The CWM Experience Implementing a UML-Based Data Warehouse Metamodel

Doug Tolbert

doug.tolbert@unisys.com

Agenda

- Overview of CWM
- Modeling Issues
  - Package boundaries
  - Minimizing package dependencies
  - Associations, references, and reuse
- Was UML a useful foundation metamodel for CWM?
  - Design vs. Implementation
  - Package granularity is critical
  - The CWM ObjectModel
- Suggestions for UML 2

CWM Charter

CWM

"Standard interfaces that can be used to enable easy interchange of warehouse and business intelligence metadata between warehouse tools, warehouse platforms and warehouse metadata repositories in distributed heterogeneous environments." – www.omg.org

Charter

"A complete specification of the syntax and semantics needed to export/import warehouse metadata and the common warehouse metamodel. This may consist of a specification for the common warehouse metamodel, APIs (in IDL), and/or interchange formats." -- Common Warehouse Metadata Interchange RFP, OMG document ad/98-07-16, page 2.

Sponsors

Co-submitters

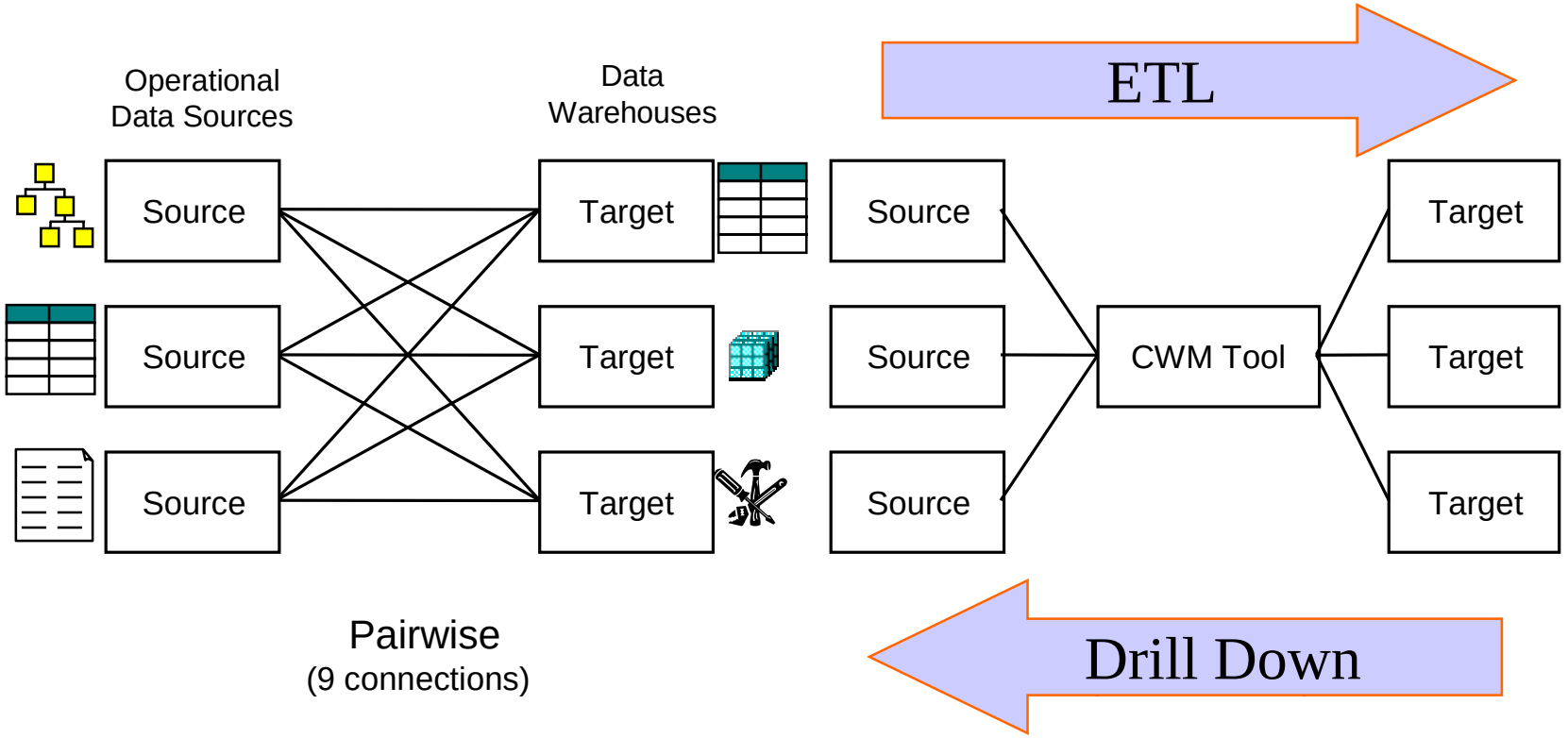
IBM, Unisys, NCR, Hyperion, Oracle, UBS, Genesis, Dimension EDI

Supporters

Deere, Sun, HP, Data Access Technologies, InLine Software, Aonix, Hitachi, SAS Institute, Meta Integration Technology

Specs: http://www.omg.org Info: http://www.cwmforum.org

The CWM Metamodel



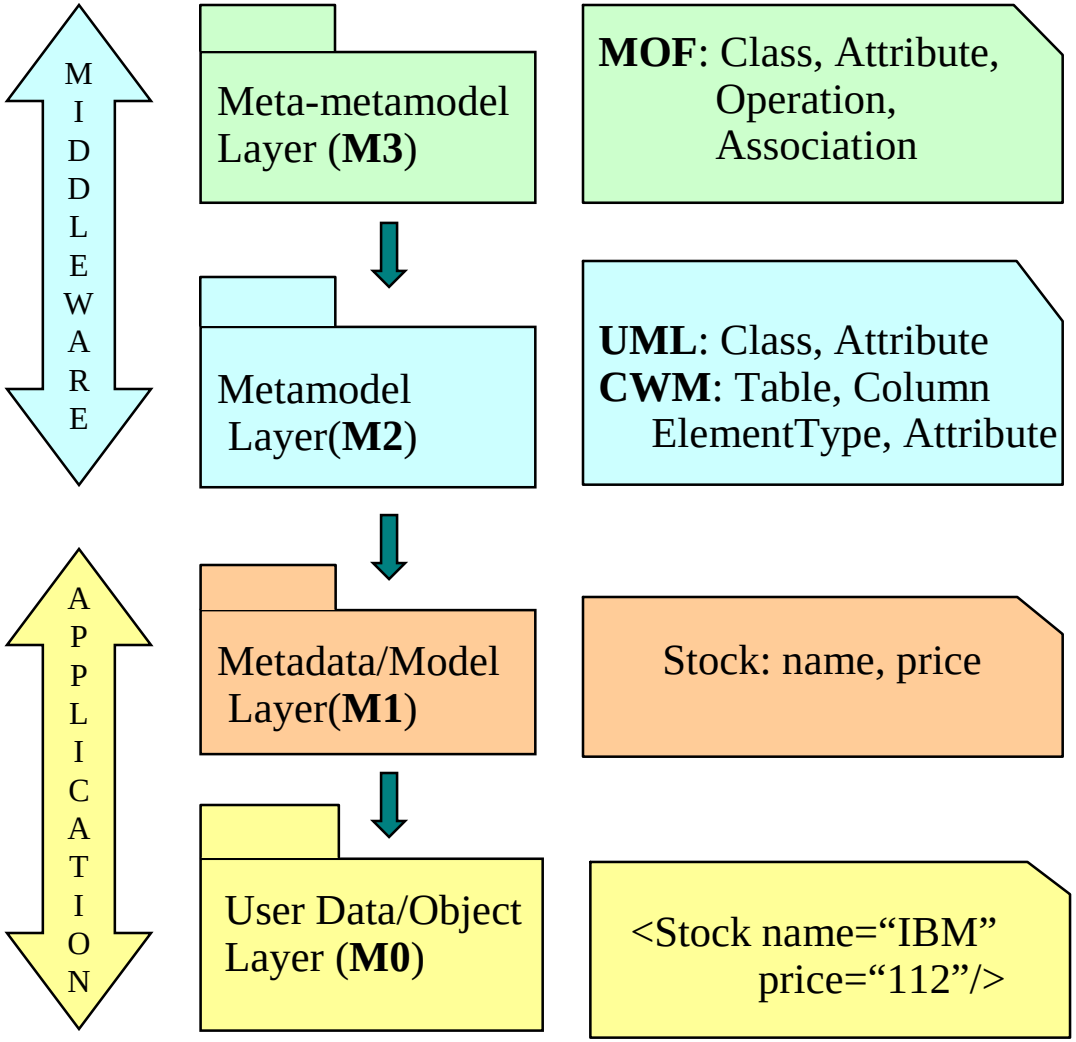
OMG Metamodel Architecture

Standard Components

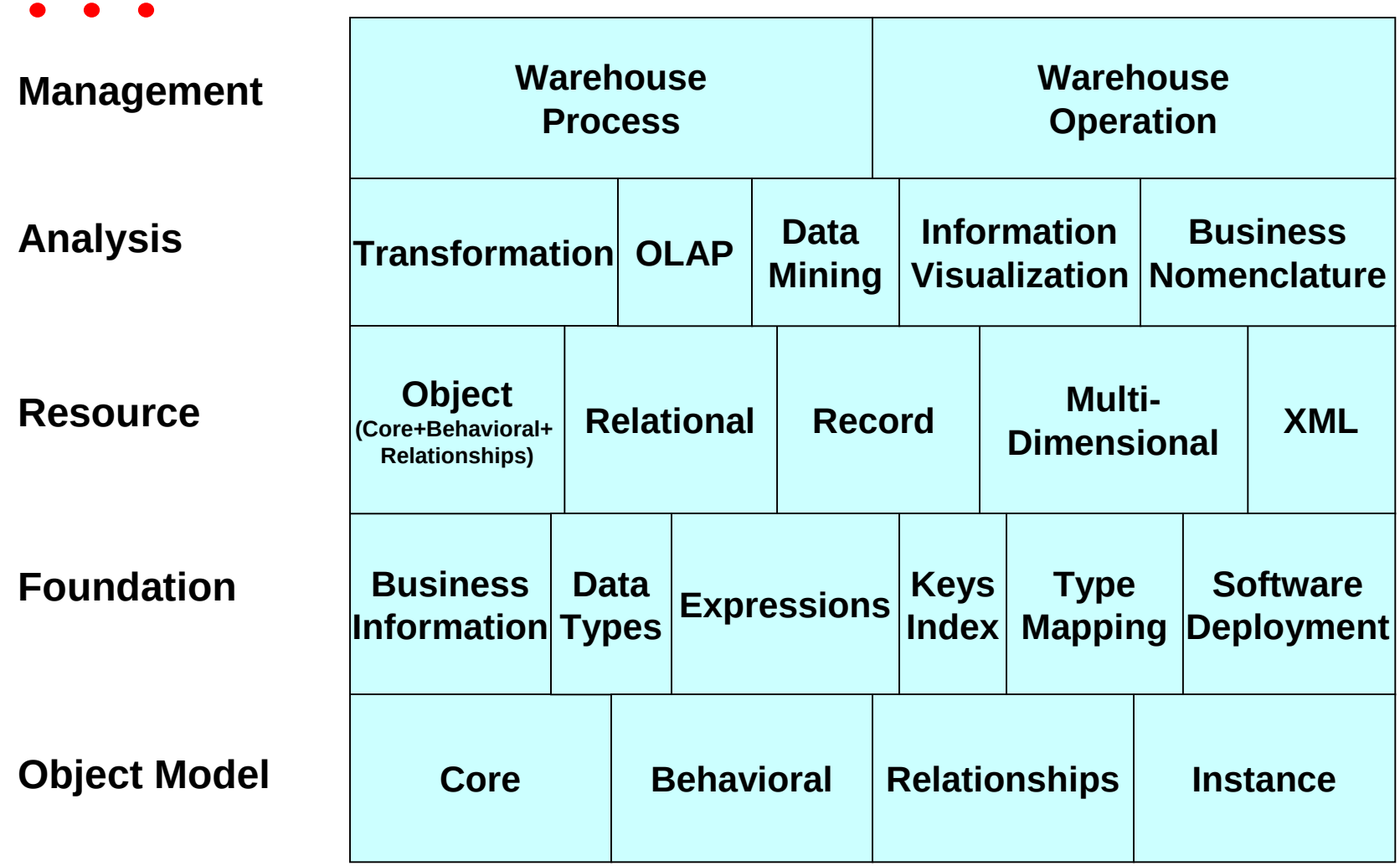
- ✓Modeling Notation: UML
- ✓Metadata Interchange: XMI
- ✓Metadata API:
  - MOF IDL Mapping
  - JMI – MOF/Java Mapping

CWM 1.0 is based on

- ✓UML 1.3
- ✓MOF 1.3
- ✓XMI 1.1



The CWM Metamodel



CWM Statistics

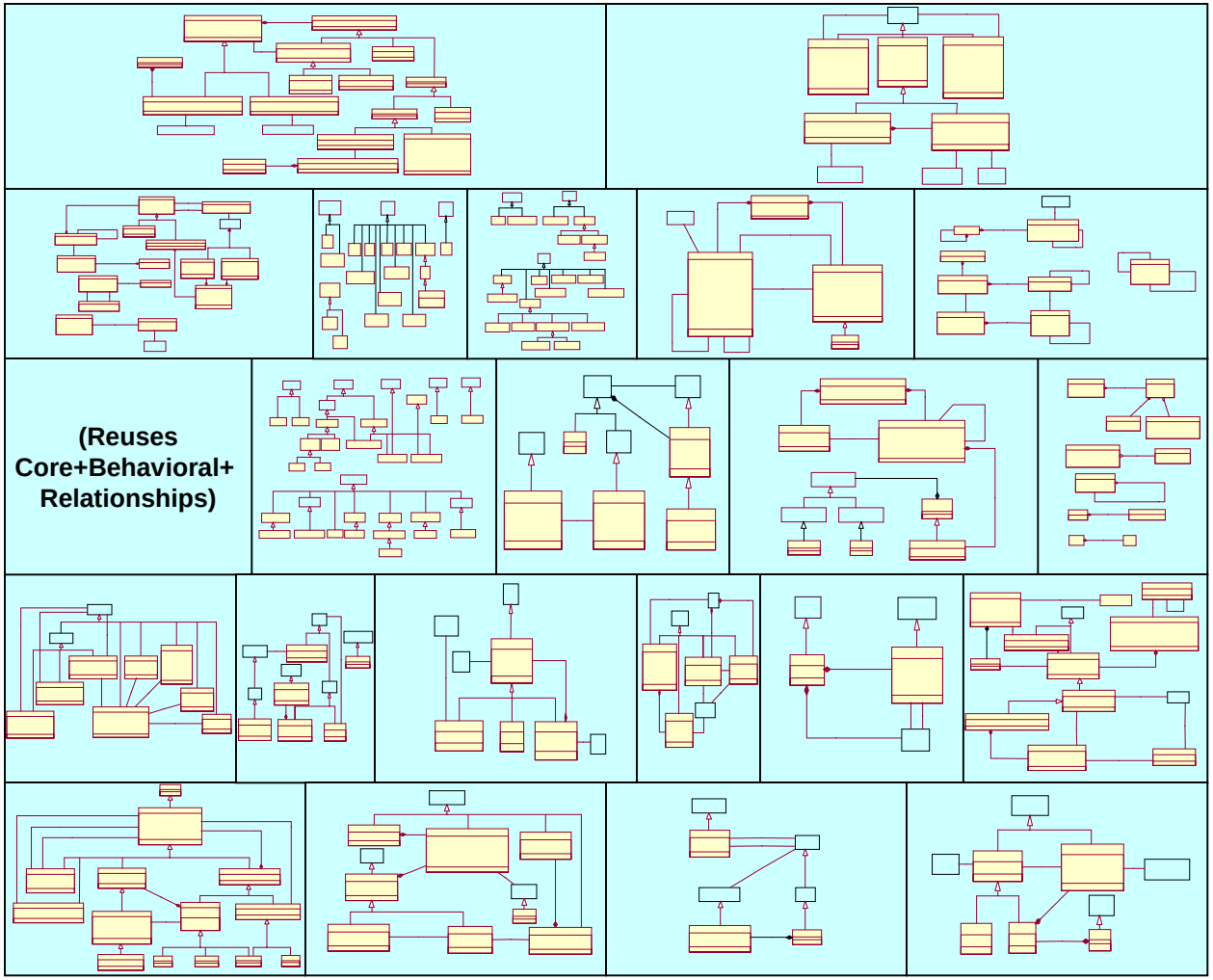
Management

Analysis

Resource

Foundation

Object Model



204 classes & 154 associations in 22 packages

CWM Model Subsets

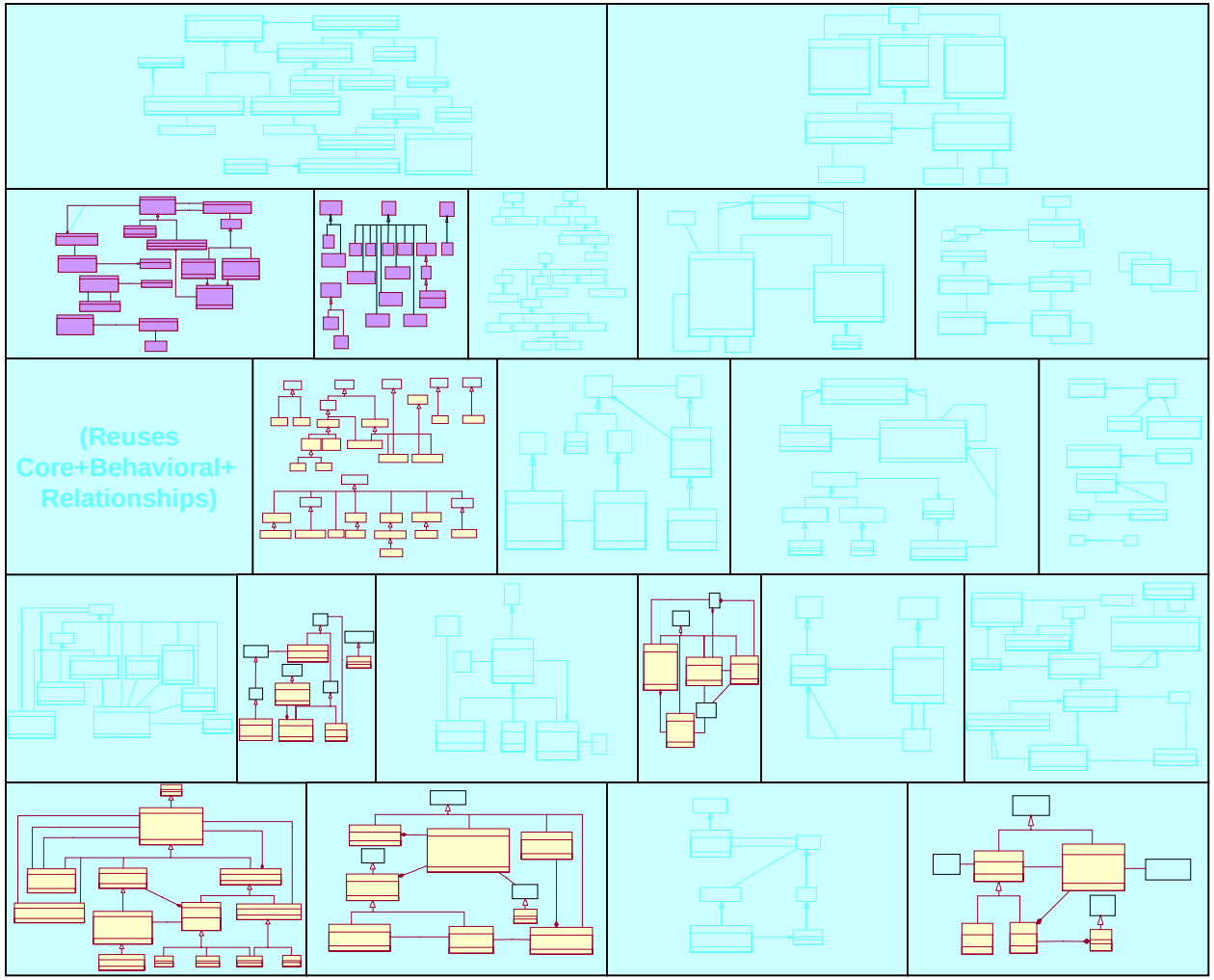
Management

Analysis

Resource

Foundation

Object Model

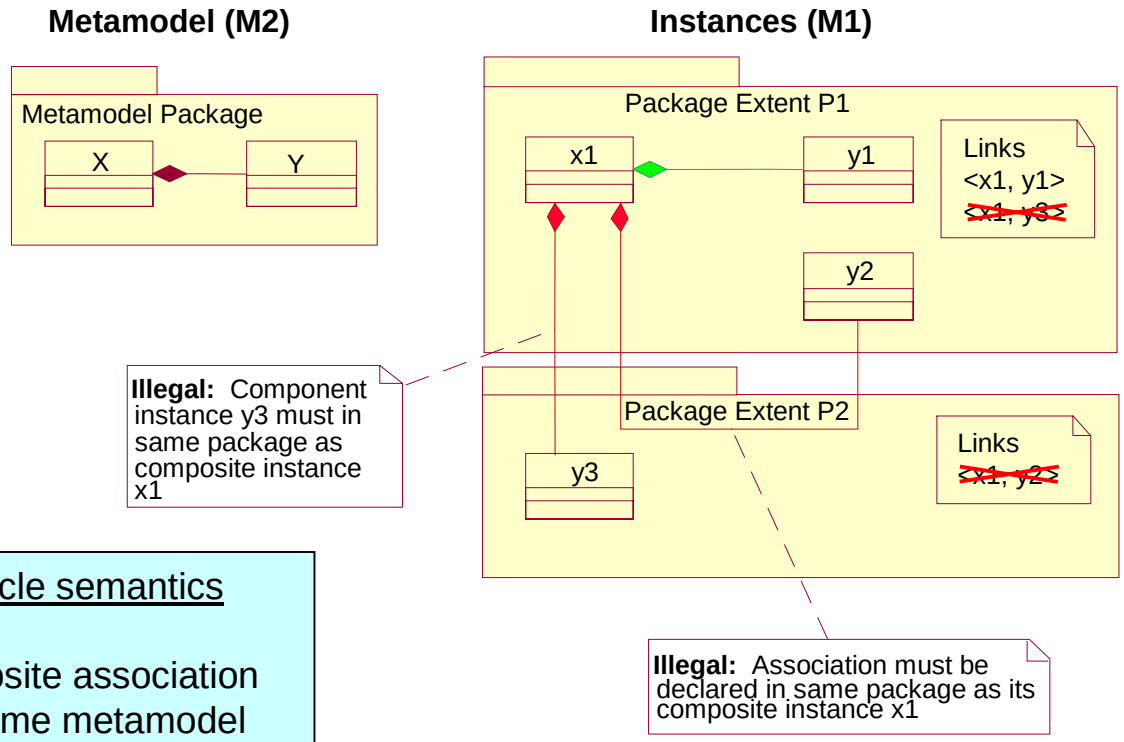


Relational subset : 68 classes & 37 associations in 6 packages

Add-on for Relational => OLAP ETL : 100 classes & 74 associations in 8 packages

Modeling Issues – Package Boundries

- MOF rules affect how associations may cross package boundaries



Composition Closure Rule

The composite and component instances in a composition along with any links that form the composition must all belong to the same outermost Package extent.

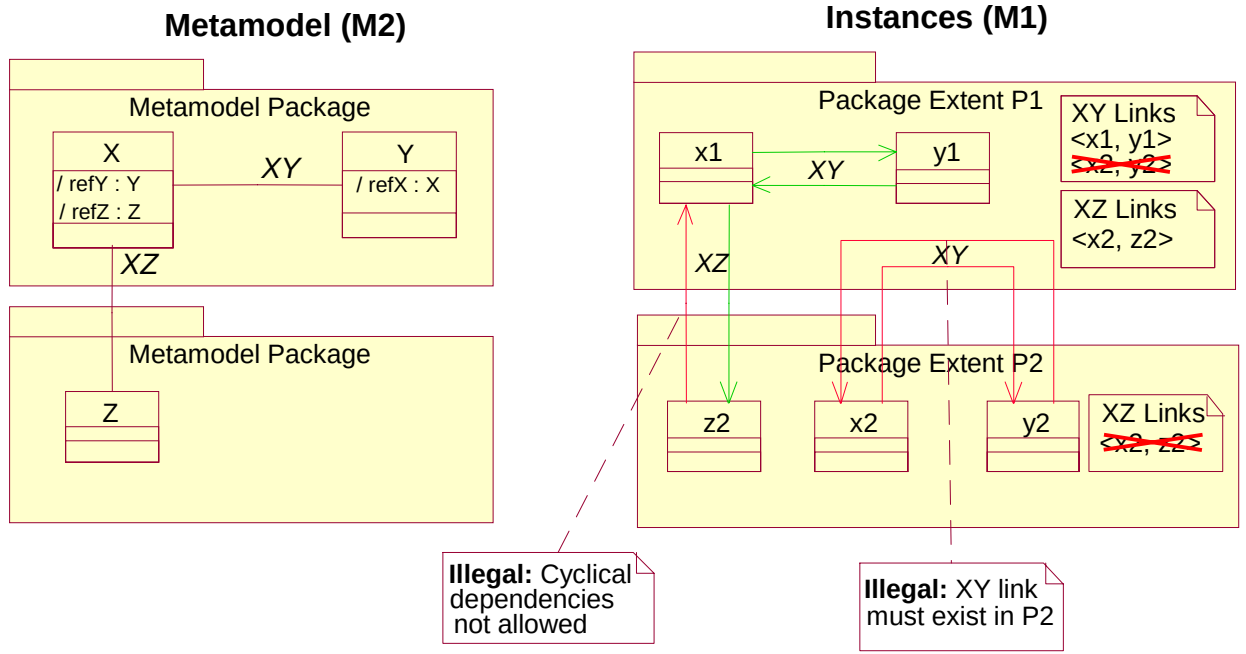
-- MOF v1.3 specification, page 4-19

Effect on CWM: Preserve lifecycle semantics

Classes X and Y and the composite association between them must be in the same metamodel package to preserve cascade delete semantics.

Modeling Issues – Package Boundries

- MOF rules affect how associations may cross package boundaries



Reference Closure Rule

If Class C has a Reference R that exposes an Association End E in an Association A, then it is illegal to cause a link to be constructed such that an instance of C (or a sub-class of C) at the exposed End belongs to a different outermost extent to the A link set containing the link.

-- MOF v1.3 specification, page 4-18

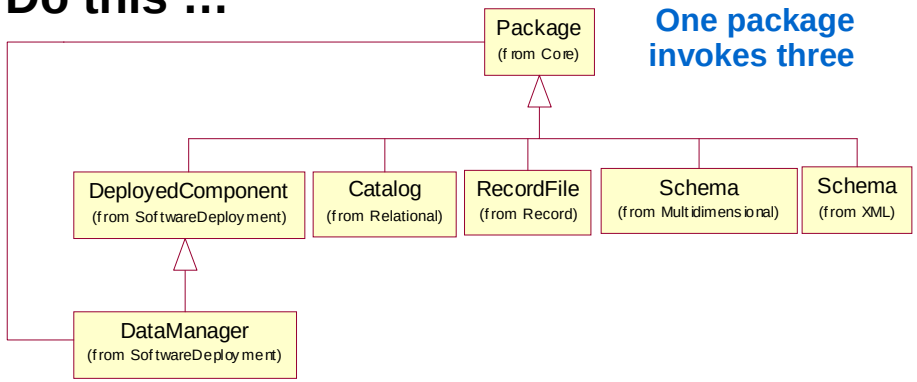
Effect on CWM: No Cyclical Dependencies

Classes X and Y and associations XY and XZ must reside in the same metamodel package. Class Z cannot have a reference to class X.

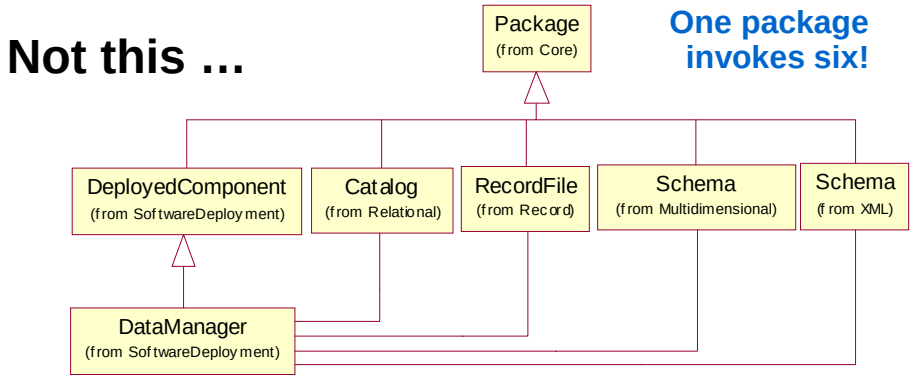
Modeling Issues -- Minimize Package Dependencies

- Reuse inherited associations wherever possible
- Without association generatization, precise semantics may require OCL constraints.

Do this …

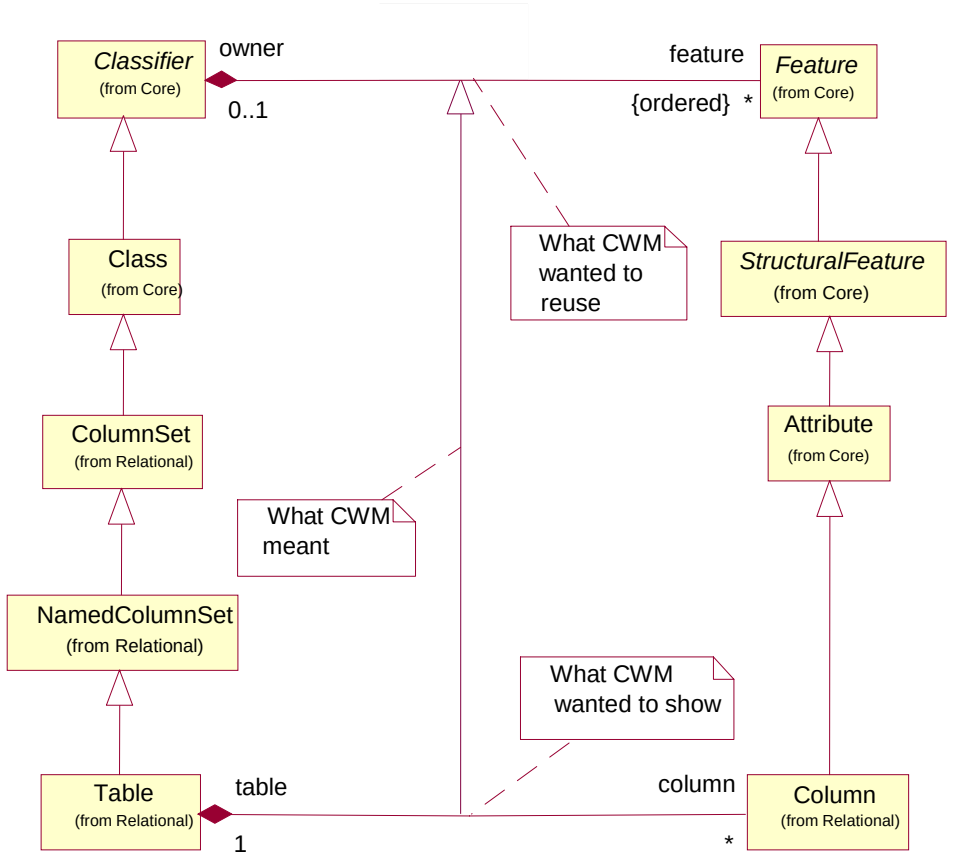


Not this …



Modeling Issues – Associations, References & Reuse

- Association generalization would define association reuse much more precisely
  - Avoids messy, situational OCL statements
- MOF's association end/reference dichotomy is foreign to UML
  - CWM invented notational conventions to handle
  - Notation & semantics for renaming inherited association ends and references is unclear



UML as a Foundation for MDA Implementations

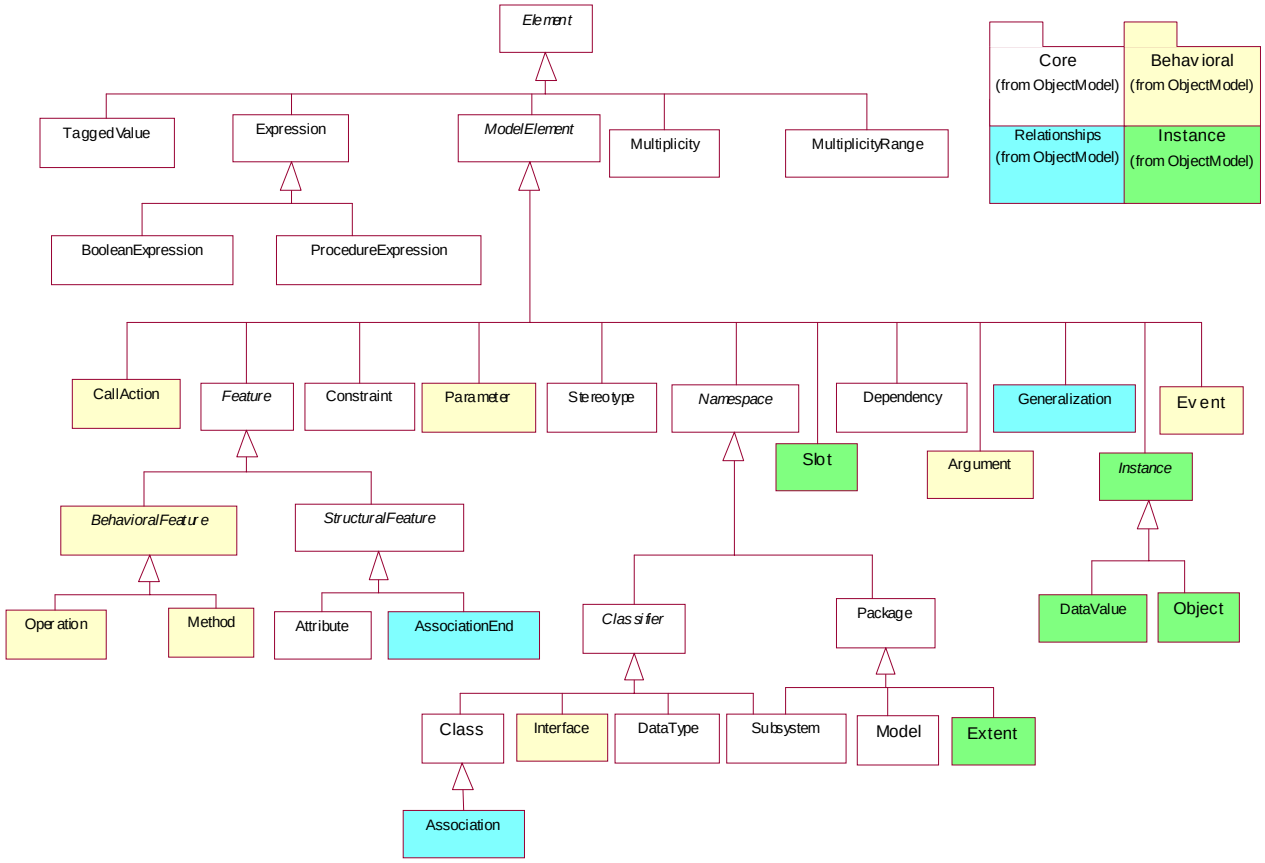
- Design vs. Implementation Orientation
  - UML is design-oriented
    - Notational clarity is paramount
    - Leads to fewer, larger packages
    - Implementation pragmatics absent
  - CWM and MOF are implementation-oriented
    - Many, smaller packages desirable
    - Implementation pragmatics enforced
    - Amenable to interface and repository generation
  - MOF/UML incompatibilities were very troublesome

UML as a Foundation for MDA Implementations

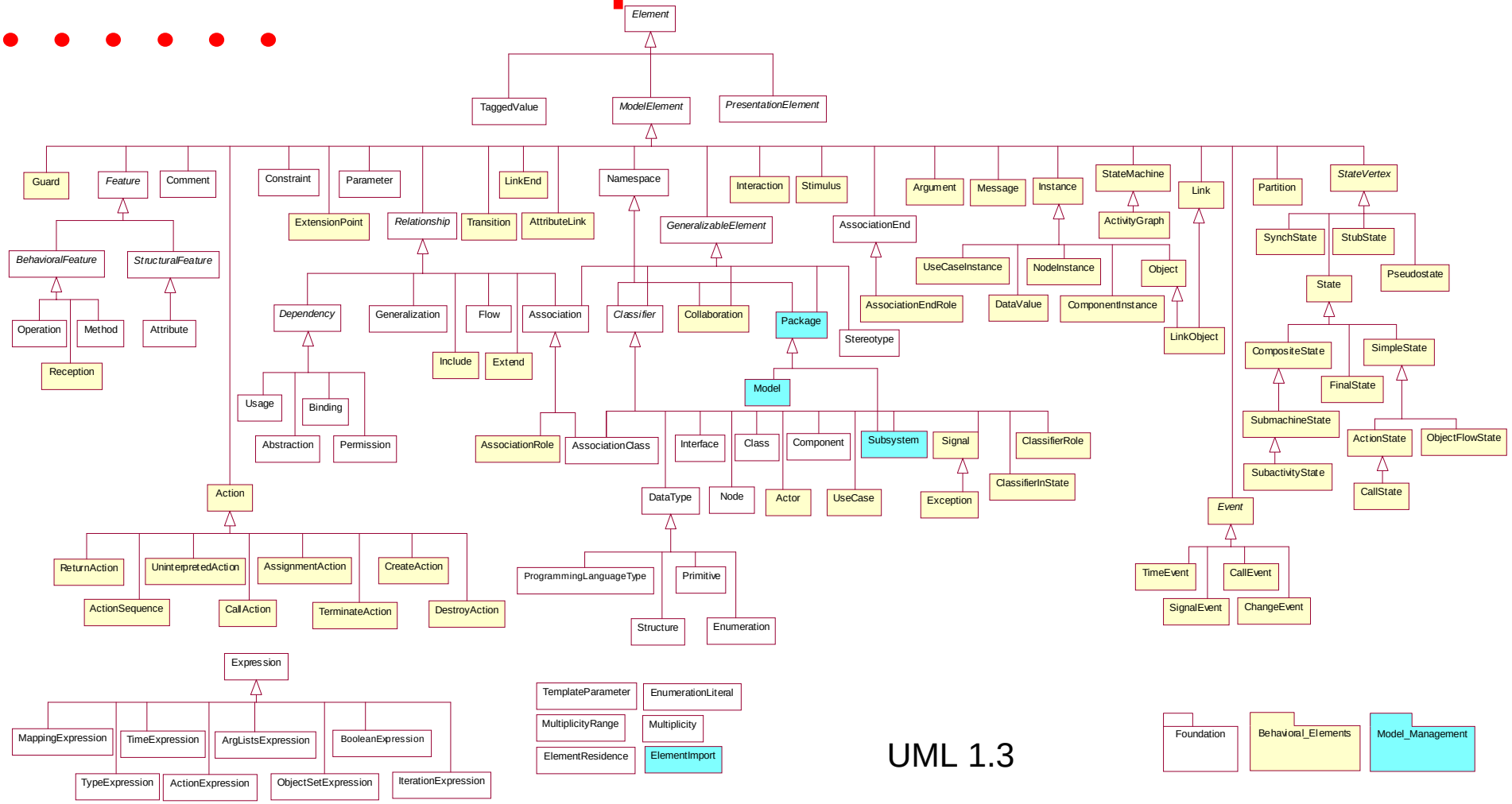
- Package granularity is critical
  - Many, smaller packages preferred
    - Group related concepts into reusable units
    - Minimize dependencies between packages
    - Many obvious parallels in the high modularity of modern programming languages
  - Reduced conceptual clutter
    - UML v1.3 has ca.125 classes
      - Many represent ideas not used by CWM
      - Deleterious impact on generated interfaces
      - Creates need to explain unused concepts to users
    - CWM needed only 37 UML classes
      - But they are from all three UML outermost packages
      - Must invoke all 125 UML classes!

UML as a Foundation for MDA Implementations

- In CWM, UML replaced by ObjectModel
  - Subset of UML needed by CWM
    - Plus a few CWM-specific simplifications
  - 37 classes
  - 25 associations



UML as a Foundation for MDA Implementations



UML 1.3

Suggestion for UML 2

- Identify a common core
  - Reusable by UML, MOF, CWM, and others
  - CWM 2.0 can test usability
- More, smaller packages containing functionally coherent groups of classes
  - Follow MOF rules when crossing package boundaries
- Support association generalization
- Resolve association/reference dichotomy