Common Warehouse Metamodel (CWM): Extending UML for Data Warehousing and Business Intelligence

OMG First Workshop on

UML in the .com Enterprise: Modeling CORBA, Components, XML/XMI and Metadata

November 6-9, 2000, Palm Springs, CA

Speakers

Daniel T. Chang
IBM
(dtchang@us.ibm.com)

John D. Poole
Hyperion Solutions
(john_poole@hyperion.com)

Objectives

- Overview of CWM concepts
- Use of UML by CWM
- UML lessons learned from CWM
- Discuss moving forward

Where to find my modified slides and notes...

http://www.cwmforum.org/

http://www.omg.org/uml

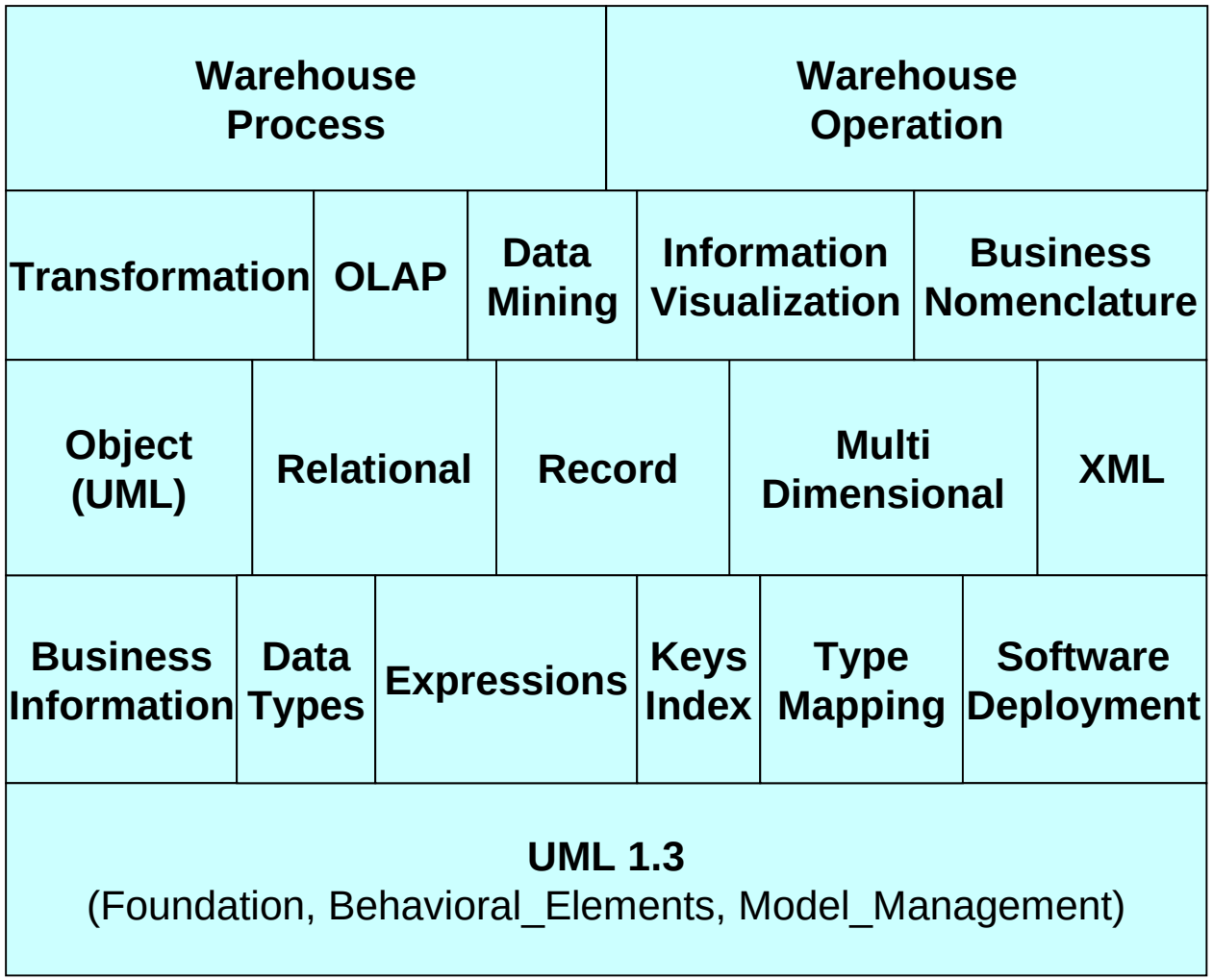
A ballot lock box down in Florida!

What is CWM?

- A complete specification of syntax and semantics that Data Warehousing and Business Intelligence tools can leverage to successfully interchange shared metadata
- A language or framework for specifiying the external representation of data warehouse metadata for purposes of interchange

CWM Provides...

- A standard language for defining the structure and semantics of metadata in a formal way (MOF / UML / OCL)
- A standard interchange mechanism for sharing metadata defined in the standard language (XML / XMI)
- A standard specification (interface) for access to, and discovery of, the metadata defined in the standard language (IDL for now, normative Java API with JSR-40)


Management
Warehouse Process
Warehouse Operation
Analysis
Transformation
OLAP
Data Mining
Information Visualization
Business Nomenclature
Resource
Object (UML)
Relational
Record
Multi Dimensional
XML
Foundation
Business Information
Data Types
Expressions
Keys Index
Type Mapping
Software Deployment
UML 1.3
(Foundation, Behavioral_Elements, Model_Management)

CWM Design Rationale

- MOF is the modeling *language* (syntax+semantics)
- UML is the modeling *notation*
- UML is the also the base metamodel
- XMI is the interchange mechanism

CWM Design Rationale

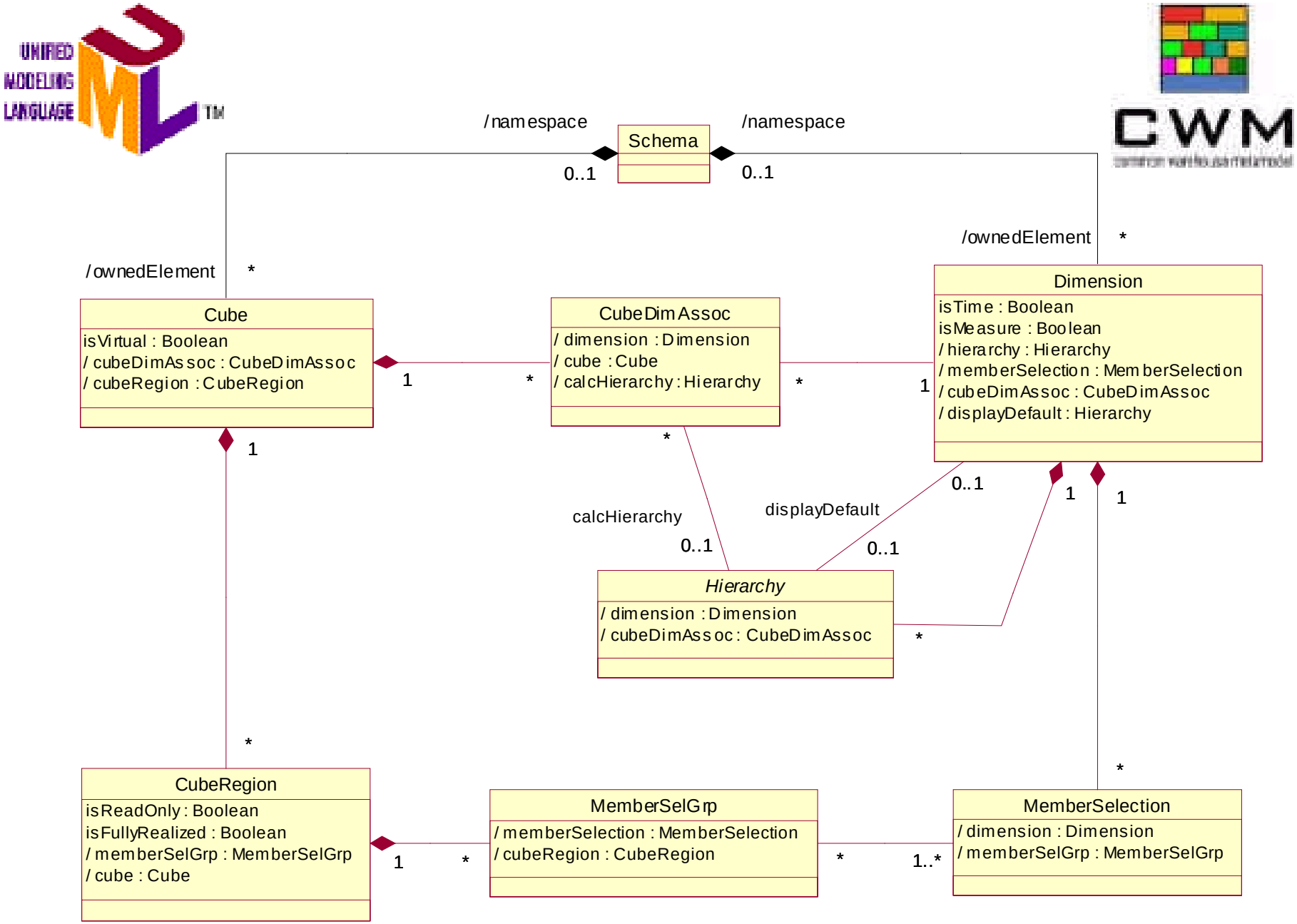
- Extend the UML metamodel with Data Warehousing and Business Intelligence domain objects
- Standardize on MOF semantics
- Yields a MOF-compliant metamodel (M2 level) for constructing DW & BI models of data (metadata)
- Effectively defines a UML-aligned notation for specifying DW & BI metadata

UML as Modeling Notation

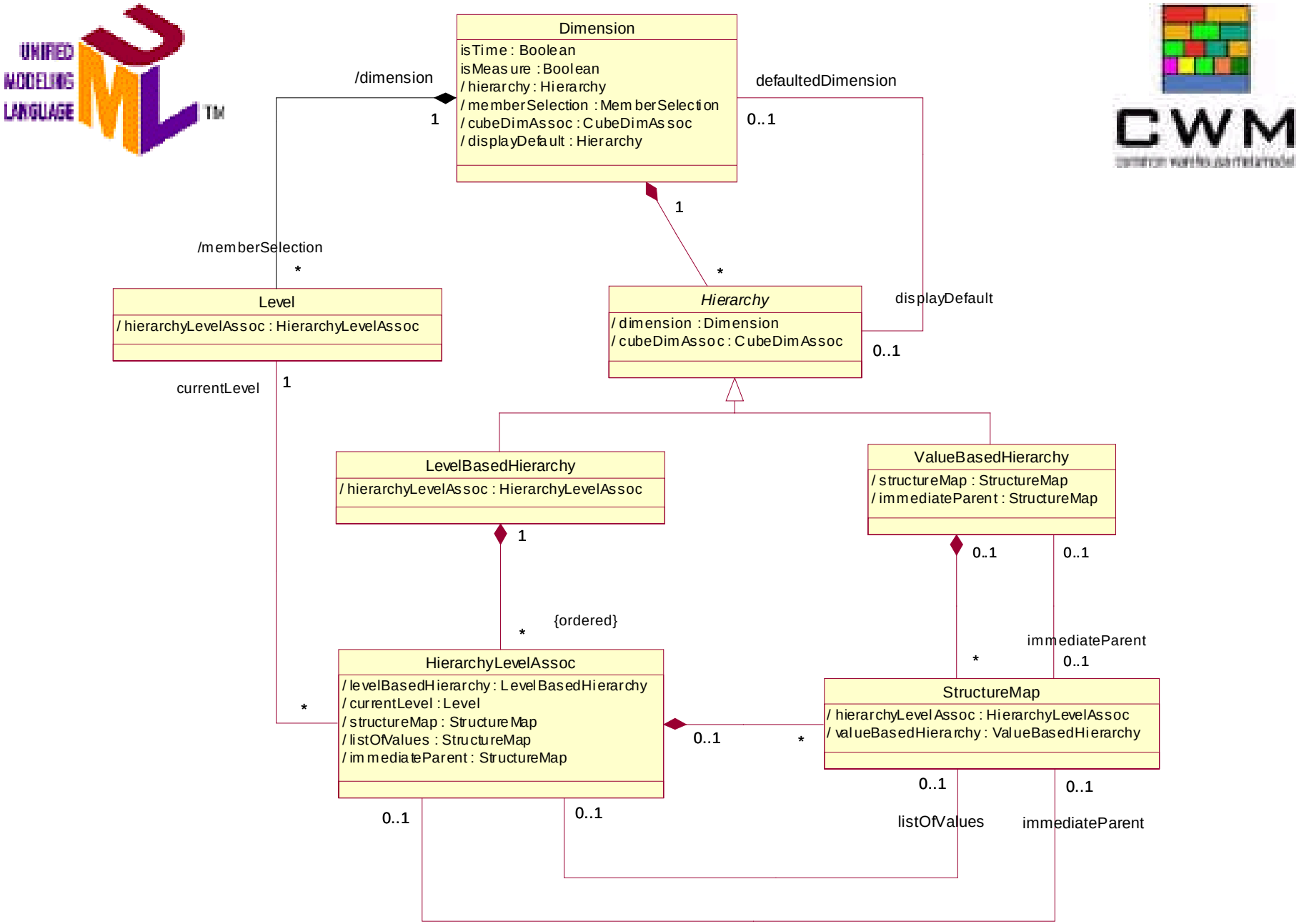
- Structure and typing in UML (versus MOF)
- Notational conventions and usage
- OCL for precise modeling

Structural issues: Association Classes

- Association classes: Defined by UML but not permitted by MOF
- CWM has a number of places where Association Classes would've been useful (to attach attributes to certain M:M associations)
- We seemed to be able to get by using Classes



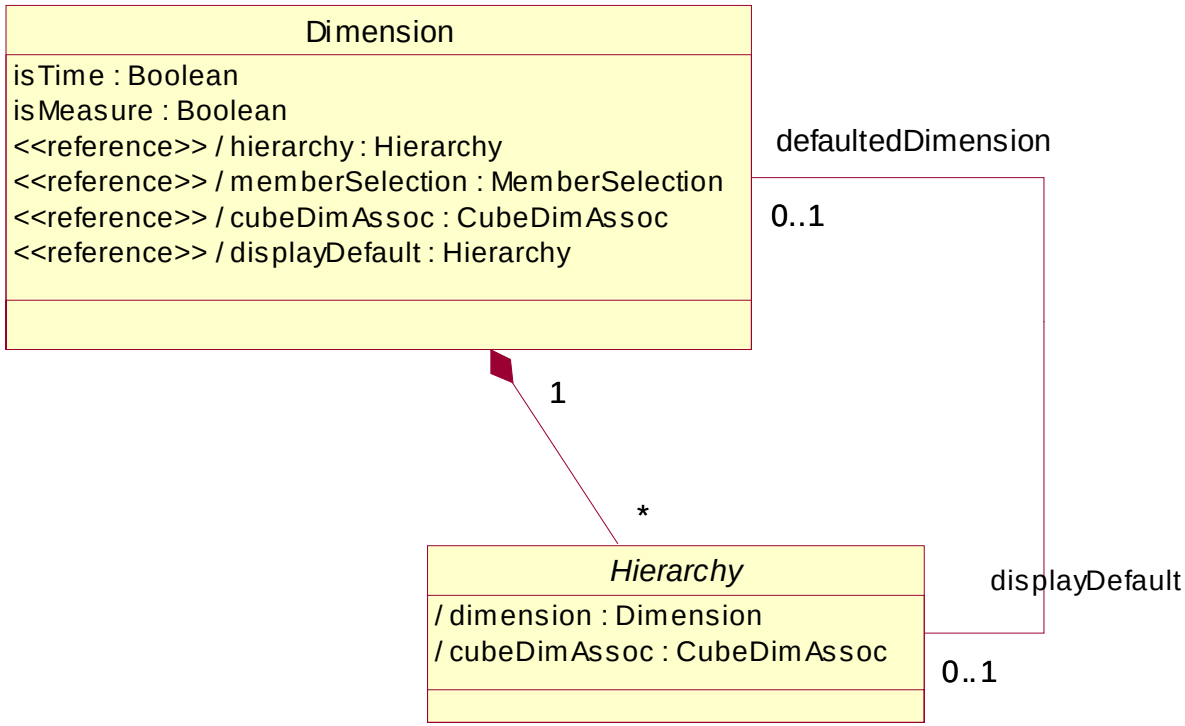
Chang, Poole -- CWM: Extending UML for Data Warehousing and Business Intelligence



Chang, Poole -- CWM: Extending UML for Data Warehousing and Business Intelligence

Structural Issues: References

- MOF allows the specification of references
- UML has no notational support for references
- Arguably an implementation issue, but CWM metamodel is a MOF metamodel…
- Notationally, we represented references as derived attributes (i.e., an attribute computed from a related association)
- “/” + attribute name, <<reference>> stereotype


Dimension
isTime : Boolean
isMeasure : Boolean
<<reference>> / hierarchy : Hierarchy
<<reference>> / memberSelection : MemberSelection
<<reference>> / cubeDimAssoc : CubeDimAssoc
<<reference>> / displayDefault : Hierarchy
defaultedDimension
0..1
1
*
Hierarchy
/ dimension : Dimension
/ cubeDimAssoc : CubeDimAssoc
displayDefault
0..1

Structural Issues: Assoc. Subclassing

- Association subclassing not supported by Rose
- Associations inherited from UML metamodel
  - Namespace/ownedElement
  - Owner/feature
- We would've liked to have been able to subclass these in some cases (restrict extent)
- Used OCL constraints on association ends instead
- Note: This is a tool issue, not a MOF/UML issue

Typing: Inheritance of UML Type System

- Both MOF and UML have type systems (an alignment issue)
- CWM was compelled to use/extend the UML type system, rather than the MOF type system, because of inheritance from UML metamodel

Use of OCL for Precise Metamodeling

- Extensive use of OCL throughout all of the CWM metamodels
- In some cases, compensate for MOF/UML/Tool dissonance (e.g., constraints on association ends)
- Need for some specialized collection operations (e.g., transitive closure)
- Where to store? (e.g., Rose Documentation pane)

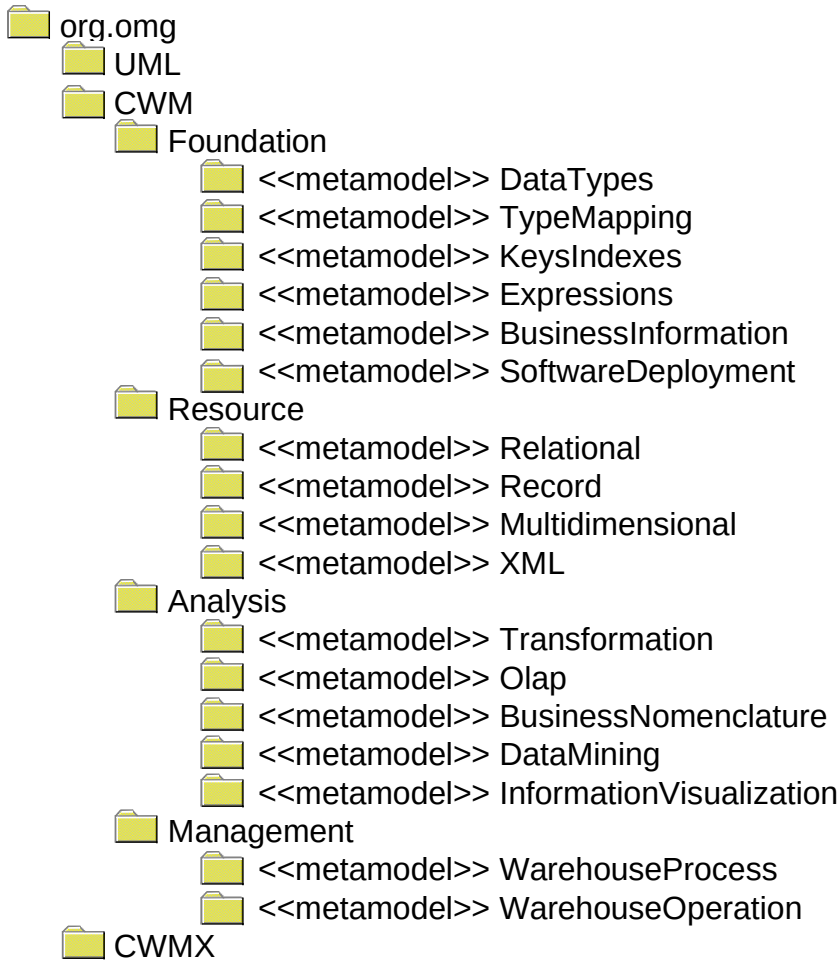
UML as Base Metamodel

- Packaging structure and modularity
- Core metamodel and extended metamodels
- Representing instances
- Traversal of modeling abstraction hierarchy

Package Structure & Modularity

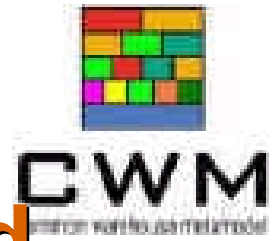
- UML 1.3 metamodel is rather course grained and tightly coupled (i.e., has many package dependencies)
- We tried to compensate in CWM through:
  - Use of a flat, namespace hierarchy
  - Use of <<metamodel>> stereotype to specialize UML Package
  - CWMX extensions: Minimal dependencies

Package Structure & Modularity



Core Metamodels & Extended Metamodels

- CWM core metamodels extend UML “core” metamodel packages:
  - Foundation (Core, Data_Types)
  - Behavioral_Elements::Common_Behavior
  - Model_Management
- Use inherited UML Extensibility Mechanisms (TaggedValue, Stereotype, Constraint) to extend M1 CWM models (without extending CWM)



Core Metamodels & Extended Metamodels

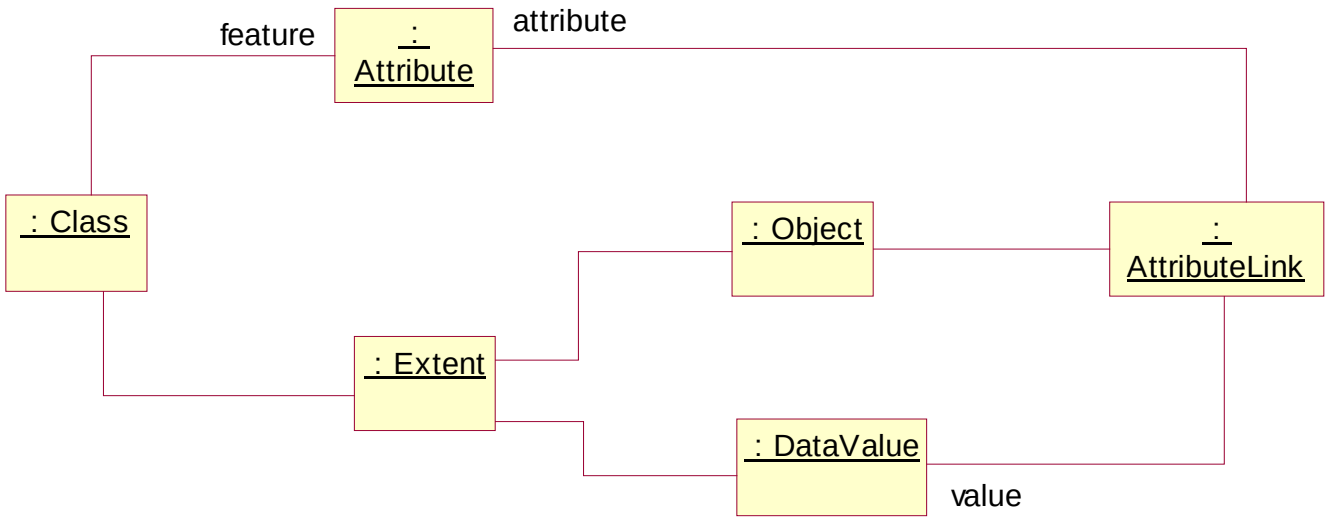
- CWMX (extension package)
- Vendor-specific metamodels that directly extend core CWM metamodels
- Fairly easy to extend the “core” CWM by introducing custom technology metamodels (facilitates interchange)
- An argument in favor of the “create a new metamodel” approach versus the UML profile approach?

Representing Instances

- UML::Behavioral_Elements::Common_Behavior
- Representations of Instance (Object, DataValue), AttributeLink (slot) and Classifier-Instance association
- CWM introduces the concept of Extent (a collection of instances -- e.g., a Relational RowSet)

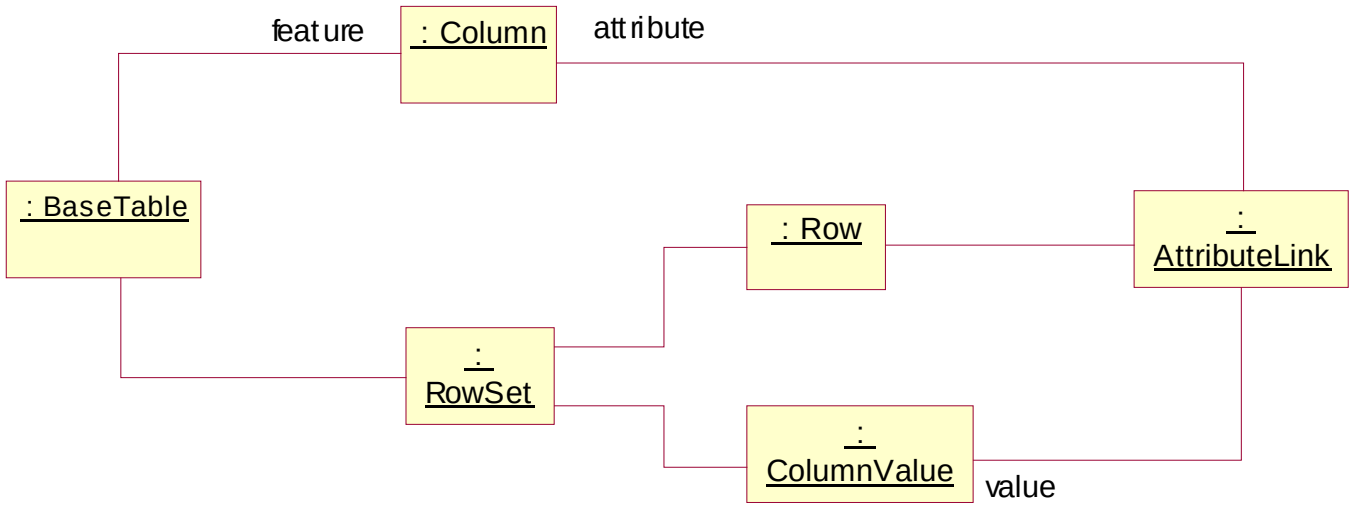
Representing Instances

Object-Oriented Resource Instance



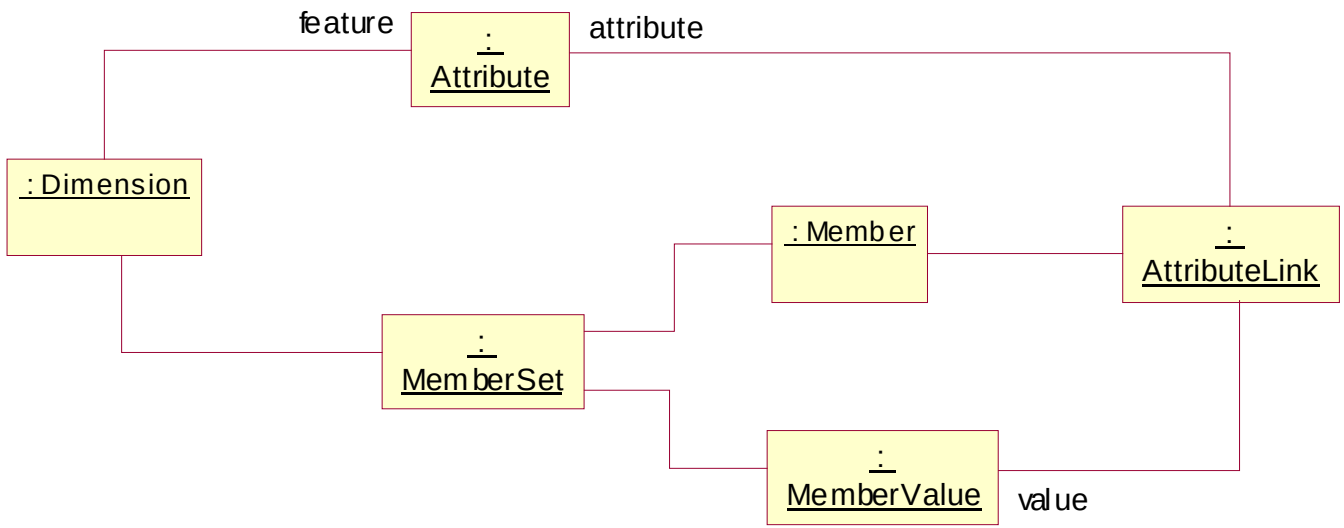
Representing Instances

Relational Resource Instance



Representing Instances

Multdimensional Resource Instance

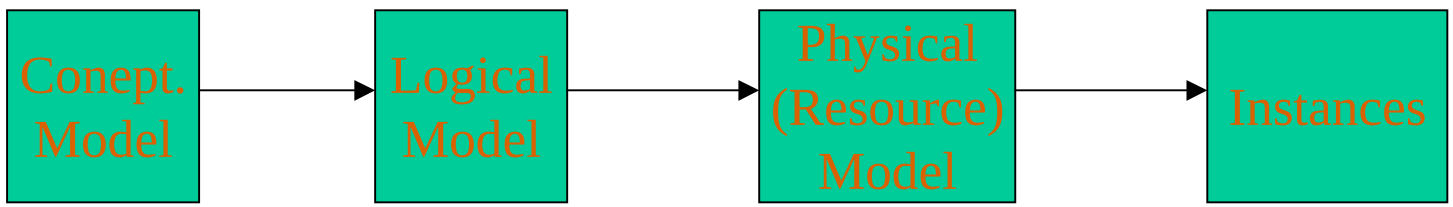


Traversal of Modeling Abstraction Layers

- With CWM, it is possible to span the modeling hierarchy (M2 through M0) in a single XMI document:
  - M2: tags describing metadata
  - M1: contents of the M2 tags
  - M0: linked content of M1 instances

Traversal of Modeling Abstraction Layers

- It is also possible to interchange a single CWM model that spans conceptual, logical and physical boundaries (plus instances!) using the CWM Transformation metamodel:



Traversal of Modeling Abstraction Layers

- The CWM Transformation Metamodel derives its power and flexibility by the way it references core elements of the UML metamodel:
  - ClassifierMap
  - FeatureMap
  - ClassifierFeatureMap

UML Tools

- UML conformance issues
- Rose Profile for UML / MOF
- Metadata repositories
- Metadata bridges/adapters/toolkits

Rose Profile for MOF

- Two-way mapping between Rose's interpretation/implementation of UML, and the MOF
- Resolves the overall dissonance between Rose, UML and MOF (at least to the extent that it assists us in generating interfaces and servers from MOF-compliant metamodels)
- Based on UML Profile for MOF

Follow-on Discussion Points

- UML and MOF alignment
  - A major problem area for CWM
  - UML 2.0: Physical Metamodel to be replaced by MOF
  - Need to ensure that this does indeed resolve the alignment issues between MOF and UML

Follow-on Discussion Points

- UML Profiles
  - Creation of new metamodels (CWM) versus definition of UML Profiles
  - New metamodels facilitate interchangeable metadata (necessarily the case with profiles?)
  - Proliferation of new metamodels: New notations
  - Proliferation of new profiles: New dialects of the same notation

Follow-on Discussion Points

- Object Constraint Language
  - Need to be able to include OCL statements as part of a UML model
  - Need to be able to harvest OCL from model automatically and submit to a model checker
  - OMG specification issues (OCL is considered normative but we have no way of checking if OCL is valid)

Follow-on Discussion Points

- Dynamic systems and dynamic object models
  - MOF reflection
  - CWM extends into DW / BI domain
- Patterns for construction
  - Variation points (stability)
- Patterns for interchange
  - Push / pull paradigm
  - Request / response paradigm

Follow-on Discussion Points

- Concept of “parameterized” transformations
  - Effectively supported by CWM Transformation metamodel
  - Complete spectrum from black box to white box transformations

CWM Information Sources

- OMG home page CWM link
- CWM Forum home page
  - Other misc. info (presentations, papers, links)
  - http://www.cwmforum.org/
- OMG UML home page
  - http://www.omg.org/uml

Java Community Process Related Efforts

- Java Metadata Interface (JMI)
- Java OLAP Interface (JOLAP)
- Java Data Mining API (JDMAPI)

(http://java.sun.com/aboutJava/communityprocess/)

Summary

- Brief overview of CWM
- CWM and UML
- UML as Modeling Language
- UML as Base Metamodel
- UML Tools
- Issues
- Information Sources