documents (next slide)
- Also applies One Level Down
- Automatically Generate Transfer Syntax
  - Useful as Metamodel Changes, e.g. to Accommodate New UML Profiles
- But the DTD is *Not* a Complete Representation of the Metamodel
  - Need the Metamodel to Reconstruct at Receiving End
  - XML Schema Representation is More Complete
Interchanging Metadata

- Data (M0)
- Metadata, Model (M1)
- Meta-Metadata, Metamodel (M2)
- Meta-MetaModel Layer (M3)
- Modeled System, Warehouse DB
- UML Model, Warehouse Schema
- UML or Profile Metamodel, CWM Metamodel
- The MOF Model

- UML Model Document
- MOF Metamodel Document
- UML DTD or Schema
- MOF DTD or Schema

Copyright © 1998-2001 Object Management Group
Big Software Projects...

• are like Buildings – they have a structure with many interlocking parts

• You wouldn’t contract to build a skyscraper without seeing plans first:
  – Elevations  – Blueprints
  – Interior Views  – Floor Plans
  – Site Plan  – Structural Analyses

• Large Software Projects deserve the same treatment

• Better Time and Cost Estimates; Less Risk
UML – a *Graphic* Language for

- **Visualizing**
  - Using the standardized graphic UML displays

- **Specifying**
  - Semantics to define
    - static structure
    - dynamic behavior
    - model organization

- **Constructing**
  - Map UML to Programming Environment and Generate some code Automatically

- **Documenting**
  - Every phase of lifecycle from analysis and design through deployment and maintenance
The UML Specification defines

- **UML Semantics**
  - Defined using a metamodel

- **UML Notation Guide**
  - Defines a graphic syntax for UML semantics

- **UML Standard Profiles**
  - Extensions for SW development and business modeling

- **UML CORBAfacility Definition**
  - A standard repository for UML models
  - Supports XMI

- **Object Constraint Language**
  - A standardized constraint language
UML Building Blocks

• Basic Building Blocks:
  – Model Elements (classes, interfaces, components, use cases, etc.)
  – Relationships (associations, generalizations, dependencies, etc.)
  – Diagrams (class diagrams, use case diagrams, interaction diagrams, etc.)

• These basic building blocks are used to create large, complex structures
Well-Formedness Rules

• Well-formed: A model or model fragment that adheres to all semantic and syntactic rules that apply to it

• UML specifies rules for:
  – Naming
  – Scoping
  – Visibility
  – Integrity
  – Execution (limited)

• But, during iterative development, you will be able to work with incomplete and inconsistent models – that is, models that are not well-formed
Class Element

- A block denotes a **class**
- A class has
  - Attributes – characterizing objects of the class
  - Operations – to manipulate the attributes, or perform other actions
- Classes may be *associated* in various ways:
  - Associations
  - Generalizations

### Store

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Totals</strong>: long</td>
<td><strong>POSlist</strong>: List</td>
</tr>
<tr>
<td><strong>Login()</strong></td>
<td><strong>getPOStotals(Totals:long)</strong></td>
</tr>
<tr>
<td><strong>updateStore(Totals:long)</strong></td>
<td></td>
</tr>
</tbody>
</table>
Interface & Component Elements

• **Interface**: Named set of operations characterizing the behavior of an element

• **Component**: Replaceable part of a system that packages implementation, and provides the realization of a set of interfaces
### Core Relationships

<table>
<thead>
<tr>
<th>Construct</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Association</td>
<td>Relationship between two or more classifiers involving connections among their instances</td>
<td></td>
</tr>
<tr>
<td>Aggregation</td>
<td>Association which specifies a whole-part relationship between the aggregate (whole) and the part</td>
<td></td>
</tr>
<tr>
<td>Generalization</td>
<td>Relationship between a more general and more specific element</td>
<td></td>
</tr>
<tr>
<td>Dependency</td>
<td>Relationship where a change to one (&quot;independent&quot;) modeling element affects the other (&quot;dependent&quot;) element</td>
<td></td>
</tr>
<tr>
<td>Realization</td>
<td>Relationship between a specification and its implementation</td>
<td></td>
</tr>
</tbody>
</table>
UML Diagrams

• **Foundation: Structural Diagrams** – static structure
  – Class Diagram
  – Object Diagram
  – Component Diagram
  – Deployment Diagram

• **Behavior: Behavioral Diagrams** – dynamic behavior
  – Use Case Diagram
  – Sequence Diagram
  – Collaboration Diagram
  – Statechart Diagram
  – Activity Diagram

• **Model Management Diagrams** – organization
  – Packages
  – Subsystems
  – Models
Types & Implementation Classes

Fig. 3-27, UML Notation Guide

```
«type»
Object

* elements

«type»
Set
addElement(Object)
removeElement(Object)
testElement(Object):Boolean

«implementationClass»
HashTable

1 body

«implementationClass»
HashTableSet

addElement(Object)
removeElement(Object)
testElement(Object):Boolean
setTableSize(Integer)
```
Static Structural Diagrams

• Two Kinds:
  – *Class Diagram* (Classifier View)
  – *Object Diagram* (Instance View)

• Shows a Graph of Classifier Elements Connected by Static Relationships
Class Diagram - Fragment

```
+initialization()
+calculateTax()
+findTaxablePrice()
-rate : float
```

```
Tax
```

```
«CORBAInterface»
ITax
```
Behavioral Diagrams

- Sequence Diagram
- Use Case Diagram
Parts of a Sequence Diagram

• A Sequence Diagram Maps to either
  – An Interaction and an underlying Collaboration, or
  – An InteractionInstanceSet and an underlying CollaborationInstanceSet

• Vertical dimension: Time “Lifelines”
  – Dotted when Inactive; Slim Rectangle when Active

• Horizontal dimension: Instances (arbitrary order)
  – Vertical/Horizontal axes may be reversed

• Timing Constraints may be noted

• Arrow Formats:
  – Procedure Call
  – Asynchronous Call
  – Return (Synch or Asynch)
  (Synchronous Return Arrow may be Omitted)
Use Case Diagram

Customer

- Check Status
- Place Order
- Fill Orders
- Establish Credit

Telephone Catalog

Salesperson
Shipping Clerk
Supervisor
Use Cases Are Good For...

- Modeling **User Requirements**
- Modeling **Test Scenarios**
- If you’re using a use-case driven method –
  - Start with use cases and derive your structural and behavioral models from it
- If not –
  - Make sure that your use cases are consistent with your structural and behavioral models
  - Thanks to Cris Kobryn for these guidelines
UML Summary

• The Way the World Does Modeling, with Universal Industry Support
• Flexible Representation of Static Structure and Dynamic Behavior
• Diagrams Map to Formally Defined Underlying Model
• Usable by Every Methodology
• An OMG Standard
• Widely Supported Upgrade to UML 2.0 Now Underway
The Metadata Problem

CWM Addresses a Problem Facing Every Enterprise:

• Many Databases
• Many Repositories
• Many Schemas Describing the “Same” Data
• Moving Data Requires Manual Schema Transformation
CWM Integrates your Data

- Integrates Existing Data Models
- Maps to Existing Schemas
- Supports Automated Schema Generation
- Supports Automated Database Loading
- The Basis for Data Mining and OLAP Across the Enterprise
CWM Defines Metamodels for:

- CWM Foundation
- Relational Data
- Record Data
- Multidimensional Data
- XML Data
- Data Transformations
- OLAP
- Data Mining
- Info Visualization
- Business Nomenclature
- Warehouse Process
- Warehouse Operation
# The CWM Metamodel

<table>
<thead>
<tr>
<th>Warehouse Process</th>
<th>Warehouse Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Transformation</strong></td>
<td>Data Mining</td>
</tr>
<tr>
<td><strong>OO (UML)</strong></td>
<td>Information Visualizatn</td>
</tr>
<tr>
<td><strong>Relational</strong></td>
<td>Multidimensional</td>
</tr>
<tr>
<td><strong>Data Types</strong></td>
<td>XML</td>
</tr>
<tr>
<td><strong>Expression</strong></td>
<td>Keys, Indexes</td>
</tr>
<tr>
<td><strong>Keys, Indexes</strong></td>
<td>Type Mapping</td>
</tr>
<tr>
<td><strong>Type Mapping</strong></td>
<td>Software Deployment</td>
</tr>
</tbody>
</table>

UML 1.3 (Core, Common_Behavior, Model_Management)

**Management**

**Analysis**

**Resource**

**Foundation**
New “Next Best Thing”: Web Services

Clipped from the ebXML Technical Architecture
What is the Model Driven Architecture™?

• A New Way to Specify and Build Systems
  – Based on Modeling and UML
  – Supports full lifecycle: A&D, implementation, deployment, maintenance, and evolution
  – Builds in Interoperability and Portability
  – Lowers initial cost and maximizes ROI
  – Applies directly to the mix of hardware and software that you face:
    • Programming language
    • Operating system
    • Network
    • Middleware
Waves of Middleware Platforms

• CORBA®: Vendor, OS- Independent Middleware
• But not the only MW. For example:
  – COM/DCOM/MTS
  – Java/EJB
  – XML/SOAP
  – C#/.Net
  – What will be Next Best Thing?
• Need to preserve value of Software Investment as the infrastructure landscape changes around it
• Need Portability and Interoperability across HW & SW vendor, operating system, programming language, network, and now middleware too!
Building an MDA™ Application

Start with a **Platform-Independent Model** (PIM), in UML and defined at multiple levels.

Base level PIM represents *only* business functionality and behavior, undistorted by technology details.

Next level adds, e.g., general aspects of components or asynch comms.

A Detailed Model, stating Pre- and Post-Conditions in OCL, and Semantics in Action Language
MDA tool applies an OMG™-standard Mapping – formally, a UML Profile – to generate a Platform-Specific Model (PSM) from the PIM.

This model, like the PIM, will be very detailed.

This step may require hand-editing, depending on the tool and environment.
OMG will standardize – and MDA tools will implement – mappings to multiple middleware platforms.

Each mapping – formally, a UML profile – defines the route from an application’s single PIM to a PSM on a target platform.
A PSM contains basically the same information as an application, but expressed in UML instead of code.

MDA development tools can generate all or most of an application from a PSM: interfaces, templates, configuration files, more.
Pathway to an MDA Application
Targeting Multiple Platforms

Input to
Mapping Tool applies

Mapping to CCM Server Platform
Mapping to EJB Server Platform
Mapping to MTS/DCOM Server Platform
Mapping to Browser Client Platform
Mapping to Pager Client Platform

To Produce

CCM Model
EJB Model
MTS/DCOM Model
Browser Client Model
Pager Client Model

From this, Code Generator produces

CCM Artifacts
EJB Artifacts
MTS/DCOM Artifacts
Browser Client Artifacts
Pager Client Artifacts

Finally, Compile, Configure, Assembly yields

CCM Server
EJB Server
MTS/DCOM Server
Browser Client
Pager Client
MDA Tools will also generate cross-platform bridging code connecting either instances of a single MDA application, or one application to another.

Standard *Pervasive Services* – directory, security, more – will also be accessed through bridging code where necessary.

MDA Tools combine application and platform knowledge to generate bridges.
MDA in Industry Standards

OMG (and other) Task Forces standardize Domain (Industry-Specific) Facilities as PIMs.

With implementations on multiple platforms, no technology or platform barriers prevent widespread adoption and use.

Interoperate cross-platform with other standard applications.

Both PIM and set of PSMs and interface code – on every mapped platform – become OMG standards.
MDA Specifications

• MDA Architecture (vote underway)
• UML 1.4 (complete) and 2.0 (in process)
• UML Profiles:
  – Profile for EDOC (in process)
  – Profile for EAI (in process)
  – Profile for CORBA (complete)
  – Profile for EJB (in JCP)
• Support from XMI, CWM (complete)
• Pervasive Services (coming)
• Domain Specifications
MDA Benefits

- Full support throughout the application life-cycle
- Reduced costs from beginning to end
- Reduced development time for new applications
- Technology-independent representation of business rules
- Optimized technical behavior - scalability, robustness, security – via generated code
- Stable, model-based approach maximizes SW ROI
- Smooth integration across middleware platform boundaries
- Rapid inclusion of emerging technologies into existing systems
MDA Links

- MDA Central:  http://www.omg.org/MDA
- Presentations:  http://www.omg.org/MDA/presentations.htm
- Contact OMG:  Email info@omg.org or siegel@omg.org
Contact OMG:

• **Web:**
  – **Home page:** [http://www.omg.org](http://www.omg.org)
  – **Work in Progress:** [http://www.omg.org/schedule.htm](http://www.omg.org/schedule.htm)
  – **About OMG:** [http://www.omg.org/gettingstarted/gettingstartedindex.htm](http://www.omg.org/gettingstarted/gettingstartedindex.htm)

• **Email:** siegel@omg.org or info@omg.org

• **Telephone:** 781-444-0404