Metadata Integration using UML, MOF and XMI

Sridhar Iyengar (Chair of MOF RTF) and Steve Brodsky (Chair of XMI RTF)

November 2000
Tutorial Series

- Lecture 1: Introduction to UML: Structural Modeling and Use Cases
- Lecture 2: Behavioral Modeling with UML
- Lecture 3: Advanced Modeling with UML
- Lecture 4: Metadata Integration with UML, MOF and XMI
Tutorial Series Goals

What you will learn:
- What the UML is and what is it not
- UML’s basic constructs, rules and diagram techniques
- How the UML can model large, complex systems
- How the UML can specify systems in an implementation-independent manner
- How UML, XMI and MOF can facilitate metadata integration
  - Basic principles of Metamodeling

What you will not learn:
- Object Modeling
- Development Methods or Processes
- Metamodeling 202!
Integration using UML, MOF and XMI

Part 1: OMG Modeling and Metadata Architecture
- Sridhar Iyengar, Unisys Corporation

Part 2: OMG Meta Object Facility (MOF)
- Sridhar Iyengar, Unisys Corporation

Part 3: OMG XML Metadata Interchange (XMI)
- Stephen Brodsky, IBM

Part 4: Wrap up and Futures
- Sridhar Iyengar and Stephen Brodsky
Architect’s Integration Dilemma

ArchiteXt (XML!)

DataMarts
SQL/Server

Designer
2000
Oracle Apps

Unisys
LINC,
UREP

Middleware (Tuxedo, JDBC, DCOM, IIOP, CCM, RMI, EJB, SOAP, COM+, …)
Information Models (MOF, UML, CWM, OIM, RSM, BODs, PIPs, …)

Warehouse
NCR
Teradata

Siebel

IBM
VisualAge

Microsoft
Visual Basic

BEA
Weblogic

Distributed, Heterogeneous, Client/Server!

Multiple Clients, Servers, Tools, Apps, O/S, Databases, Repositories, Object Models
Software Architecture Evolution

- **1980**
  - Hierarchical, Monolithic
  - E/R Models
  - Proprietary Interchange
  - IBM, Burroughs

- **1990**
  - Data Administration
  - Y2000, Data Warehouse

- **2000**
  - Components
  - Frameworks
  - Business Objects

- **OO, O-R**
  - Relational
  - Distributed Components
  - Object Models (UML)
  - XML Interchange (XMI)
  - Unisys, IBM, Oracle, MS..

**Complexity**

**1980** to **2000**

**1990**

**UML, MOF & XMI**
Solving the Integration Problem

What is needed?

- Methodology for building, evolving and integrating software
- A mechanism for cataloging and indexing and searching enterprise assets (metadata and data, internal and external)
- A flexible service based distributed component architecture that spans the enterprise
- A set of shared standard vocabularies (information models) and notation
- A metadata driven approach to automate transformations between islands of information
Metadata Appears in many Environments

- Application development tools and frameworks
  - Models, Record definitions, Database definitions...

- Component based environments
  - Interfaces, Classes, Components, Deployment info, Models...

- Data warehouses and enterprise information portals
  - Tables, Columns, Schemas, Cubes, Dimensions...

- Systems management environments
  - MIBs, Performance statistics, Configuration info...

- Middleware
  - Formats, Protocols, Configuration information etc
OMG A&D PTF Premise

- The use of Modeling and Metadata enabled architectures allows us to manage the complexity of software development, application integration and data warehouse management.

- Modeling and Metadata standards are necessary for interchange of software artifacts and interoperability between tools, applications, middleware and data stores.
Metadata Basics

Meta data is “data about the data”, “model about models”, “definitions of computing and business assets” - and more

- Technical: Design and Middleware meta data
- Administrative and process meta data
- Business meta data

End users, Software designers and IT personnel need to know:

- What does the data mean? What is its structure and format?
- Where did it come from? How was it calculated?
- When was it loaded? Who owns it? Where is it used?
- Etc.

Most metadata undocumented and usually locked up causing serious interchange and interoperability problems
There is more & more metadata lurking around!

- **Name**: Char
- **Marital Status**: Enumeration (M, D, W, S)
- **Age**: Integer
- **LastName**: String
- **First Initial**: String
- **Employee Number**: Number

**Data**:
- **John**: M 43
- **Mary**: D 27
- **Bill**: W 62
- **George**: S 18

**REPOSITORY**
- **Person Table**
- **User::John Doe**
- **Created-by**
- **Meta Data**
- **Space**
- **Archive**
- **Last good Archive 9/28/1998 : 2 AM**
Why (Meta)Model?

- Understand and describe the problem domain
- Help others understand the problem domain by using the same language
- Define a vocabulary for the elements in the problem domain
- Manage complexity by raising the level of abstraction at which we think and design
- Civil, Mechanical and hardware engineers have been adept at this for much longer than us
- Metamodels are used in the domain of modeling middleware and type systems (UML, IDL, CWM...).
Why Metadata enabled Systems?

- Manage complexity of systems and applications
- Dynamic development tools and applications need component and legacy meta data (MOF, XML)
  - Component assembly, work flow driven applications, wrappers
- Data warehouse environments are becoming more dynamic and mission critical!
  - The days of weekly and daily batch updates are being replaced by more frequent synchronization
- Distributed systems evolve independently
  - Need meta data for logical centralization and for reconciling discrepancies as systems evolve
- Lower cost of integration and evolution
The “meta” prefix
- Indicates one level of abstraction higher than root
- Is used in a relative manner

A metamodel is a model of a model

The concept can be recursively applied to itself
- A meta-metamodel is a model of a metamodel
- A meta-meta-metamodel is a model of a meta-metamodel … and so on

Typically 4 layers (M0-M3) are considered adequate

Tools now assist ‘Visual metamodelling’

OMG is now using M0-M3 notation for modeling

Metamodels are usually defined for specific technology domains
OMG - Modeling and Metadata Standards

- OA&D, Metadata Repository RFI 1994-1995
- Distributed Metadata Management: Meta Object Facility (MOF) 1997
- Stream based Model Interchange Format: XML Metadata Interchange (XMI) 1999
- CORBA Component Model, IDL Metamodel: 1999
- Data Warehouse Management: Common Warehouse Metamodel (CWM) 2000
- UML profile for CORBA 2000

*In development
OMG Metamodel Architecture

Defined By

OMG Terms

Sample Objects

XML/Java Terms

APPLICATION

User Objects Layer (M0)

<Acme_Software_Share 98789>, 654.56, sell_limit_order, <Stock_Quote_Svr 32123>

User Objects
XML Docs/Java Objects
Business data

Model Layer (M1)

StockShare, askPrice, sellLimitOrder, StockQuoteServer
[Models, Interfaces, BODs]

Business Models, DTDs, Java Classes
App Schemas

Metamodel Layer (M2)

UML::Class, Attribute,
CWM::Table, CWM::Cube
[UML, CWM, EAI, IDL…]

CWM, EAI, IDL,
UML, DTD/Schema
Middleware Schema

Meta-Metamodel Layer (M3)

MOF::Class, MOF::
Attribute, MOF::Operation
[MOF]

MOF DTD/Schema
Schema of Schema!!

UML, MOF & XMI

16
OMG Metamodel Architecture

M3 Layer: Specifies meta-metaclasses for the UML metamodel

M2 Layer: Specifies metaclasses for the UML metamodel, such as Class

M1 Layer: Specifies classes for the UML user models, such as Passenger, Ticket, TravelAgency

M0 Layer: User objects that are instances of UML user model classes, such as instances of Passenger, Ticket, TravelAgency
OMG OMA & Metadata Architecture

Repositories
Tools
Applications

Modeling and Metadata Facility
MOF
XMI
UML, CWM

Object Request Broker

Object Services (Naming, Transactions…)

Iyengar (c) 1995, Unisys
Modeling and Metadata Environment

<<Subsystem>>
UML Modeling Tools

<<Subsystem>>
Java/C++ IDE's

<<Subsystem>>
Component Assembly Tools

<<Subsystem>>
Component Management Tools

<<OMG Metadata>>
XMI

<<Subsystem>>
UML Repositories

<<Subsystem>>
CWM Repositories

<<Subsystem>>
MOF Repositories

<<Subsystem>>
Component Repositories

CORBA ORB CORE, IIOP Engine, Java/IDL, COM/CORBA...
(from OMG Object Management Architecture (MOF/UML/XMI))

OMG Metadata Architecture showing use of CORBA or XMI
(c) 1999, Unisys (Iyengar99)
OMG Modeling and Metadata Framework

- **UML**
  - Model & Design

- **XMI**
  - MOF2XML DTD
  - MOF2XML Doc
  - MOF2XML Schema*

- **IDL**
  - MOF2IDL

- **JMI**
  - MOF2Java

* Coming
JSR-40 JMI : Java Metadata Interface

- Vertical Industry Specifications
- Data Warehousing
- B2B Application Integration
- Model Driven App Development
- MetaData Management
- And So On
Software mess to Architected Elegance

Tijuana “shantytown”:
http://www.macalester.edu/~jschatz/residential.html

Fallingwater:
http://www.adelaide.net.au/~jpolias/FLW/Images/FallingWater.jpeg
OMG E-Business Integration Vision
The Big Picture of how it all ties together

Community & Enterprise Information Portals (KM...)

- HealthCare
- Financial
- Manufacturing
- Insurance...

E-Business Application Development
UML, CCM

E-Business Application Integration
CCM, EJB, UML4EAI

E-Business Intelligence
CWM

Information Models, Components and Metadata (MOF, XMI...)

Directory, Security, Database, Web, Transaction, Caching, Metadata, Services..

Distributed Runtime Middleware (IIOP, XML/Value, HTTP...)

(C) Iyengar 2000
UML representation of the emerging OMG E-Business integration vision

<<subsystem>>
Application Development (UML...)

<<subsystem>>
Industry Frameworks
Healthcare, Manufacturing...

<<Subsystem>>
Application Integration EAI/IAI Frameworks (EJB, CCM,

<<Subsystem>>
CORBA IIOP, Component Model,
CORBA/EJB/COM/HTTP

<<Subsystem>>
Business Intelligence Frameworks(CWM)

<<Subsystem>>
Community and Enterprise Portal Frameworks (KM)

<<Subsystem>>
Object Services
(Transactions, Security..)

<<Subsystem>>
Metadata and Component Repositories/Interchange (MOF, XMI)

<<Subsystem>>
Application Management Frameworks
Importance of MOF, XMI and UML

- Model driven approach to software design
- Metadata driven approach to defining and managing software artifacts
- Metamodels and middleware work together to enable integration of systems
- Even middleware can be modeled
- A unifying middleware neutral modeling and metadata framework
- Brings together the best of OMG, W3C, Java and Microsoft Technologies
- ‘A new architectural synthesis’ Paul Harmon, Cutter Information Group
Importance of MOF, XMI and UML

Understand, analyze and design software using UML

- Design and integrate tools, middleware and applications in a flexible manner using MOF
  - Across previous islands of information
- Interchange metadata and objects using XML across platforms in a loosely coupled manner (XMI)
- Interchange metadata and objects using IDL across platforms across languages (MOF2IDL, JMI...)
- A unifying model driven architecture
The UML is a graphical language for
- specifying
- visualizing
- constructing
- documenting
the artifacts of software systems
Added to the list of OMG adopted technologies in November 1997 as UML 1.1
Most recent minor revision is UML 1.3 (November 1999)
Covered in detail already
OMG UML 1.3 Specification

- UML Summary
- UML Semantics
- UML Notation Guide
- UML Standard Profiles
  - Software Development Processes
  - Business Modeling
- UML CORBAfacility Interface Definition (IDL)
- UML XML Metadata Interchange DTD, Document
- Object Constraint Language
Meta Object Facility (MOF) is a *model driven distributed object framework* for

- Specifying
- Constructing
- Managing
- Interchanging and
- Integrating

*metadata* in software systems, thus enabling the flexible integration of systems.
MOF : The Motivation

- Lots of proprietary metadata sources
- Need to manage IT assets (components, documentation, schemas...)
- New metamodules are proliferating
  - OMG, W3C and MDC standards (among others)
  - Customized development methods
  - XML Models (DTDs, Schemas...)
  - Products addressing both new and existing areas
- Need to bridge the gap
  - Common language across metamodules
  - Interoperability and federation of tools, data sources
  - Ease metamodel, tool and middleware development
MOF 1.3 History

- MOF work rooted in OMG Repository RFI in 1995
- MOF 1.1 was adopted along with UML in November 1997 as an OMG standard.
- MOF 1.3 a minor revision was adopted in November 1999 along with UML 1.3
- Key contributors to the MOF 1.1 include
  - Unisys, DSTC, IBM, Oracle, ICL, Objectivity, SSA, CA/Platinum (submitters)
  - EDS, Rational … (Supporters)
- MOF RTF continues refining the spec
  - mof-rtf@omg.org ; www.dstc.edu.au/mof is RTF home page
OMG MOF 1.1/1.3 Specification

- MOF Semantics: OMG Meta-meta model (M3)
- MOF Reflective Interfaces for generic manipulation of metadata {using CORBA IDL}
- MOF to IDL mappings for type safe manipulation of metamodel specific information
- MOF facility for finding and managing federated repositories
- MOF to XML Mapping: OMG XMI Spec
  - XML DTD, XML Documents, XML Schema (initial submission)
- MOF to Java Mapping: SUN JSR-40, JMI Spec
- Covered in detail in Part 2 of this tutorial
XML Metadata Interchange (XMI) is a model driven distributed XML framework for

- representing,
- Interchanging and
- Integrating

metadata and objects in software systems, thus enabling the flexible integration of systems

- XMI is a mapping of MOF to W3C XML
OMG XMI 1.1 History

- UML 1.1 and MOF 1.1 specified interchange using CORBA IDL
  - Not very practical for transmitting large amounts of fine grained data between systems
- OMG issued Stream based Model Interchange Format RFP in December 1997
- Key contributors to XMI include
  - Unisys, IBM, DSTC, Oracle, CA/Platinum, Fujitsu, Softeam, Reccerca, Daimler-Benz (submitters)
  - 20+ OMG members including EDS, NCR, Rational, Inline, Nihon Unisys… (supporters)
OMG XMI 1.0/1.1 Specification

- Mapping of MOF compliant metamodels to XML DTDs
  - DTD provide standard representation of metamodel
  - XML document that conforms to MOF DTD is a preferred representation of metamodel
    - Use XML document itself to represent metamodel
- Mapping of MOF compliant models to XML documents (streams)
  - Transmit and interchange instance data that conforms to the metamodel
  - Mapping to XML Schema is in progress
- XMI Covered in detail in Part 3 of this tutorial
- A new proposal XMI for XML Schema is now being standardized
Integration using UML, MOF and XMI

- Part 1: OMG Modeling and Metadata Architecture
  - Sridhar Iyengar, Unisys Corporation

- Part 2: OMG Meta Object Facility (MOF)
  - Sridhar Iyengar, Unisys Corporation

- Part 3: OMG XML Metadata Interchange (XMI)
  - Stephen Brodsky, IBM

- Part 4: Wrap up and Futures
  - Sridhar Iyengar and Stephen Brodsky
MOF Drill Down

- Definitions and Concepts: Quick tour
  - MOF Architecture
  - MOF Model
  - MOF Reflective Interfaces
  - MOF to IDL Mappings
  - MOF Facility
- MOF based metamodels and applications
  - UML, CWM, IDL, CCM etc
  - Use of Reflective VS Tailored Interfaces
MOF Quick Tour

Meta Object Facility (MOF) is a *model driven distributed framework* for

- Specifying
- Constructing
- Managing
- Interchanging and
- Integrating

*metadata* in software systems, thus *enabling the flexible integration* of systems.
MOF: Specifying Metadata

- The MOF Model (OMG Meta-metamodel) is used to precisely specify technology and domain specific metamodels and object definitions.

- The design specification uses OMG UML notation:
  - Stereotypes and Tagged Values are used to capture additional MOF metadata needed for IDL and XML generation (e.g.: resolve name conflicts for bindings).

- MOF supports a subset of the UML Class modeling concepts:
  - Association Classes and N-Ary Associations need to be modeled as Classes.
  - Makes MOF lightweight and optimized for metadata.
MOF: Constructing and Managing

Constructing Metadata Systems
- The MOF is first used to define a specific metamodel (such as UML or CWM)
- The MOF to IDL mapping allows automatic generation of CORBA IDL interfaces from the metamodel definition
- MOF based CORBA, Java or COM Servers can also be generated to enable more rapid tool construction

Managing Metadata Systems
- MOF based metadata (such as UML designs, CWM Database schemas and transformations) can be programmatically manipulated locally or across a network
- This metadata can be used to manage and tune systems
MOF: Interchanging Metadata

- Systems that conform to the MOF can interchange metadata using CORBA from a variety of programming languages (Java, C++, C...) and middleware systems (COM, DCE...)
  - Typically used for application integration scenarios
- Using XMI (mapping of MOF to XML), MOF based metadata can be exchanged using XML
  - Application development, data warehousing, document management, application integration...
- Using JMI: Java Metadata Interface currently under development using the SUN Java Community Process, Java based metadata can be interchanged
MOF: Integrating Metadata

- Most systems have proprietary metadata used to construct, manage or optimize the system.
- This metadata can be modeled using MOF for each legacy or new system.
- Using MOF and CORBA or MOF and XML, these different sources of metadata can be combined or subsetted to create federated views of the overall system.
  - Example: Various parts of application development life cycle or data warehousing life cycle have different but related metadata.
- You integrate a system by integrating its metadata.
MOF: Model-driven distributed framework

Model driven
- UML is used for design of metamodels
- MOF model (a subset of UML) is used for implementing metamodels

Distributed
- MOF uses CORBA for federation and distribution
- Different MOF based metadata sources and servers can be accessed across the network
- All distributed CORBA services can be leveraged

Framework
- The MOF::Reflective interfaces can be used across meta levels
- The MOF to IDL, XML and Java mappings allow consistent manipulation of all compliant metadata based on well defined design patterns
- The framework is extensible using standard OO techniques
MOF Drill Down

- Definitions and Concepts: Quick tour
  - MOF Architecture
  - MOF Model
  - MOF to IDL Mappings
  - MOF Reflective Interfaces
  - MOF Facility
  - MOF based metamodels and applications
    - UML, CWM, IDL, CCM etc
    - Use of Reflective VS Tailored Interfaces
MOF Overview

Foundation for OMG metadata and modeling architecture

- Model, design and implement metamodels
  - Use UML notation
- Provides 4 generic meta-object interfaces for introspection (Used by all MOF based Meta-models...)
- Provides MOF-IDL mapping to automate generation of concrete object interfaces for specific metamodels
  - Can be used with C++, Java, COM... using appropriate language bindings/gateways
- Provides MOF-XML mappings to automate generation of XML DTDs and Documents (XMI specification)
- Provides MOF-Java mapping as part of SUN Java Community Process JSR-40 (In progress)
MOF Architecture

- Four layered metamodell architecture
  - Integrated with CORBA, UML, XML and Java

- Packages/CORBA Modules
  - Model : the meta-metamodell
  - Reflective : generic metadata interfaces
  - Facility : repository/model management

- Aligned with the OMG UML Core
  - Uses UML Class Diagram notation for metamodell schemas
MOF is at the root of the OMG 4 layered Metamodel Architecture.

- **M3 Layer**: Specifies meta-metaclasses for the UML metamodel.
- **M2 Layer**: Specifies metaclasses for the UML metamodel, such as Class.
- **M1 Layer**: Specifies classes for the UML user models, such as Passenger, Ticket, TravelAgency.
- **M0 Layer**: User objects that are instances of UML user model classes, such as instances of Passenger, Ticket, TravelAgency.
MOF Package Architecture

Discover & Manipulate metadata

Find and Manage Metadata Repositories, Servers

Model using UML Class Diagrams precisely

<<OMG Meta-metamodel>>
MOF Model

<<CORBA IDL Module>>
MOF 1.3 Reflective

MOF Facility
The MOF Model (OMG M3)

- Relatively small but not minimal to facilitate UML alignment
- Rich enough for variety of metamodels
  - Proven in application development (UML), distributed middleware (IDL, CCM, EJB), data warehouse (CWM)
- Inbuilt and model-specific constraints
- Supports discovery, extensibility, sharing
- CORBA-based and distributable
- An interface not an implementation (could be mapped to a relational catalog, memory, XML...)

UML, MOF & XMI
OMG MOF Model: Key Classes

- **Constraint**
  - Namespace
  - Package
  - Classifier
  - Operation
  - Reference
  - MofAttribute

- **ModelElement**
  - GeneralizableElement
  - BehavioralFeature
  - StructuralFeature

- **Contains**
  - +container
    - {ordered}
  - +containedElement

- **Generalizes**
  - +subtype
    - {ordered}0..*
  - +supertype
    - {ordered}0..*

- **ModelElement**
  - 0..1
  - 0..*

- **Feature**
  - 0..*
  - 1..*

- **Constraint**
  - 0..*

- **Namespace**
  - +container
  - 0..1
  - +containedElement
    - 0..*

- **Classifier**
  - +supertype
    - 0..*
  - +subtype
    - 0..*

- **Operation**
  - 0..*

- **Reference**
  - +referent
    - 0..*
  - +referencedEnd
    - 1

- **MofAttribute**
  - +referent
    - 0..*
  - +referencedEnd
    - 1

- **AssociationEnd**
  - 1..*

- **Association**
  - +constrainedElement
    - 1..*

- **Constraint**
  - 0..*

- **ModelElement**
  - +constrainedElement
    - 0..*
MOF Inheritance Hierarchy

ModelElement

- Reflective
  - RefBaseObject
    - RefPackage
    - RefObject
    - RefAssociation

ModelElement

- Tag
- Constraint
- Import
- Namespace
- Feature
- TypedElement

GeneralizableElement

- Classifier
- Operation
- MoF Exception
- Reference
- MoF Attribute

DataType

Class

Association
MOF Model vs. UML Core

MOF does not need a full blown object: Generalization, Dependency

- Associations are simpler: binary only, not objects in own right (no AssociationClass), mapped to References on classes, no N-ary associations
  - These constraints are well formedness rules.
- Simpler types/options e.g. Multiplicity
- More constrained - less freedom, more completeness needed
- Assumption of federation (e.g. a class does not know its subclasses)
MOF Model Interfaces

Core

- Package (Model)
- Class
- DataType
- Attribute
- Operation
- Reference
- Association
- AssociationEnd
- Constraint

Supporting

- Exception
- ModelElement
- Namespace (Container)
- Classifier
- BehavioralFeature
- StructuralFeature
- Parameter
- Tag
- Various Associations
MOF Details: Classifiers

**ModelElement**
- name : NameType
- annotation : AnnotationType
- qualifiedName : NameType
- verify()
- isFrozen()
- findRequiredElements()
- isVisible()
- isRequiredBecause()

**Namespace**
- lookupElement()
- resolveQualifiedName()
- nameIsValid()
- findElementsByType()

**Package**
- externalize()
- internalize()

**Classifier**
- Class
  - sSingleton : boolean
- DataType
  - typeCode : TypeDescriptor
- Association
  - isDerived : boolean
MOF Details: Features

- ModelElement
  - name : NameType
  - annotation : AnnotationType
  - qualifiedName : NameType
  - verify()
  - isFrozen()
  - findRequiredElements()
  - isVisible()
  - isRequiredBecause()

- Feature
  - visibility : VisibilityKind
  - scope : ScopeKind

- StructuralFeature
  - multiplicity : MultiplicityType
  - isChangeable : boolean

- BehavioralFeature

- Reference

- Attribute
  - isDerived : boolean

- Operation
  - sQuery : boolean

- Exception
MOF Detail: TypedElements

ModelElement
- name : NameType
- annotation : AnnotationType
- qualifiedName : NameType
- verify()
- isFrozen()
- findRequiredElements()
- isRequiredBecause()
- isVisible()

TypedElement

AssociationEnd
- multiplicity : MultiplicityType
- aggregation : AggregationKind
- isNavigable : boolean
- isChangeable : boolean
- otherEnd : AssociationEnd

Parameter
- direction : DirectionKind
- multiplicity : MultiplicityType

Constant
- value : any

StructuralFeature
- multiplicity : MultiplicityType
- isChangeable : boolean
- sChangeable : boolean

TypeAlias

Reference

<<MofAttribute>>
- Attribute
- sDerived : boolean
MOF Drill Down

- Definitions and Concepts: Quick tour
  - MOF Architecture
  - MOF Model
  - MOF to IDL Mappings
  - MOF Reflective Interfaces
  - MOF Facility
  - MOF based metamodels and applications
    - UML, CWM, IDL, CCM etc
    - Use of Reflective VS Tailored Interfaces
MOF-IDL Mapping & Generation

- MOF spec includes a MOF-IDL mapping spec that can be used to automate IDL generation for metamodels
  - This mechanism was used to generate the IDL for both MOF and UML
  - This mechanism can be used to generate CORBA metamodel servers
  - A similar mapping from UML to IDL is necessary (UML Profile for CORBA addresses this partially)

- Users can choose generic (reflective) MOF interfaces or specific (concrete) interfaces for manipulating the UML metamodel
MOF - IDL Mapping

- Templates provided for each MOF construct
  - Package, Class, Attribute, Association, Reference, Operation etc.
  - For example each MOF Class is mapped to two interfaces
    - Factory interface which has constructor and finder methods
    - Object interface which has the methods on individual objects
  - Another example: each MOF Attribute is mapped to two accessor and mutator methods

- Conformance to MOF implies that these interfaces are implemented
  - Auto generation not required by spec but usually implemented
Reflective vs. Generated Interface

Generated
- AttributeClass::create_Attribute(…constructor args…)
- Attribute::set_InitialValue(string)
- string Attribute::initialValue()

Reflective
- refObject::createInstance(…constructor args…)
- refObject::setValue(DesignatorType, ValueType)
- refObject::value(DesignatorType)

Feature Designator found by discovery or look-up by name

More coding involved

But allows working with models unknown when tool built

Opens way for new class of generic tools
BehavioralFeature: MOF - IDL

Namespace

Feature
visibility : VisibilityKind
scope : ScopeKind

BehavioralFeature

Operation
isQuery : boolean

+operation 0..*

CanRaise
+exception 0..* {ordered}

MofException
Interface OperationClass

interface BehaviouralFeature :
    BehaviouralFeatureClass, Feature, Namespace {};

interface OperationClass : BehaviouralFeatureClass {
    readonly attribute OperationUList all_of_kind_operation;
    readonly attribute OperationUList all_of_type_operation;

    const string operation_has_at_most_one_return_parameter =
        "::Model::Operation::operation_has_at_most_one_return_parameter";
    const string operation_containment_rules =
        "::Model::Operation::operation_containment_rules";
    const string exception_visible_from_operation =
        "::Model::Operation::exception_visible_from_operation";

    Operation create_operation (  
        /* from ModelElement */ in ::Model::NameType name,  
        /* from ModelElement */ in ::Model::AnnotationType annotation,  
        /* from Feature */      in ::Model::ScopeKind scope,  
        /* from Feature */      in ::Model::VisibilityKind visibility,  
        /* from Operation */    in boolean is_query)  
        raises (Reflective::SemanticError);
}; // end of interface OperationClass
interface Operation : OperationClass, Behavioural Feature {
    boolean is_query ();
    raises (Reflective::StructuralError, Reflective::SemanticError);
    void set_is_query (in boolean new_value)
    raises (Reflective::SemanticError);
    MofExceptionUList exceptions ()
    raises (Reflective::ConstraintError, Reflective::SemanticError);
    void set_exceptions (in MofExceptionUList new_values)
    raises (Reflective::StructuralError,
            Reflective::ConstraintError,
            Reflective::SemanticError);
    void add_exceptions (in MofException new_value)
    raises (Reflective::StructuralError,
            Reflective::ConstraintError,
            Reflective::SemanticError);
    void add_exceptions_before (in MofException new_value,
                                 in MofException before)
    raises (Reflective::StructuralError,
            Reflective::NotFoundException,
            Reflective::ConstraintError,
            Reflective::SemanticError);
    void modify_exceptions (in MofException old_value,
                            in MofException new_value)
    raises (Reflective::StructuralError,
            Reflective::NotFoundException,
            Reflective::ConstraintError,
            Reflective::SemanticError);
    void remove_exceptions (in MofException old_value)
    raises (Reflective::StructuralError,
            Reflective::NotFoundException,
            Reflective::ConstraintError,
            Reflective::SemanticError);
}; // end of interface Operation
MOF Reflective Interfaces

- RefBaseObject
- RefPackage
- RefObject (Class)
- RefAssociation

- Used for general purpose (metamodel independent) metaobject manipulation
- Used for interoperability across metamodels
MOF 1.3 Reflective Package

- All meta-models based on the MOF inherit this package.

- Can be used to manipulate any MOF based metadata.
Reflective vs. Generated Interface

Generated
- `AttributeClass::create_Attribute(…constructor args…)`
- `Attribute::set_InitialValue(string)`
- `string Attribute::initialValue()`

Reflective
- `refObject::createInstance(…constructor args…)`
- `refObject::setValue(DesignatorType, ValueType)`
- `refObject::value(DesignatorType)`

Feature Designator found by discovery or look-up by name

More coding involved

But allows working with models unknown when tool built

Opens way for new class of generic tools
Metamodelling using MOF & UML

Understand the metamodeling (representation of models) domain

- Object modeling (UML), Business Objects (UML For Business Objects...), Database Management, Data warehousing etc.
- Define initial metamodel reusing concepts from existing metamodels when possible
- Use UML core modeling constructs available

Use UML based visual modeling tools
- These can be customized for the MOF using UML/MOF extension mechanisms
Metamodeling Using MOF & UML

Use UML for analysis and design of metamodels (these are models after all!)

- Define metamodels using the MOF
  - Use UML based modeling tools, or MOF interfaces (normative) or XMI
  - Enables life cycle metadata interoperability and design reuse across metamodels
  - Relationships and subtyping across metamodels supported by the MOF

- Use MOF-IDL mappings for concrete IDL interfaces to metamodels
- Inherit MOF Reflective interfaces for interoperable metaobjects across metamodels
- Use XMI for exchanging metadata
Practical scenarios of MOF Usage

- Exchange of object models between development tools using UML
- Metadata foundation for development of business object applications (refinement and relationships across metamodels)
- Use of object repositories for integrating development tools, data warehouse tools and business object tools
- Interoperable metadata registries/repositories
- Exchange metadata over the Web using IIOP, HTTP, MOF and XMI
XMI/MOF for E-Business Application Development OMG November 98 Demo

- IBM VisualAge
- Oracle Repository
- Unisys UREP
- Select
- WebSphere
- Rose
- VA TC
- DTD Gen
- VA Java
- XMI
- MOF/XMI
- Distributed Runtime Middleware (IIOP, HTTP...)
- Community & Enterprise Information Portals (KM...)
- HealthCare
- Financial
- Manufacturing
- Insurance...
- E-Business Application Development UML, CCM
- E-Business Intelligence CWM, OIM
- E-Business Application Integration CCM, EJB, COM+, XML
- Information Models, Components and Metadata (XML, MOF...)
- Directory, Security, Database, Web, Transaction, Caching, Metadata, Services...
- Healthcare
- Financial
- Manufacturing
- Insurance...

UML, MOF & XMI
Common Warehouse Metamodel (CWM)

Scope
- Data Warehouse lifecycle metadata management
- Contributors: IBM, Unisys, NCR, Hyperion, Oracle, Genesis, UBS, Dimension EDI...
  - Metamodel - Single logical & physical!
  - Generated XML DTDs
  - Generated MOF - IDL mappings
  - Generated XML document
- Adopted by OMG in June 2000 (Oslo)
- CWM Interoperability showcase in Sep 2000
CWM 1.0 Overview {02/2000}
Common Warehouse Metamodel

Warehouse Management
- Warehouse Process
- Warehouse Operation

Analysis
- Transformation
- OLAP
- Data Mining
- Information Visualization
- Business Nomenclature

Resources
- Object-Oriented (UML)
- Relational
- Record-Oriented
- Multi Dimensional
- XML

Foundation
- Business Information
- Data Types
- Expressions
- Keys
- Index
- Type Mapping
- Software Deployment

UML 1.3
(Foundation, Behavioral_Elements, Model_Management)
The major packages in CWM:
- org.omg.uml  { UML 1.3 }
- org.omg.cwm    { CWM Core }
- org.omg.cwmx   { Extensions }
CWM 1.0 - Overview

- Foundation
- Resource

- Management
- Analysis

- CORBAIDL
- ER

<<metamodel>>
CWM 1.0 - Management

<<metamodel>>
WarehouseProcess

<<metamodel>>
WarehouseOperation

No Extensions
MOF/XMI for Application Integration and Industry Specific Domains

- OMG XML/Value (2000)
- UML Profile for EAI, I-EAI (2000)
- Integration of OMG and OASIS Technologies/Frameworks
  - XML/XMI Repository Discussions underway with ebXML
- Knowledge Management (MDC and OMG efforts)
- OMG Healthcare, Manufacturing, ECDTF…
- Most EAI vendors using/planning XML for B2B
Travel Business Service Scenario

Customer
• requests Itinerary

BXS Travel
• interacts with multiple systems
• responds with Itinerary

BXS Travel Service: Make Travel Arrangements

Flight Reservation

Hotel Reservation

Customer

Custom Repository
### Fragments of the ‘Travel Business Model’

- **ItineraryRequestDocument**
  - `custName : string`
  - `pin : short`
  - `departureDate : string`
  - `departureCity : string`
  - `arrivalDate : string`
  - `arrivalCity : string`
  - `economyClass : string`
  - `accomodationsAreOptional : string`
  - `numberOfDays : string`

- **CustomerDocument**
  - `name : string`
  - `custNum : short`
  - `address : string`
  - `telephone : string`
  - `e_mail : string`
  - `category : short`

- **AccomodationRequest**
  - `checkInDate : string`
  - `numberOfDays : short`
  - `city : string`
  - `country : string`

- **AccomodationDocument**
  - `hotel : string`
  - `type : short`
  - `station : string`
  - `telephone : string`
  - `url : string`

- **FlightRequestDocument**
  - `departureDate : string`
  - `departureCity : string`
  - `arrivalDate : string`
  - `arrivalCity : string`

- **FlightDocument**
  - `departureDate : string`
  - `arrivalDate : string`
  - `flightNumber : string`
  - `charge : short`
  - `economyClass : string`
<?xml version="1.0" encoding="utf-8"?>
<![DOCTYPE XML (View source for full doctype...)]>
  <xml version="1.0">
    <xml header>XML model xmlns="Travel" xml:version="1.3" />
    <xml metamodel xmlns="Uml" xml:version="1.3" />
    <xml metamodel xmlns="Model" xml:version="1.3" />
  </xml header>
  <xml content>
    <ItineraryRequestSignal xmlns="b.1">
      <ItineraryRequestSignal id="a.1">
        <ItineraryRequestDocument xmlns="Travel">
          <departureCountry>USA</departureCountry>
          <economyClass>yes</economyClass>
          <accommodationsAreOptional>yes</accommodationsAreOptional>
          <custName>Mr. Cust</custName>
          <departureCity>Miami</departureCity>
          <arrivalCountry>USA</arrivalCountry>
          <arrivalCity>Las Vegas</arrivalCity>
          <departureDate>20010125</departureDate>
          <arrivalDate>20010125</arrivalDate>
          <numberOfDays>1</numberOfDays>
        </ItineraryRequestDocument>
      </ItineraryRequestSignal id="a.1">
    </ItineraryRequestSignal>
  </xml content>
</xml>
MOF Compliant Metamodels

- UML 1.3 (Physical)
- MOF 1.3
- CWM 1.0
- IDL (CORBA 2.3)
- CORBA/CCM
- CIAS
- Various emerging UML profiles
- Vendor specific metamodels etc.
An Integration and Interoperability Bus for Objects, Components, Models, Metadata and Knowledge
Further Info

Web:
- OMG UML Resource Page
  - www.omg.org/uml/
- UML Tutorial 4 will be posted next week
- www.unisys.com/marketplace/urep
- www.dstc.edu/au/mof

Email
- Sridhar Iyengar : sridhar.iyengar2@unisys.com

Conferences & workshops
- UML World 2000, NYC, March ‘00
- Java One, SFO, June ‘00
- UML ’00, York, England, Oct. ‘00
- Enterprise Repository 2000, England, Nov 00