Accredited Standards Committee
X3 - INFORMATION TECHNOLOGY

Doc. No: X3H7-96-01
Date: January 5, 1996

Reply to:
Haim Kilov
IBM T J Watson Research Center
30 Saw Mill River Road
Hawthorne NY 10532 USA
kilov@watson.ibm.com

Subject: Some materials on relationships for the enterprise viewpoint of RM-ODP

This document shows how the ISO General Relationship Model (GRM) describes relationships, and presents other related considerations about relationship specifications. GRM [GRM 95] is an IS accepted in 1995 which provides very good definitions and guidelines for describing relationships independently of their representation mechanisms; the definitions are reasonably rigorous; and the generic relationship examples show how these definitions can be used in enterprise business specifications.

In what follows, substantial portions of the GRM have not been referenced. In particular, the syntax of templates, the details of mapping from the relationship world onto the systems management world, as well as issues related to cardinalities have been omitted. The latter have not been included because all cardinality constraints are just conjuncts of appropriate invariants and pre- and postconditions described below.

Declarative specifications; invariant

A relationship is defined by GRM as a collection of objects together with an invariant referring to the properties of the objects; and a relationship class is defined as a named set of relationships sharing the same definition. Thus, the relationship invariant defines the "collective state" of the relationship participants. A participant fulfils a role in a relationship, and a role describes the properties common to a particular kind of participant in a relationship.

GRM further states that relationship behavior is representation-independent and is described in terms of invariants over participant roles and invariants, pre-conditions, and post-conditions over relationship management operations and notifications. Thus, relationship operations (and notifications) describe, in a declarative manner, the "collective behavior" [OODBTG 91] of the relationship participants; in other words, these operations are not owned by any of the participants. An invariant is defined by GRM in a somewhat non-traditional manner: as a logical predicate that must remain true during some scope; a scope might be the lifetime of a managed relationship or the execution of a relationship management operation.

An invariant the scope of which is the execution of an operation may be represented eg in the Z operation schema (by convention [Spivey 92]) using Ξ. In other words, a name beginning with a Ξ denotes a schema declaring "what remains unchanged" during the operation, i.e. declaring those variables that are referred to but remain unchanged in the operation. All other variablesnot explicitly changed in the operation (i.e. not represented in the Z operation schema (by convention) using Δ) remain totally unused in the operation.

Operations

The GRM further states that relationship management operations and notifications are expressed in terms of "prototypical" ones, i.e. establish, terminate, bind, unbind, query, and notify. The semantics of these prototypical operations and notification are given by their pre- and postconditions (and - for some operations - the operation invariant). For example, **establish** is defined using pre- and postconditions: the precondition states that the managed relationship does not exist; managed objects that have mandatory participation in the managed relationship do not exist; other managed objects specified in the establish operation to be bound are of a class permitted to take on the role and exist but are not bound into the managed relationship; and the postcondition states that the managed relationship exists; objects that have mandatory participation in the managed relationship exist and are bound into the relationship. The definition of **unbind**, in addition to pre- and postconditions, does include an operation invariant: the relationship exists and the objects specified in the unbind operation exist; the role and relationship cardinality constraints are not violated. The same approach is used in information modeling [Kilov, Ross 94]: relationships are defined by their invariants, and elementary relationship operations are defined by their pre- and postconditions. Obviously, a relationship does not have to support all of these basic operations: asymmetric relationship, for example, does not support the **bind** or **unbind** operations.

Representation

A relationship, its invariant and operations may be represented in terms of (ie refined using) "ordinary" objects, their attributes, and operations (GRM uses the term "systems management operations" for the latter). These representations are usually convenient for implementation, but difficult to use for understanding the semantics of the original relationship. The semantics of a representation is precise, but not abstract, and therefore the (abstract, representation-independent) semantics of the original relationship is lost in representation details irrelevant for its understanding. Moreover, the existence of several possible representations of the same relationship implies that the requirement to use a particular representation in the specification of a relationship imposes an unnecessary - and often a non-optimal - choice on the specifier, further distracting him from understanding the subject matter and from conveying this understanding to the users. Message-oriented - rather than generalized - object models [OODBTG 91]are a well-known example of this distraction: their users have to invent an object-owner for each operation (and invariant!) described in the specification, whereas in most cases inventing such an owner at the (business) specification level is quite unnatural.

The GRM states that the mapping of relationship operations and notifications is such that the pre- and post-conditions and invariant for the relationship operations or notifications and the invariant for the relationship are respected by the systems management operations and notifications.

Reuse; genericity

The GRM describes specialization as the derivation of classes from existing relationship classes by means of inheritance and incremental specification: a relationship class may be specialized by combining characteristics inherited from one or several relationship classes with characteristics specified in the relationship class template. The specialized class is referred to as the subclass of the original class(es); the original class(es) are referred to as the superclass(es) of the specialized class. (Note that the existing definitions in Part 2 of RM-ODP clearly differentiate "type" from "class" while this is not done in GRM).

The GRM provides several examples of generic relationships, such as dependency, symmetric relationship, and composition. These generic relationships (and a very limited number of other ones) can be used to provide a richer built-in infrastructure for a specification. Specificationsof these generic relationships are abstract and use formal parameters. They are instantiated with different actual parameters and thus applied in avariety of contexts which share some common characteristics. An example of a generic relationship includes, in particular, the relationship invariant and the signature, pre- and postconditions for appropriate basic relationshipoperations.

For example, the invariant for the **dependency** relationship is defined in GRM as "There exist two roles in this relationship class: parent role and dependent role. The existence of a participant in the dependent role implies the existence of at least one corresponding participant in the parent role. The participant fulfilling the parent role should belong to a managed object class different from the participant(s) fulfilling the dependent role in the same instance of this relationship class."; and the operation **bind dependent** is defined as "*Signature*: The class and identity of the participant in the parent role; the class and identity of the appropriate participant in the dependent role to be associated with this participant in the parent role. *Precondition*: The corresponding instance of this Dependency relationship class exists. The participant in the parent role to be associated with this participant in the dependent role exists and is bound into this instance of this Dependency relationship class. There exists at least one other participant in the dependent role bound into this instance of the Dependency relationship class. *Postcondition*: Both the participant in the parent role and the appropriate participant in the dependent role to be associated with this participant in the parent role exist and are bound in the corresponding instance of this Dependency relationship class."

Asymmetricity

We can consider a relationship using a somewhat different approach which is in very good agreement with the concepts described in GRM, but provides additional information and helps to represent relationships in a natural anduniform manner eg in Z. A relationship may be considered as a binary asymmetric relation. A relation type relates a source type to a target type[Potter 91]. Any of these types may be nonelementary (e.g. an instance of such a type may be a set, a set of sets, etc.). If a nonelementary type is a set, then its corresponding elementary type will be an element of this set. Let usconsider some (generic) examples.

- A Dependency (see also the GRM example above) is a relation between the source type (parent) and the target type (dependent). Both the source and the target types are elementary. The existence of an instance of the Dependency relationship is equivalent to the existence of the corresponding instances of the source and target types. For example, the existence of an instance of an *bank customer - account* dependency is equivalent to the existence of an instance of a *bank customer* and the existence of corresponding instance(s) of account (for this *bank customer*).
- A Composition is a relation between the source type (composite) and the target type (set of component types). Here the source type is elementary, and the target type is not. An instance of the composite type corresponds to a set of sets of instances for each of its component types. In a *document* composed of *texts, pictures,* and *tables*, an instance of the *document* corresponds to a set consisting of three elements. Each element, in turn, is a set of instances of pieces of *text, pictures*, and *tables*, correspondingly. Each of these sets may be empty.
- A Subtyping is a relation between the source type (supertype) and the target type (set of subtypes). Again, the source type is elementary, and the target type is not. The existence of an instance of the Subtyping relationship is equivalent to the existence of the corresponding instances of the source and target types. For any given Subtyping relationship instance, the set of instances of the target type will be a subset of the set of instances of the source type (this does not usually hold for other relationships). For a supertype *employee* and subtypes *technical employee* and *managerial employee*, the set of all instances of *technical employees* and *managerial employees* will be a subset of the set of instances of *employee* (if the Subtyping is exhaustive, then the union of these sets of subtype instances will be equal to the set of supertype instances).
- A Symmetric Relationship is a relation between the source type (symmetricrelationship object) and the target type (set of participating - regular -entity types). The existence of an instance of the Symmetric Relationship is equivalent to the existence of the corresponding instances of the source and target types. The number of elements in the latter set must always be more than one; if it is equal to two, the Symmetric Relationship is traditionally called "binary," whereas if it is greater than two, the Symmetric Relationship is traditionally called "n-ary." The number of elements in each set of typed participating entity instances is equal to one.

Consider the sets of types of an instance of a source type and an instance of an elementary target type (an object instance in RM-ODP usually satisfiesseveral predicates - types). These sets should be different, although they may have a nonempty intersection, so that the following invariant holds: for any relationship instance, the set of types for an instance of its source type is not equal to the set of types for a corresponding instance of its elementary target type. For example, in a Dependency, a parent instance and a dependent instance may satisfy a common type, *person*, but they will satisfy other, different, types as well: the parent instance will satisfy the *employee* type, and the dependent instance will satisfy the *dependent* type. For another example, in a Composition between *things*, both a composite instance and a component instance will satisfy the thing type, but the composite instance will also satisfy the *assembly* type.

In most, but not all, cases, additional conjuncts may be discovered for the relationship invariant. For example, in most cases the existence of a relationship instance is equivalent to the existence of an instance of the source type and the existence of an instance of its target type. For some kinds of composition, however, this need not be the case. Therefore, inconsidering relationships it makes sense to investigate whether this predicate holds.

For another example, in most cases (with the notable exception of Subtyping!), the relation between a source type and its elementary target type is irreflexive. The same is true for the transitive closure of this relation. In other words, in most cases for any relationship instance (and its transitive closure), the sets of instances of its source and elementary target types have an empty intersection: an instance that belongs to a source type cannot belong to an elementary target type in the same relationship instance. Indeed, in an instance of Dependency, a parent instance cannot have itself as a dependent; in an instance of Composition, a composite instance cannot have itself as one of the components; and so on. Moreover, a parent instance cannot have itself as an indirect dependent; and a composite instance cannot be, directly or indirectly, its own component. Again, inconsidering relationships it makes sense to investigate whether this predicate holds.

References

[GRM 95] ISO/IEC JTC1/SC21, Information Technology - Open Systems Interconnection - Management Information Services - Structure of Management Information - Part 7: General Relationship Model. ISO/IEC 10165-7, 1995.

[Kilov, Ross 94] H.Kilov, J.Ross. Information modeling: an object-oriented approach. Prentice-Hall, 1994.

[OODBTG 91] Object Data Management Reference Model. (ANSI Accredited Standards Committee. X3, Information Processing Systems.) Document Number OODB 89-01R8. 17 September 1991. (Also in: Computer Standards & Interfaces, Vol. 15 (1993), pp. 124-142.)

[Potter 91] B.Potter, J.Sinclair & D.Till. An introduction to formal specification and Z. Prentice-Hall International Series in Computer Science, 1991.

[Spivey 92] J.M.Spivey. The Z notation: a reference manual. Second Edition. Prentice-Hall International Series in Computer Science, 1992.