CWM Metadata Interchange Patterns (MIP) Specification

This OMG document replaces the submission document (ad/02-10-06). It is an OMG Final Adopted Specification and is currently in the finalization phase. Comments on the content of this document are welcomed, and should be directed to *issues@omg.org* by July 21, 2003.

You may view the pending issues for this specification from the OMG revision issues web page *http://cgi.omg.org/issues/*; however, at the time of this writing there were no pending issues.

The FTF Recommendation and Report for this specification will be published on September 19, 2003. If you are reading this after that date, please download the available specification from the OMG Specifications Catalog.

OMG Adopted Specification
ptc/03-10-13

CWM Metadata Interchange Patterns Specification

October 2003
Adopted Specification
ptc/03-10-13

An Adopted Specification of the Object Management Group, Inc.

USE OF SPECIFICATION - TERMS, CONDITIONS & NOTICES

The material in this document details an Object Management Group specification in accordance with the terms, conditions and notices set forth below. This document does not represent a commitment to implement any portion of this specification in any company's products. The information contained in this document is subject to change without notice.

LICENSES

The companies listed above have granted to the Object Management Group, Inc. (OMG) a nonexclusive, royalty-free, paid up, worldwide license to copy and distribute this document and to modify this document and distribute copies of the modified version. Each of the copyright holders listed above has agreed that no person shall be deemed to have infringed the copyright in the included material of any such copyright holder by reason of having used the specification set forth herein or having conformed any computer software to the specification.

Subject to all of the terms and conditions below, the owners of the copyright in this specification hereby grant you a fully-paid up, non-exclusive, nontransferable, perpetual, worldwide license (without the right to sublicense), to use this specification to create and distribute software and special purpose specifications that are based upon this specification, and to use, copy, and distribute this specification as provided under the Copyright Act; provided that: (1) both the copyright notice identified above and this permission notice appear on any copies of this specification; (2) the use of the specifications is for informational purposes and will not be copied or posted on any network computer or broadcast in any media and will not be otherwise resold or transferred for commercial purposes; and (3) no modifications are made to this specification. This limited permission automatically terminates without notice if you breach any of these terms or conditions. Upon termination, you will destroy immediately any copies of the specifications in your possession or control.

PATENTS

The attention of adopters is directed to the possibility that compliance with or adoption of OMG specifications may require use of an invention covered by patent rights. OMG shall not be responsible for identifying patents for which a license may be required by any OMG specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. OMG specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

GENERAL USE RESTRICTIONS

Any unauthorized use of this specification may violate copyright laws, trademark laws, and communications regulations and statutes. This document contains information which is protected by copyright. All Rights Reserved. No part of this work covered by copyright herein may be reproduced or used in any form or by any means--graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems--without permission of the copyright owner.

DISCLAIMER OF WARRANTY

WHILE THIS PUBLICATION IS BELIEVED TO BE ACCURATE, IT IS PROVIDED "AS IS" AND MAY CONTAIN ERRORS OR MISPRINTS. THE OBJECT MANAGEMENT GROUP AND THE COMPANIES LISTED ABOVE MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS PUBLICATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF TITLE OR OWNERSHIP, IMPLIED WARRANTY OF

MERCHANTABILITY OR WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE OR USE.
IN NO EVENT SHALL THE OBJECT MANAGEMENT GROUP OR ANY OF THE COMPANIES LISTED ABOVE BE LIABLE FOR ERRORS CONTAINED HEREIN OR FOR DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL, RELIANCE OR COVER DAMAGES, INCLUDING LOSS OF PROFITS, REVENUE, DATA OR USE, INCURRED BY ANY USER OR ANY THIRD PARTY IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

The entire risk as to the quality and performance of software developed using this specification is borne by you. This disclaimer of warranty constitutes an essential part of the license granted to you to use this specification.

RESTRICTED RIGHTS LEGEND

Use, duplication or disclosure by the U.S. Government is subject to the restrictions set forth in subparagraph (c) (1) (ii) of The Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013 or in subparagraph (c)(1) and (2) of the Commercial Computer Software - Restricted Rights clauses at 48 C.F.R. 52.227-19 or as specified in 48 C.F.R. 227-7202-2 of the DoD F.A.R. Supplement and its successors, or as specified in 48 C.F.R. 12.212 of the Federal Acquisition Regulations and its successors, as applicable. The specification copyright owners are as indicated above and may be contacted through the Object Management Group, 250 First Avenue, Needham, MA 02494, U.S.A.

TRADEMARKS

The OMG Object Management Group Logo®, CORBA®, CORBA Academy®, The Information Brokerage®, XMI® and IIOP® are registered trademarks of the Object Management Group. OMG™, Object Management Group™, CORBA logos™, OMG Interface Definition Language (IDL)™, The Architecture of Choice for a Changing World™, CORBAservices™, CORBAfacilities™, CORBAmed™, CORBAnet™, Integrate 2002™, Middleware That's Everywhere™, UML™, Unified Modeling Language™, The UML Cube logo™, MOF™, CWM™, The CWM Logo™, Model Driven Architecture™, Model Driven Architecture Logos™, MDA™, OMG Model Driven Architecture™, OMG MDA™ and the XMI Logo™ are trademarks of the Object Management Group. All other products or company names mentioned are used for identification purposes only, and may be trademarks of their respective owners.

COMPLIANCE

The copyright holders listed above acknowledge that the Object Management Group (acting itself or through its designees) is and shall at all times be the sole entity that may authorize developers, suppliers and sellers of computer software to use certification marks, trademarks or other special designations to indicate compliance with these materials.

Software developed under the terms of this license may claim compliance or conformance with this specification if and only if the software compliance is of a nature fully matching the applicable compliance points as stated in the specification. Software developed only partially matching the applicable compliance points may claim only that the software was based on this specification, but may not claim compliance or conformance with this specification. In the event that testing suites are implemented or approved by Object Management Group, Inc., software developed using this specification may claim compliance or conformance with the specification only if the software satisfactorily completes the testing suites.

ISSUE REPORTING

All OMG specifications are subject to continuous review and improvement. As part of this process we encourage readers to report any ambiguities, inconsistencies, or inaccuracies they may find by completing the Issue Reporting Form listed on the main web page http://www.omg.org, under Documents & Specifications, Report a Bug/Issue.

Contents

Preface

About the Object Management Group

The Object Management Group, Inc. (OMG) is an international organization supported by over 600 members, including information system vendors, software developers and users. Founded in 1989, the OMG promotes the theory and practice of object-oriented technology in software development. The organization's charter includes the establishment of industry guidelines and object management specifications to provide a common framework for application development. Primary goals are the reusability, portability, and interoperability of object-based software in distributed, heterogeneous environments. Conformance to these specifications will make it possible to develop a heterogeneous applications environment across all major hardware platforms and operating systems.

OMG's objectives are to foster the growth of object technology and influence its direction by establishing the Object Management Architecture (OMA). The OMA provides the conceptual infrastructure upon which all OMG specifications are based.

What is CORBA?

The Common Object Request Broker Architecture (CORBA), is the Object Management Group's answer to the need for interoperability among the rapidly proliferating number of hardware and software products available today. Simply stated, CORBA allows applications to communicate with one another no matter where they are located or who has designed them. CORBA 1.1 was introduced in 1991 by Object Management Group (OMG) and defined the Interface Definition Language (IDL) and the Application Programming Interfaces (API) that enable client/server object interaction within a specific implementation of an Object Request Broker (ORB). CORBA 2.0, adopted in December of 1994, defines true interoperability by specifying how ORBs from different vendors can interoperate.

OMG Documents

The OMG Specifications Catalog is available from the OMG website at:

http://www.omg.org/technology/documents/spec_catalog.htm

The OMG documentation is organized as follows:

OMG Modeling Specifications

Includes the UML, MOF, XMI, and CWM specifications.

OMG Middleware Specifications

Includes CORBA/IIOP, IDL/Language Mappings, Specialized CORBA specifications, and CORBA Component Model (CCM).

Platform Specific Model and Interface Specifications

Includes CORBAservices, CORBAfacilities, OMG Domain specifications, OMG Embedded Intelligence specifications, and OMG Security specifications.

Obtaining OMG Documents

The OMG collects information for each book in the documentation set by issuing Requests for Information, Requests for Proposals, and Requests for Comment and, with its membership, evaluating the responses. Specifications are adopted as standards only when representatives of the OMG membership accept them as such by vote. (The policies and procedures of the OMG are described in detail in the *Object Management Architecture Guide*.)

OMG formal documents are available from our web site in PostScript and PDF format. Contact the Object Management Group, Inc. at:

OMG Headquarters

250 First Avenue

Needham, MA 02494

USA

Tel: +1-781-444-0404

Fax: +1-781-444-0320

pubs@omg.org

http://www.omg.org

Typographical Conventions

The type styles shown below are used in this document to distinguish programming statements from ordinary English. However, these conventions are not used in tables or section headings where no distinction is necessary.

Helvetica bold - OMG Interface Definition Language (OMG IDL) and syntax elements.

`Courier bold` - Programming language elements.

Helvetica - Exceptions

Acknowledgments

The following companies submitted and/or supported the CWM Metadata Interchange Patterns specification:

- Hyperion Solutions Corporation
- IBM Corporation
- Oracle Corporation
- Unisys Corporation

Introduction 1

Contents

This chapter includes the following topics.

1.1 Purpose

The purpose of the Common Warehouse Metamodel specification is to define a common interchange specification for metadata in a data warehouse. This definition provides a common language and metamodel definitions for the objects in the data warehouse. CWM describes a format to interchange metadata, but lacks the knowledge to describe any particular type of interchange. The need to define the context of a CWM interchange was discovered when the CWM co-submitting companies produced the CWM Interoperability Showcase. In order to make an effective demonstration of CWM technology, the participants needed to agree upon the set of metadata to be interchanged.

The purpose of this specification (CWM Metadata Interchange Patterns, or CWM MIP) is to add a semantic context to the interchange of metadata in terms of recognized sets of objects or object patterns. We will introduce the term “Unit of Interchange” (UOI) to define a valid, recognizable CWM interchange. From this information, a user of CWM, working in conjunction with CWM MIP, should be able to produce truly interoperable tools.

CWM MIP augments the current CWM metamodel definitions by adding a new metamodel package. This new metamodel will provide the structural framework to identify both a UOI and an associated model of a pattern, and providing the necessary object definitions to describe both.

1.2 Design Rationale

1.2.1 Design Overview

The goal of this specification is to leverage the existing CWM as much as possible. This specification extends CWM and conforms to the design rationale as outlined in the CWM specification, Design Rationale section.

1.2.2 The Relationship between CWM MIP and CWM

CWM has been adopted as OMG's standard for representing data warehouse metadata. The CWM MIP metamodel extends the CWM standard without altering any existing CWM implementations. This allows users of CWM MIP to use the data warehouse models defined by CWM and add the semantics of how to produce a UOI without an intrusion into the CWM definition.

CWM MIP Specification 2

Contents

This chapter includes the following topics.

2.1 Introduction

The CWM metamodels contain a robust common object definition of concepts found in many datawarehouse tools. These common object definitions can be interchanged between tools to provide some level of interoperability. Through the production of the CWM Interoperability Showcase the CWM Team discovered that without some prior knowledge as to how the interchange document was produced seamless interoperability between the respective tools was not achieved. The key piece of information that was missing was what was the context of the interchange. In general, the entire document needed to be read and processed before the tool could then try to determine if this interchange was possible.

CWM covers a large number of subject areas in the datawarehouse space. It is unlikely that any individual tool will cover the entire CWM model. By identifying and cataloging inter-related concepts that can be interchanged together, tools can code for smaller domain specific portions of CWM. We call the identification of these inter-related CWM concepts *CWM metadata interchange patterns*.

2.2 Applying Pattern Concepts to the Metadata Interchange Problem

Definition: A *Software Design Pattern* is a description of communicating objects and classes that are customized to solve a general design problem within a particular context (Gamma et al, 1995).

The traditional "gang of four (GOF)" design patterns helped to standardize the usage of object-oriented software and components in solving typical, recurring, programming problems. Software design patterns may be formally modeled in the UML by providing generic class models along with associated, sample collaborations; i.e., *parameterized collaborations* (see Rumbaugh et al, 1999, pp. 387-388, Booch et al, 1999, pp. 381-392, and Sunye et al, 2000).

The formalization of Metadata Interchange Patterns, as put forth in this specification, is highly analogous to software design patterns, borrowing GOF terminology and introducing metadata structuring concepts that have direct analogues in the software patterns world.

Definition: A *CWM Metadata Interchange Pattern* is an identified subset of the CWM metamodel, optionally with constraints on the instances of the metamodel subset.

The subset traverses one or more CWM metamodels, and defines the *metamodel subspace* of the solution. The constraints (if defined) establish *boundaries* on the solution subspace. Note that these constraints are components of the pattern itself, and are not to be confused with the constraints defined by the CWM metamodel proper.

Defining metadata intechange patterns in this manner is done for two purposes:

1. By defining these patterns, we constrain the universe of all possible combinations of instances of CWM model elements (which is a countably infinite set, by virtue of the fact that there are many unbounded association ends used throughout the metamodel).

2. This constrained set of metadata patterns should also generally reflect the more common metadata interactions that take place within the data warehousing and business intelligence/analysis domains. Thus, we are not only attempting to overcome combinatorial explosions by restricting the model search space in some formal manner, but we are also ensuring that this restricting subspace contains elements that are semantically meaningful to the application domain.

The main benefit provided by metadata interchange patterns, of course, is that they greatly enhance interchange and interoperability by enabling software tools to focus on only those metadata patterns that are meaningful to what a given tool expects from the

metadata it imports. This greatly increases the probability that metadata interchange attempts occuring within a particular DW/BI context will be successful. It also greatly simplifies tool design.

Note that the metadata interchange solution specification, as described above, is totally non-intrusive to CWM; it consists only of identifying and constraining subsets of the CWM metamodel. It does not require any modification of the CWM metamodel.

2.3 The CWM MIP Model

The CWM metadata interchange pattern approach described previously may be realized though the use of another metamodel that (non-intrusively) extends certain CWM model elements, but is otherwise “parallel” to the rest of the CWM model structure.

To formally model a CWM interchange pattern specification, first note that the identification of inter-related concepts in CWM (i.e., the *solution subspace*) is really the identification of inter-related CWM packages and their contents. The new metamodel must allow users to catalog the characteristics of the types of units of interchange they intend to produce. This model must also capture several key details about how the interchange will be performed. The CWM MIP Model is the proposed, formal model of a CWM metadata interchange pattern, and is illustrated in Figure 2-1.

Note that the CWM MIP Model, as a formalization of a CWM metadata interchange pattern, is roughly equivalent to the specification of a *parameterized collaboration* in the UML modeling of GOF-style software design patterns, and the CWM Metamodel itself is analogous to the class diagrams that are generally used in defining the domains of GOF-style patterns.

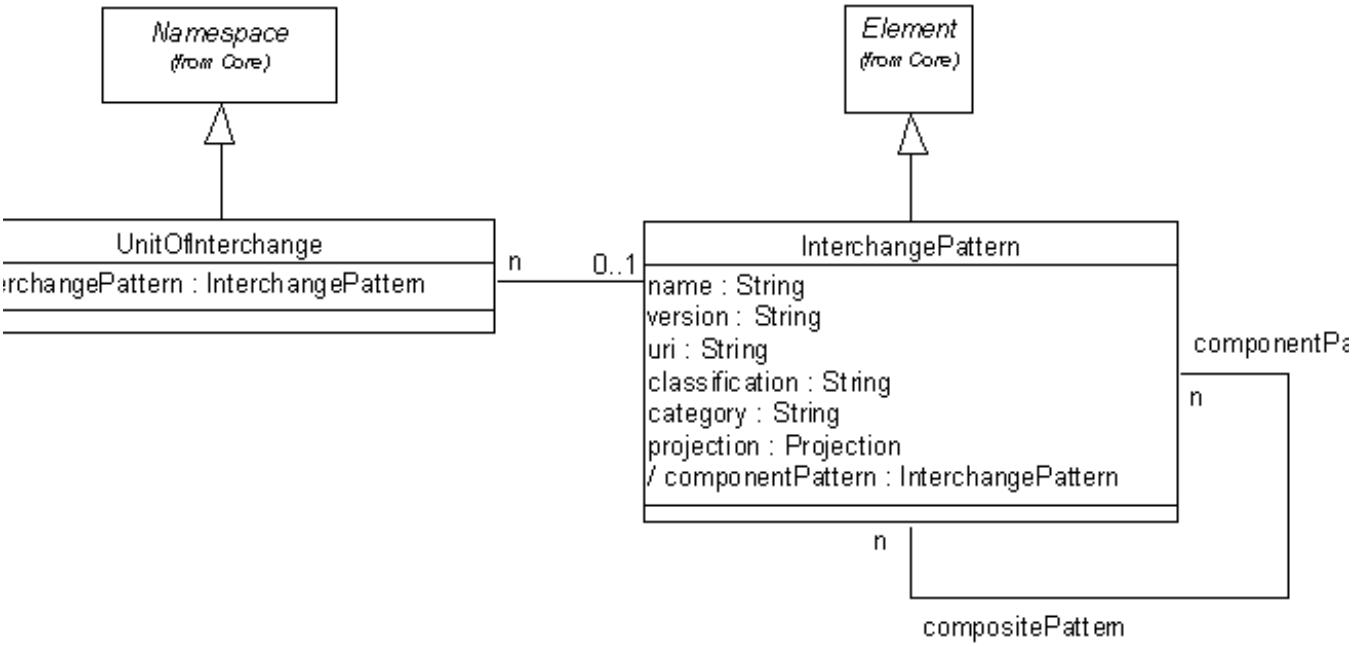

Figure 2-1 CWM MIP Model: Main classes

There are two primary classes: *UnitOfInterchange*, a subclass of CWM *Namespace* that contains the complete CWM instance (model/metadata) being interchanged, and *InterchangePattern*, which formally describes the pattern that the content of UnitOfInterchange conforms to.

InterchangePattern also contains a reflexive association to model pattern composition, in which patterns are comprised from other pattern definitions. The attributes of InterchangePattern are as follows:

Name: The name of the interchange pattern.

Version: The version of the interchange pattern.

URI : A URI identifying a human-readable pattern specification document that describes the interchange pattern.

Classification: The structural classification of the pattern: Micro Pattern, Domain Pattern, Macro Pattern (these are defined in Section 2.6, “Pattern Specification and Cataloging,” on page 2-11).

Category: The usage category of the pattern: Interchange, Mapping, Typing, Extension, Interpretation, Generation, Structural (these are defined in Section 2.6, “Pattern Specification and Cataloging,” on page 2-11).

Projection: An object-valued array of type Projection (see the definition of the Projection class below) that describes the metamodel graph subset defining the semantic context of the pattern. If this attribute contains more than one value, then the complete projection of the pattern is the graph union of multiple projection object values. This attribute has a multiplicity of 1..*.

/ComponentPattern: A reference to the collection of component patterns that this pattern is comprised of (if any). This reference is a derived attribute based on the componentPattern end of the reflexive association.

shows the remaining classes of the model. Note that these classes are used exclusively to define the types of certain object-valued attributes in other parts of the model.

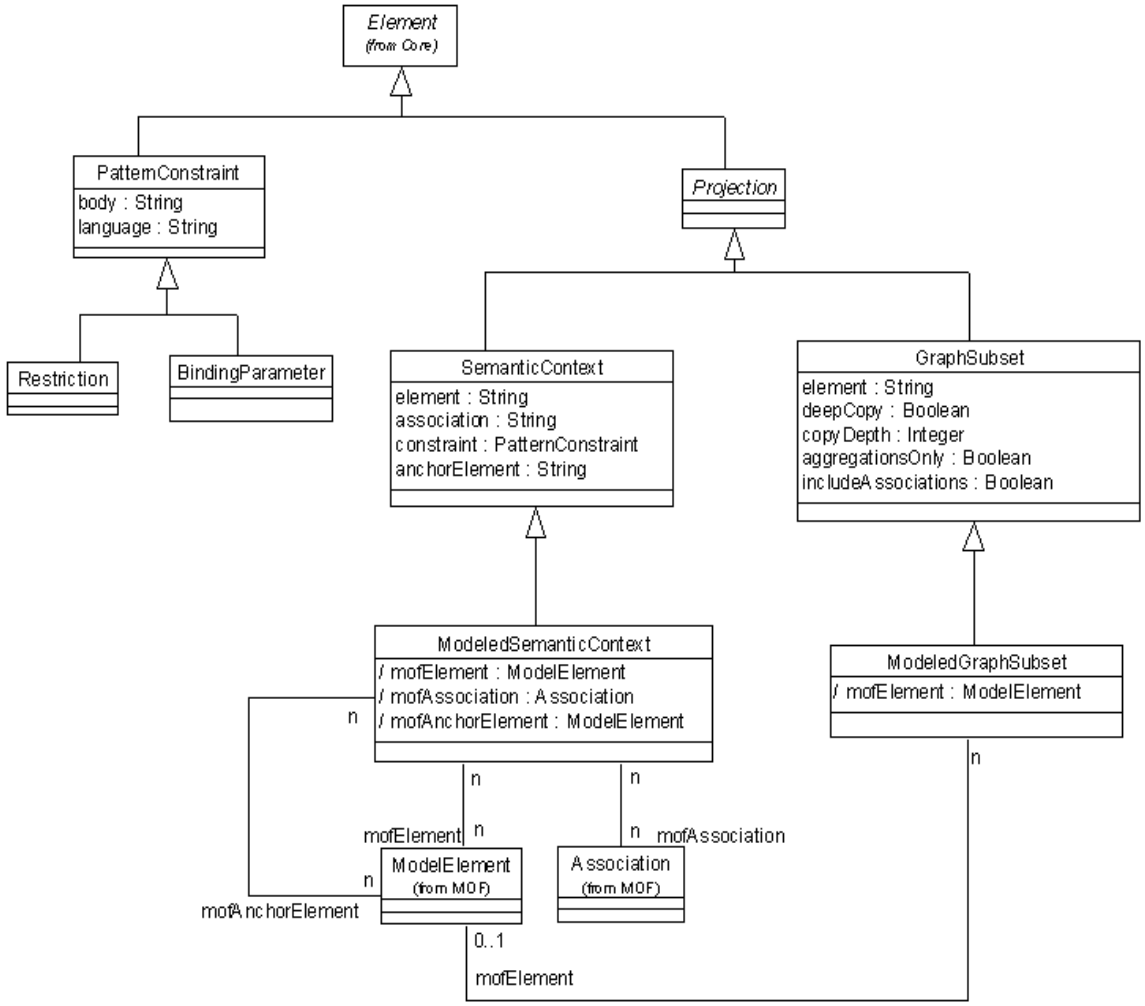

Figure 2-2 CWM MIP Model: Types

PatternConstraint is used to define constraints against the metamodel Projection (described below). There is no requirement to use any particular constraint language. However, if OCL is used, the OCL expression contained by PatternConstraint must be a valid OCL expression relative to the modeling context of the Projection.

Projection is an abstract base class that models the semantic context of the pattern, where by semantic context we mean a well-defined subset of the CWM metamodel graph. Projection has two concrete subclasses: *SemanticContext* and *GraphSubset*. SemanticContext specifies a projection by naming the CWM model elements and

associations comprising the projection, along with any constraints on those elements. Note that this is the formal projection model as described in Section 2.6, “Pattern Specification and Cataloging,” on page 2-11).

GraphSubset, on the other hand, allows the projection to be specified by an expression which, when evaluated, describes some physical sub-graph of the CWM UML model.

In very general terms, SemanticContext allows for a very finely grained, but explicitly enumerated, projection, whereas GraphSubset allows for a course-grained, but more efficiently specified, projection.

The attributes of the SemanticContext class are as follows:

Element: Names of the CWM metamodel classes comprising the projection, based on logical names expressed as strings. This attribute has a multiplicity of 0..*. These logical names may refer to elements from multiple CWM metamodel packages. For example, a value for this attribute might consist of the following collection of the strings:

“org.omg.cwm.analysis.olap.Dimension”

"org.omg.cwm.objectmodel.core.Attribute"

Association: Names of the CWM associations comprising the projection, based on logical names expressed as strings. This attribute has a multiplicity of 0..*. These logical names may refer to associations from any number of CWM packages, including associations that span packages. For example, a value for this attribute might consist of the string:

"org.omg.cwm.objectmodel.core.ClassifierFeature"

Constraint: A collection of pattern constraints that apply to the elements and associations comprising the projection. This attribute has a multiplicity of 0..*. For example, a value for this attribute might consist of the OCL expression:

"context Dimension inv: self.feature->size => 1 and self.feature->exists(f | f.oclType(Attribute) and f.oclAsType(Attribute).ModelElement::name = "key")->size = 1"

There are two kinds of constraints that might be applied against the projection: *Restrictions* and *Binding Parameters* (or simply *Parameters*). In general, a restriction imposes structural constraints on an instance of a projection (e.g., by forcing an upper bound on an otherwise unbounded association end). The resulting structure is then regarded as being invariant with regard to other values that might be supplied in the construction of a pattern instance (e.g., values of certain class attributes). An example of a restriction on structure is the requirement stated above that every instance of Dimension must have at least one related Attribute. Parameters, on the other hand, typically assert values (or value ranges) for class attributes. Parameters do not influence structure, but influence content instead.

An example of a parameter is the requirement stated above that precisely one of a Dimension’s Attributes must carry a name value of “key.” A given pattern definition does not necessarily have to specify value constraints for all class attributes, in which case, those class attributes are said to be free. It is up to the process constructing the

pattern instance to decide what values to populate free class attributes with (or to not supply any values). Note that a parameter might also specify that some class attribute must not be without a value (e.g., must be non-blank or non-zero) without specifying precisely what that value is to be.

In many cases, distinctions between restrictions and parameters may sometimes become somewhat blurred. In order to differentiate between the two types of constraint when necessary, the model supplies two subclasses for PatternConstraint: Restriction and BindingParameter. These subclasses are provided for naming/typing purposes only. A pattern definition may choose to distinguish between either type of constraint, or may choose to simply blend constraints together by using the PatternConstraint superclass rather than either of the two subclasses.

One of the main reasons why one might use the Restriction and BindingParameter subclasses is to provide a trace-back to the non-model-based pattern specification template prescribed by the CWM Developer's Guide (John Wiley & Sons, 2002 -- forthcoming). This template differentiates between restrictions and parameters as part of the pattern formulation process.

AnchorElement: Names CWM metamodel elements from the projection that have some distinguished meaning. Generally, these elements are thought of as root objects of the metamodel projection graph and are used as starting points for navigating the graph. This attribute has a multiplicity of 0..*. Although a projection may have multiple anchor elements, in practice, most projections will usually have a single anchor point. For example, a value for this attribute might consist of the string:

"org.omg.cwm.analysis.olap.Dimension"

The attributes of the GraphSubset class are as follows:

Element: The logical name of a CWM metamodel element (usually a package or class) comprising the physical projection For example, a value for this attribute might consist of the following string:

“org.omg.cwm.resource.relational.Schema”

DeepCopy: A Boolean attribute which, when true, implies that all connected elements and their attributes, within the boundaries of the specified package, are to be included in the pattern projection.

CopyDepth: An integer value that specifies the number of graph edges to traverse when establishing the physical graph projection, in cases when deepCopy is false.

AggregationsOnly: A Boolean attribute which, when true, specifies that only composite elements and their components are to be included in the physical graph projection.

IncludeAssociations: A Boolean attribute which, when true, specifies that associations are to be included in the physical graph projection.

Finally, two additional classes are defined as a means of providing explicitly modeled pattern definitions; that is, pattern definitions composed of MOF metaclass instances (or, MOF metaobjects), rather than the use of logical metamodel class names. These classes consist of *ModeledSemanticContext* and *ModeledGraphSubset*.

ModeledSemanticContext is a subclass of SemanticContext that adds references to instances of MOF:ModelElement and MOF:Association, as well as an association to MOF:ModelElement as a means of specifying anchor classes. Constraints may be specified within ModeledSemanticContext via the inherited SemanticContext::constraint attribute.

ModeledGraphSubset is a subclass of GraphSubset that adds a reference to a single instance of MOF:ModelElement. Like ModeledSemanticContext, it similarly inherits all of its other useful attributes from its base class.

It is expected that, in general, developers of the CWM Metadata Interchange Patterns models that wish to build their solutions around full MOF implementations and object-rich modeling structures will implement and use ModeledSemanticContext and ModeledGraphSubset. On the other hand, developers who wish to construct lighter-weight implementations will simply use the SemanticContext and GraphSubset base classes directly, since these base classes rely on logical name references to CWM metamodel elements, rather than direct physical references to instantiated MOF metaclasses.

2.4 Sample Interchange Patterns

We now present two simple examples of both the specification of a metadata interchange pattern and an instance realizing the pattern. The first example illustrates a projection based on the SemanticContext class, the second illustrates a projection based on the GraphSubset class.

For the first example, note that the various example values provided previously in the description of the SemanticContext class collectively define a simple pattern. The projection of the pattern consists of the subset of the CWM metamodel graph containing the metaclasses Dimension and Attribute, and the inherited ClassifierFeature composition relating them. There must be at least one Attribute on the Dimension and precisely one of these Attributes must have the name “key.” And within the pattern as a whole, Dimension serves as the anchor element. Any navigation

of an instance of this pattern should always start with some instance of Dimension, rather than with a dimensional attribute. A valid instance (or *realization*) of this simple pattern is illustrated in Figure 2-3.

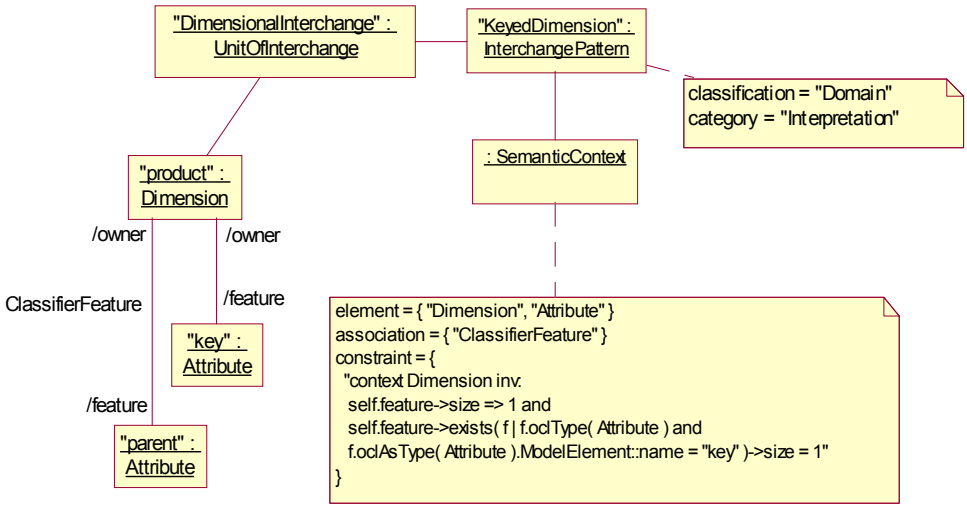

Figure 2-3 Sample Instance Diagram of a Possible Dimensional Metadata Interchange

Figure 2-4 illustrates a sample metadata pattern expressed using the CWM MIP metamodel, with the pattern projection modeled using the GraphSubset class.

In this example, a user wishes to interchange a specific relational schema. This pattern specifies that all objects owned by the relational schema will be included in the transfer. The pattern also indicates that this is not a deepTransfer. This means that we will not traverse associations of object aggregated by the relational schema.

The second part of the pattern states that the type system used by the relational schema must accompany the transfer. Notice that the aggregationsOnly attribute is false. This is because in many cases, by identifying the name of the type system a user could make sense of the transfer.

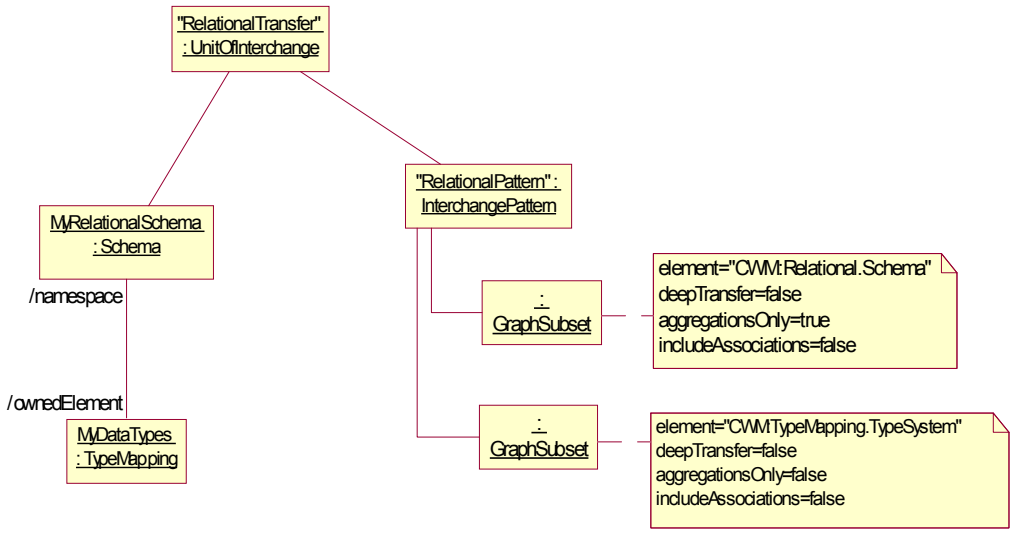

Figure 2-4 Sample Instance Diagram of a Possible Relational Metadata Interchange

2.5 Conformance Points

An implementation of CWM MIP must minimally provide the InterchangePattern class of Figure 2-1, and the following classes of Figure 2-2: PatternConstraint, Restriction, BindingParameter, Projection, Semantic Context, and GraphSubset.

This means that implementations of CWM MIP are allowed to interchange pattern models only, without necessarily being required to also support the exchange of related pattern instances.

2.5.1 Optional Conformance Points

An implementation of CWM MIP may optionally provide the UnitOfInterchange class of Figure 2-1, and/or the ModeledSemanticContext and ModeledGraphSubset classes of Figure 2-2.

2.6 Pattern Specification and Cataloging

The CWM MIP specification recommends, but does not prescribe, that users of CWM who take the patterns-based approach to metadata interchange document their patterns according to the following, general, documentation structure (or template):

Name: The name of the pattern (this is the value of the name attribute of an instance of any UnitOfTransfer that realizes this pattern).

Classification: Identifies the structural classification for the pattern. Possible classifications include (but are not limited to) the following:

- *Macro pattern*: A course-grained pattern that represents an overall organizational framework for interchanged metadata.
- *Domain pattern*: A medium-grained pattern that represents some domain-specific context. In general the notion of providing context in meta data interchange is largely realized by specifying and combining various domain patterns.
- *Micro pattern*: A fine-grained pattern that represents some frequently occurring way of organizing basic metadata structures. Higher-level patterns, such as domain patterns, are generally comprised of many different micro patterns.

Category: Identifies a general category that characterizes the usage of the pattern. Possible categories might include (but are not limited to) the following:

- *Interchange*: Any patterns that generally facilitate interchange (e.g., the CWM MIP Model can be viewed as a meta-pattern for facilitating interchange).
- *Mapping*: Any pattern used for establishing mappings/correspondences between model elements.
- *Typing*: Any pattern used to further classify model elements in some fashion.
- *Extension*: Any pattern that extends or expands the semantics or structure of model instances.
- *Interpretation*: Any pattern that aids in the interpretation of the semantics of some model instance.
- *Generation*: Any pattern that facilitates the automated generation of an implementation of some form.
- *Structural* or *Constructive*: Any pattern that facilitates the construction of more complex structures from simpler ones.

Intent: Statement of purpose/objective of the pattern.

Motivation: Detailed description of the business requirements driving the pattern. Generally amounts to a description of some common data warehousing / business analysis scenario and it requirements for metadata interchange/interoperability.

Also Known As: Provides any other pattern names that used as synonyms of this pattern name.

Projection: A description of the M2-level subset of the CWM metamodel used to establish a common context for metadata interchange. Note that the Projection class of the CWM MIP model and its various subclasses are used for formally modeling such projections.

Restriction(s): A description of M2-level constraints used to restrict or limit the extent of instances of the projection. Generally, restrictions result in structural boundaries placed on instances. Note that the Constraints attribute of SemanticProjection is used to formally express restrictions.

Parameter(s): A set of formal parameters that are used to realize an instance of a pattern. Generally, parameters specify or constrain values for class variables. Note that the Constraints attribute of SemanticProjection is used to formally express parameters.

Consequences: Any consequence (positive or negative) that users of this pattern need to be aware of.

Related Patterns: Any other existing patterns that might be used closely in conjunction with this pattern to achieve some broader, metadata interchange or interoperability goal.

Example: Any sample instance diagrams, programmatic code, or XMI fragments that enable developers to better understand how an instance of this pattern is constructed.

Once again, the above recommendations are just that: Recommendations for documenting patterns and pattern usage. They are not prescribed by this specification. The CWM MIP Model is the only prescriptive aspect of this specification. It is anticipated that some de facto means of communicating useful CWM metadata interchange patterns between members of the CWM user community will eventually evolve over time, much as it has within the software patterns community.

References A

A.1 List of References

(Gamma et al, 1995) Gamma, Erich, Richard Helm, Ralph Johnson, and John Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison Wesley Longman, Inc., Reading, MA, 1995.

(Rumbaugh et al, 1999) Rumbaugh, James, Ivar Jacobson, and Grady Booch, *The Unified Modeling Language Reference Manual*, Addison Wesley Longman, Inc., Reading, MA, 1999.

(Booch et al, 1999) Booch, Grady, James Rumbaugh, and Ivar Jacobson, *The Unified Modeling Language User Guide*, Addison Wesley Longman, Inc., Reading, MA, 1999.

(Sunye et al, 2000) Sunye, Gerson, Le Guennec, Alain, and Jezequel, Jean-Marc, "Design Patterns Application in UML", Proceedings of the 14th European Conference on Object-Oriented Programming, *Lecture Notes in Computer Science #1850*, Springer-Verlag, Berlin, 2000.

(Poole et al, 2003) Poole, Chang, Tolbert, Mellor, *Common Warehouse Metamodel Developer's Guide*, John Wiley & Sons, New York, NY, 2003.