SAND2000-2720J

Leveraging the UML Metamodel: Expressing ORM Semantics Using a UML Profile

by David Cuyler, Sandia National Laboratories

Abstract

This paper is a proposal for a UML Profile to facilitate expression of Object Role Modeling semantics in terms of the UML 1.3 Metamodel. The profile uses the extension mechanisms inherent to UML to clarify usage and semantics where necessary, and it proposes the use of the XML Metadata Interchange (XMI) specification for model exchange. Once expressed in terms of the UML Metamodel, ORM models can then be shared among UML-based tools and can be stored, managed and controlled via UML-based repositories. The paper provides an example of an ORM model converted to the XMI format, in accordance with the profile.

UML

Since its inception in the mid-1990s the Unified Modeling Language (UML) has become the dominant Object Oriented software modeling language. The UML specification prescribes both a diagram notation and a metamodel. The notation is particularly complete and capable as the means to document the structural and behavioral characteristics of software. However, modeling persistent storage structures has not been one of UML's strong points. The UML notation especially neglects constructs useful for precise analysis, design, development and management of relational schemata. Data modeling-specific notations and techniques have generally been stronger at this task than those oriented around UML. The UML metamodel, however, provides a structure that accommodates semantic information beyond what is typically expressed in the UML notation. In particular, the UML extension mechanisms of stereotypes, tagged values and constraints provide a basis for significantly expanding the applicability of the UML. This paper proposes a means by which the semantics of a specific data modeling notation (Object Role Modeling - ORM) could be accurately expressed in terms of the UML metamodel and its native extensions with no loss of semantic content.

XMI

Analysis and design tools today generally lack sufficient mechanisms for sharing content with other tools. Recently the OMG has published the definition of XMI, the XML Metadata Interchange Format, for interchange of model information among tools. XMI is not the first attempt to address the issue of a common format for sharing models. CASE Data Interchange Format (CDIF), MetaData Interchange Specification (MDIS) and others represent attempts to define such a format. As an XML grammar, XMI has an advantage over its forerunners, in that XML is a freely published standard and is supported by a growing number of effective and inexpensive tools.

ORM

Object Role Modeling (ORM), as defined by the work of Dr. Terry Halpin, and with a heritage in Natural Language Information Analysis Method (NIAM), has one of the richest content models of any persistent modeling grammar. ORM is unique among the data modeling techniques mentioned above as it can be used to document a persistent data model for both relational and object schemata. Dr. Halpin has recently published several works comparing ORM with UML, and in them has implied that conversion of an ORM model to UML might be possible. This

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000.

paper provides a definition, in the form of a UML Profile, that provides the extensions necessary to perform this conversion and to accurately reflect the semantic content of an ORM model. It also demonstrates a significant conversion via software. The XMI 1.1 definition provides the UML output format. A sample model that comes with a popular ORM tool provides the input. ORM semantics and usage differ from those typically associated with UML primarily in the following areas:

- What would normally be considered an Attribute in UML is represented in ORM as an Association (FactType).
- A typical ORM Constraint restricts the allowed population of an AssociationEnd (Role) or a set of AssociationEnds. This contrasts with the UML, where constraints typically govern whole Associations, Classes, or Behavioral Features.
- The ORM analysis process relies heavily on sample populations of associations (Links) to assist in the determination of Constraints. This is not consistently used in UML techniques.
- ORM methods are typically used to model persistent data stores, helping to optimize the data structure and reduce the incidence of anomalies in the population of the data store. UML is typically used to model run-time characteristics of software.

None of these differences violates intrinsic capabilities of the UML metamodel. Rather they represent deviations from normal usage of the UML notation.

Author: David Cuyler, Sandia National Laboratories: mailto:dscuyle@sandia.gov
Sandia is a multi-program laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

References

- Model: A UML Profile for Object Role Modeling v.0.5 (this model)
- MetaModel: UML v.1.3
- MetaMetaModel: MOF v.1.3
- XML Metadata Interchange (XMI) v.1.1 (Proposed)
- Requirements for UML Profiles (Green Paper)
- White Paper on the Profile mechanism
- W3C XML 1.0 Definition
- W3C XSL Definition
- Object Role Modeling Web Site
- Dr. Terry Halpin on UML and ORM
- Journal of Conceptual Modeling
- Persistence Modeling in the UML
- An ORM Model Expressed in Accordance With This Profile
- XSL to Transform an XMI Model to HTML
- Dr. Terry Halpin, *Conceptual Schema and Relational Database Design, revised 2nd ed.*, (WytLytPub, 1999).

DISCLAIMER

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

UML Profile for Object Role Modeling

1. Introduction

Object Role Modeling techniques produce a detailed domain model from the perspective of the business owner/customer. The typical process begins with a set of simple sentences reflecting facts about the business. The output of the process is a single model representing primarily the persistent information needs of the business. This type of model contains little, if any reference to a targeted computerized implementation. It is a model of business entities not of software classes. Through well-defined procedures, an ORM model can be transformed into a high quality object or relational schema.

2. Summary of Profile

1. Identified Subset of UML 1.3

UML Package	MetaClass
Core	Association
Core	AssociationClass
Core	AssociationEnd
Core	Attribute
Core	Class
Core	Constraint
Core	DataType
Core	Dependency
Core	Generalization
Model Management	ElementImport
Model Management	Model
Model Management	Package
Model Management	SubSystem
Extension Mechanisms	Stereotype
Extension Mechanisms	TaggedValue
Common Behavior	DataValue
Common Behavior	Instance
Common Behavior	Link
Common Behavior	LinkEnd
Common Behavior	LinkObject

Common Behavior	Object

2. Stereotypes

MetaClass	Stereotype	Supertype(s)
Class	«entityType»	
Class	«valueType»	
Association	«factType»	
Association	«derivedFactType»	«factType»
Association	«unaryFactType»	«factType»
AssociationEnd	«ormRole»	
AssociationClass	«nestedObjectType»	
Attribute	«referenceMode»	
Attribute	«valueAttribute»	
Constraint	«ormConstraint»	
Constraint	«unique»	«ormConstraint»
Constraint	«mandatory»	«ormConstraint»
Constraint	«frequency»	«ormConstraint»
Constraint	«ring»	«ormConstraint»
Constraint	«set»	«ormConstraint»
Link	«factInstance»	
LinkEnd	«roleInstance»	

3. Tagged Values

Stereotype	Tag
«factType»	sentence
«derivedFactType»	isStored
«derivedFactType»	derivationRule
«ormRole»	textBefore
«ormRole»	textAfter
«ormRole»	reading
«nestedObjectType»	sentence

«unique»	isPrimary
«frequency»	multiplicity
«ring»	ringType
«set»	setType
«set»	numberOfSets
«set»	rolesPerSet

4. Constraints

Stereotype	Constraint
«ring»	binaryFactOnly
«ring»	roleHeritage
«set»	ordered
«set»	roleHeritage
«set»	min2Sets

3. Stereotypes and Notation

1. Static Structure Stereotypes

Stereotypes of constructs typically associated with a Static Structure (Class) Diagram.

1. **«entityType»** : Class An object type (Class) that represents a collection of real-world objects having similar characteristics, some of which may be useful for identification. The characteristics may be inherited from a supertype. On transformation, an entity type generally maps to a Class. In an ORM model, an entity type appears as a solid ellipse.

2. **«valueType»** : Class An object type (Class) that represents a domain or a set of allowed literal values. On transformation a value type generally maps to a Class Attribute. In an ORM model, a value type appears as a dotted ellipse.

3. **«factType»** : Association A fact type consists of one or more object types and predicates indicating the roles played by the object types. In an ORM model, a fact type appears as a contiguous series of a number of role rectangles equal to the number of object connections.

   Required Tags

   1. sentence : string [0..1] A skeletal verbalization of the primary fact sentence with a placeholder for each role. This sentence can be derived by concatenating the text values and placeholders

associated with each role.

4. **«derivedFactType»** «factType» : Association A fact type whose instances can be derived algorithmically from instances of other fact types.

   Required Tags

   1. isStored : boolean [0..1] An indicator of whether the derived instances of this fact type are stored. If not stored, instances are calculated at run-time.

   2. derivationRule : string [0..1] The algorithm by which instances of this fact type are derived.

5. **«unaryFactType»** «factType» : Association A fact type with a single role. Indicates truth of the predicate when populated. Since UML well-formedness rules disallow an Association with only one AssociationEnd, it is necessary to represent a unary fact type as a binary fact type with a Boolean value type playing the second role.

6. **«ormRole»** : AssociationEnd The predicate text assigned to a role is read in a specific order and includes a placeholder for each object type in a coherently readable sentence structure. In an ORM model, a role appears as a rectangle with a line connection to an object.

   Required Tags

   1. textBefore : string [0..1] Text associated with a role that appears before the role's placeholder in the primary reading of the fact sentence.

   2. textAfter : string [0..1] Text associated with a role that appears after the role's placeholder in the primary reading of the fact sentence.

   3. reading : string [0..1] The full sentence skeleton from this role's perspective (this role is the first role in the sentence). For fact types with more than two roles, includes '..' at the position of each role.

7. **«nestedObjectType»** : AssociationClass A nested-facttype is an entity type that is formed from a facttype to enable the facttype itself to play roles. Generally, every role in the facttype must be constrained by a single unique constraint. In an ORM model, a nested-facttype appears as a solid ellipse surrounding the facttype.

   Required Tags

   1. sentence : string [0..1] A skeletal verbalization of the primary fact sentence with a placeholder for each role. This sentence can be derived by concatenating the text values and placeholders associated with each role.

8. **«referenceMode»** : Attribute The attribute or set of attributes that is used to uniquely identify an instance of an entity type. May be derived from associations or the object type's heritage.

9. **«valueAttribute»** : Attribute The attribute that holds the value assigned to an instance of a value type.

2. Constraint Stereotypes

The ORM language is rich with constraint types. Constraints consistently inolve interactions among roles. This contrasts with standard UML, which focuses on constraints for classes, associations and attributes.

1. **«ormConstraint»** : Constraint A general constraint exhibiting properties common to all ORM constraint types. This sterotype may be used to document constraints that do not conform to the pre-defined constraint stereotypes. Other ORM constraint stereotypes inherit from ormConstraint.

2. **«unique»** «ormConstraint» : Constraint Constrains a set of roles in such a way that each instance of the set of constrained roles is distinguishable from every other instance. A unique constraint appears as a two-headed arrow spanning the constrained roles.

   Required Tags

   1. isPrimary : boolean [0..1] A value of "true" indicates that a unique constraint acts as the primary identifier for the relevant entity type.

3. **«mandatory»** «ormConstraint» : Constraint Specifies that every instance of an object type's population must play the connected role. A mandatory constraint may involve more than one role, in which case each instance of the object type must play at least one of the connected roles. A mandatory constraint appears as a small, filled circle at the point where a role connector(s) touches the object type.

4. **«frequency»** «ormConstraint» : Constraint Places a restriction on the number of times a particular value, or set of values, can appear in the population of one or more roles in a given facttype. Uses standard UML constraint notation with the frequency range indicated in the constraint body.

   Required Tags

   1. multiplicity : multiplicityRange [0..1] The minimum and maxinum number of instances permitted.

5. **«ring»** «ormConstraint» : Constraint Constrains a pair of roles to conform to certain predefined characteristics, indicated by the semantics of the ringType. This type of constraint only applies to the roles in a binary fact type that associates an entity type with itself.

Required Tags

1. ringType : enum [0..1] The type of ring constraint.

Stereotype Constraints

1. binaryFactOnly : {Must constrain both roles of a binary fact type}
2. roleHeritage : {Both constrained roles must be played by object types with common heritage}

2. **«set»** «ormConstraint» : Constraint A set constraint expresses a data dependency among two or more ordered sets of roles.

Required Tags

1. setType : enum [0..1] The type of set constraint where '**subset**' specifies that instances of one role set must be a subset of instances of a second role set; '**equality**' requires that the instances of each constrained role set must be the same as the instances of each of the other constrained role sets; '**exclusion**' prevents instances of any one constrained role set from appearing as instances of the other constrained role sets.
2. numberOfSets : int [0..1] The number of sets constrained.
3. rolesPerSet : int [0..1] The number of roles in each constrained set.

Stereotype Constraints

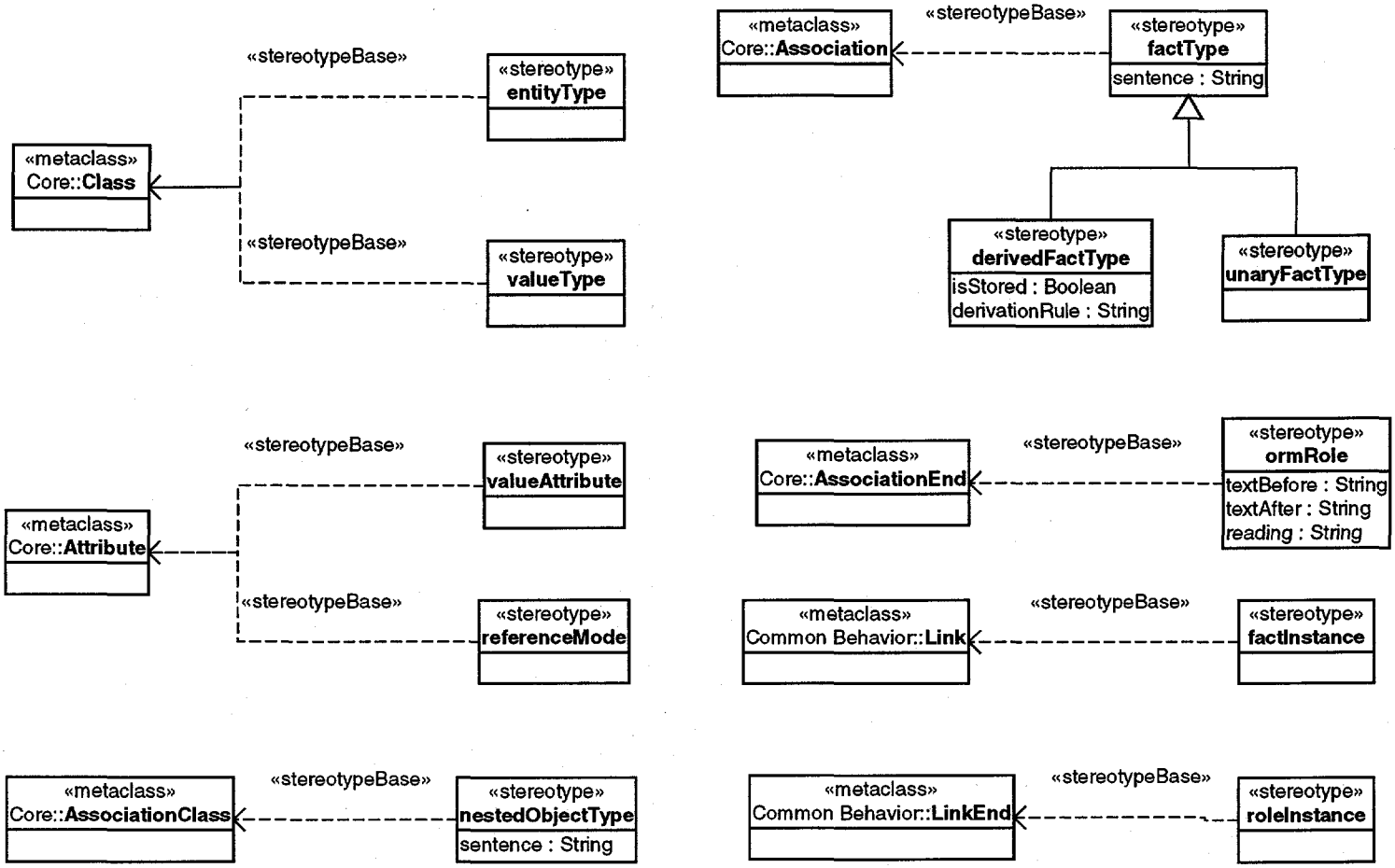
1. ordered : {Roles constrained are ordered by set sequence, then by role within set}
2. roleHeritage : {Roles occupying the same position in each role set must be played by object types with common heritage}
3. min2Sets : {Constrains at least two sets of one or more roles}

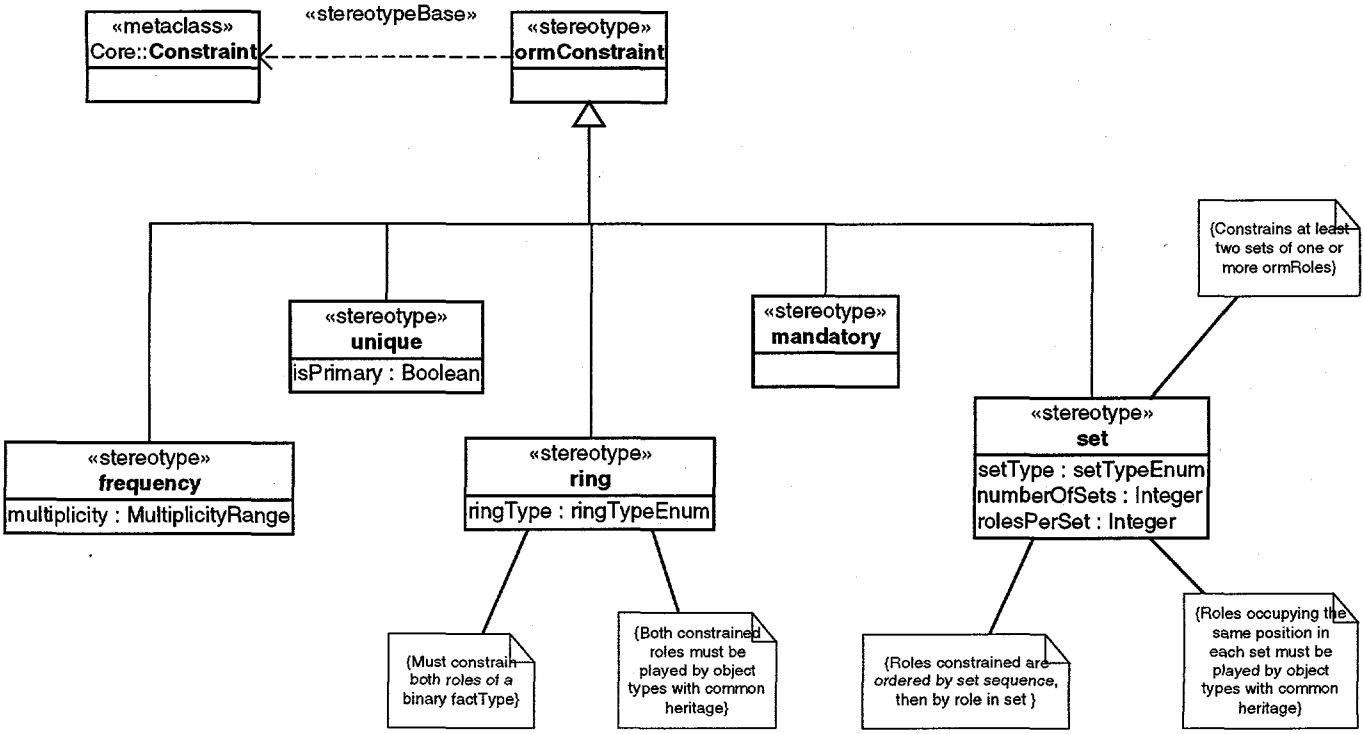
2. Instance Stereotypes

Object Role Modeling techniques rely on concrete examples as the basis for developing a model. Examples are always populations of fact types.

1. **«factInstance»** : Link An example fact instance useful for discussion with user-owners to validate facts and derive constraints.
2. **«roleInstance»** : LinkEnd The value associated with a role in a fact instance.

ORM Profile Extensions





ORM Profile UML Subset

«metaclass»
Extension Mechanisms::**Stereotype**

«metaclass»
Extension Mechanisms::**TaggedValue**

«metaclass»
Model Management::**Package**

«metaclass»
Model Management::**Model**

«metaclass»
Model Management::**SubSystem**

«metaclass»
Model Management::**ElementImport**

«metaclass»
Core::**Association**

«metaclass»
Core::**AssociationClass**

«metaclass»
Core::**AssociationEnd**

«metaclass»
Core::**Attribute**

«metaclass»
Core::**Class**

«metaclass»
Core::**Constraint**

«metaclass»
Core::**DataType**

«metaclass»
Core::**Dependency**

«metaclass»
Core::**Generalization**

«metaclass»
Common Behavior::**Link**

«metaclass»
Common Behavior::**LinkEnd**

«metaclass»
Common Behavior::**Instance**

«metaclass»
Common Behavior::**DataValue**

«metaclass»
Common Behavior::**Object**

«metaclass»
Common Behavior::**LinkObject**

Tag Data Types

«datatype»
Data Types::**Boolean**

«datatype»
Data Types::**String**

«enumeration»
Data Types::**ringTypeEnum**

- acyclic
- antisymmetric
- asymmetric
- intransitive
- irreflexive
- symmetric
- acyclic-intransitive
- asymmetric-intransitive
- intransitive-symmetric
- irreflexive-symmetric

«enumeration»
Data Types::**setTypeEnum**

- equality
- exclusion
- subset

«datatype»
Data Types::**MultiplicityRange**