miUML Metamodel Descriptions
Population Subsystem

Leon Starr

Wednesday, October 28, 2009

mint.miUMLmeta.td.7

Version 2.2.1

Table of Contents

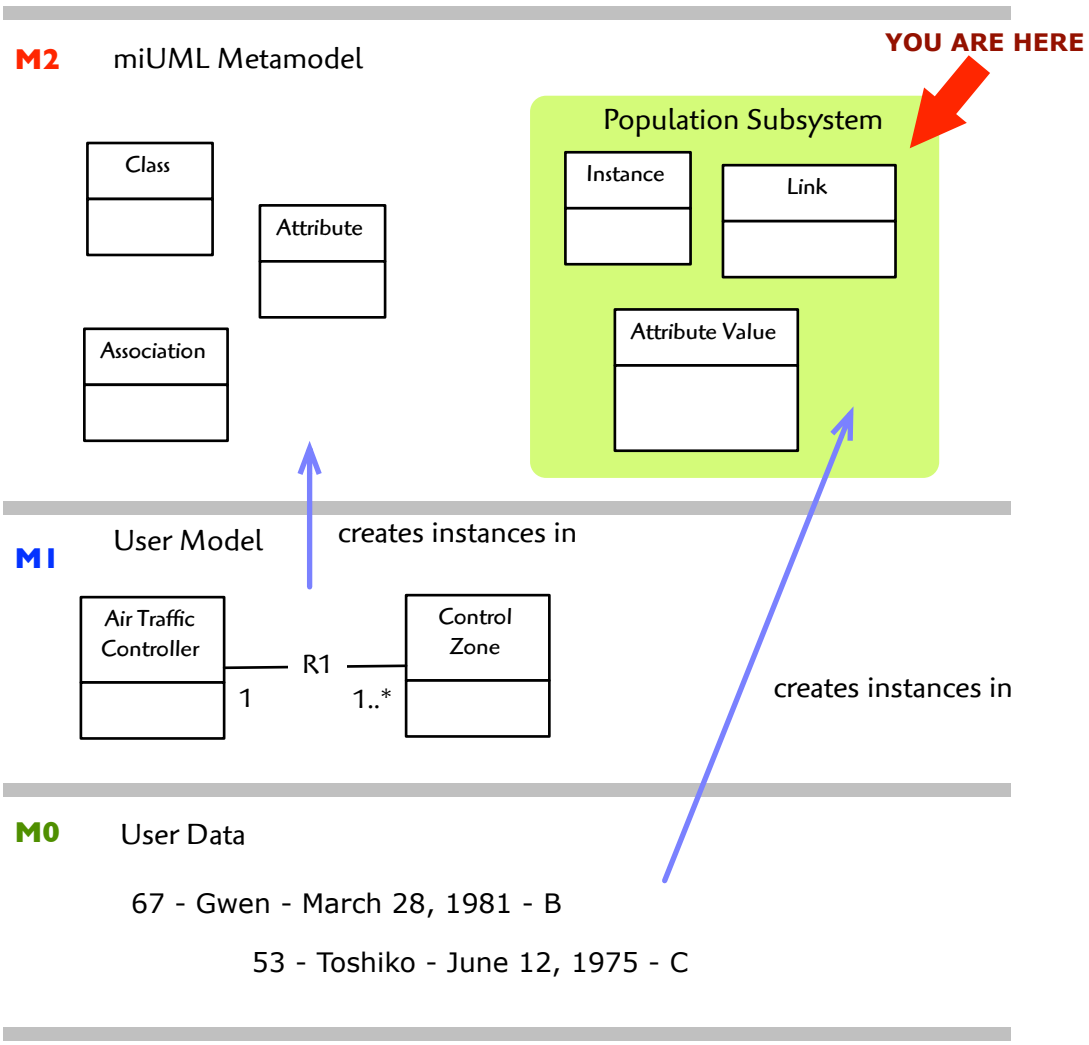
Introduction

In Model Driven Architecture (MDA) terminology, the other (non-population) miUML metamodel subsystems constitute the M2 (meta) layer which will be populated with M1 (model) layer data.

Let's say you have an executable UML model for the Air Traffic Control application with classes like "Air Traffic Controller", "Runway" and so forth. This would be an M1 layer model with respect to our M2 miUML metamodel. The M1 classes become objects loaded into the M2 classes. So class "Air Traffic Controller" becomes an object populated into the Class class. The association "R3 - is logged into" populates the Association and Perspective metamodel classes appropriately.

Now, where do we put the M0 (value) layer data such as "Gwen", "Owen", "27R", "210", and so forth? That's were the Population Subsystem comes in. It accommodates M0 layer values associated with an M1 layer user model BOTH populated into the miUML M2 layer model.

MDA Abstraction Layers



Thus, the Population Subsystem of the miUML Metamodel makes it possible to instantiate Classes and Associations with Objects, Links and Attribute values. Current state values and generalization relationship links must also be accommodated, though they will be added in a future version.

The generalization population subsystem subsystem/cluster is in process and should be released in a week or three.

Key to Styles and Conventions

I often use COLOR to convey information. If you have the ability to print or display this document in color, it is highly recommended. Here are a few notes on the font, color and other styles used in class model descriptions.

Naming classes, types and subsystems

Modeled elements such as classes, types and subsystems are written with initial caps. The text Air Traffic Controller would probably be a class. If you see the text Application Domain, you know it is a modeled element, most likely a domain.

Naming attributes

Attributes are named with an initial cap followed by lower case, Time_logged_in, for example.

Instance values

Instance data, such as attribute values, are either in quote marks or using the instance data style. Some Air Traffic Controller names might be "Gwen" and "Owen". But usually I use the following color/style: Gwen and Owen to denote instance data. Sorry about the lack of consistency here, but I'm still experimenting with the best presentation.

Whitespace

When programming, I use the squashedNamingStyle. That's fine for programming, but analysis is all about easy readability so, whenever possible whitespace is employed, for example: Air Traffic Controller, Time logged in. When whitespace leads to ambiguity in some context, underscores are used instead. On_Duty_Controller.Time_logged_in. Type-ability is sacrificed a bit for readability, but analysis is more read than write intensive if it's done well.

Attribute heading color

Attribute headings are colored according to the role the attribute plays in its class. Referential attributes are brown, **Logged in controller**, for example. Naming attributes are blue, **Number**, for example. Finally, descriptive attributes are red, **Time logged in**, for example. (One of the nice things about color is that it helps eliminate the need for underscores).

Referential renaming

A referential attribute will often have a name that reflects its role rather than the name of the base reference. Station.Logged_in_controller may refer to On_Duty_Controller.ID, for example. Such renaming will be defined in the referential attribute's description.

Local Data Types

The following data types are used by attributes in this subsystem.

Long Name

This is a string of text that can be long enough to be readable and descriptive. A handful of words is okay, a full tweet is probably too much. Since the exact value may change to suit a particular platform, it is represented symbolically as "LONG_TEXT".

Domain: String [LONG_TEXT]

Nominal

This is a whole number used purely for naming with no ordinal or computational properties.

Identifiers used only for referential constraints may be stripped out during implementation (assuming the constraints are still enforced). During simulation and debugging, however, these can be nice to have around for easy interpretation of runtime data sets without the need for elaborate debug environments. "File-6:Drawer-2:Cabinet-"Accounting" is easier for a human to read than a bunch of pointer addresses.

Domain: The set of positive integers {1, 2, 3, …}.

Association Population

Just as a Class is populated with Instances, an Association is populated with Links. An Association Population will exist only if an Association is Populated with at least one Link. This class was to makes it possible to number Links uniquely within each Association.

This similarity of Class Population and Association Population and instantiation relationships brings up the possibility of defining a "Populatable Element" as a superclass. In fact, Subsystem Element (Domain Subsystem) appears to fit the bill, but only by coincidence. If the need arises for yet another type of Population, the generalization option can be revisited. At present, there would be no net reduction in the model, so let's keep it flexible for now and keep an eye on it.

Attributes

Population

Population.Name

Rnum

Association.Rnum

Domain

Population.Domain and Association.Domain (constrained to be identical)

Identifiers

1> Population + Rnum + Domain

Relationships

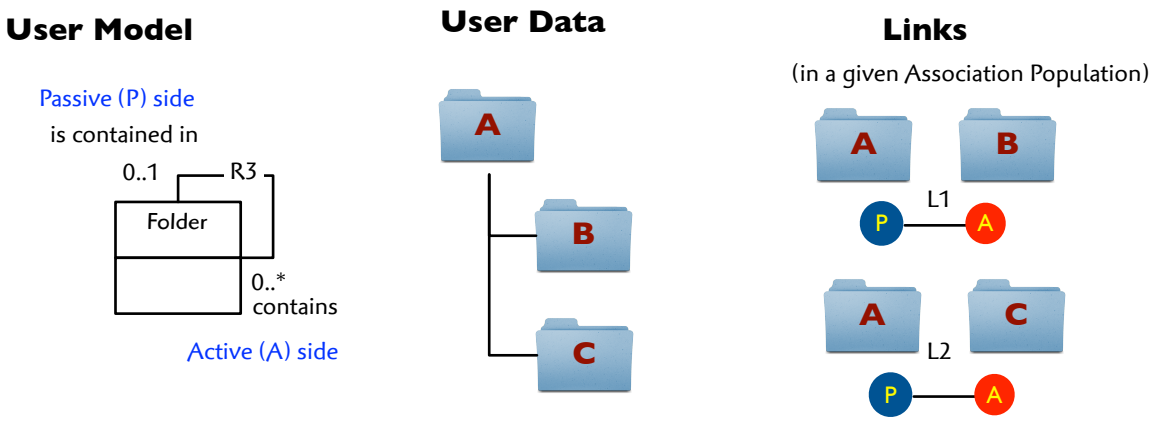
R302
Population INCLUDES zero, one or many Association
Association IS INCLUDED IN zero, one or many Population

An Association Population is defined when at least one Link is created for an Association in a given Population.

An Association may be populated with a different set of M0 Links in each Population. A null Population could be defined with no instance data and consequently no Association Populations.
.

Asymmetric Link

This type of Link is asymmetric because each side of the Link corresponds to a distinct Active or Passive Perspective. A common example might be a hierarchy of "Folders" with a Binary Association "contains (active) / is contained within (passive)". The two Perspectives are not identical and thus, the Links are asymmetric. The Association could have been named differently "is parent of / is child of" with the same intended meaning. In this case, the assignment of active and passive sides would be arbitrary, but the Perspectives differ, nonetheless.



Attributes

A_side_instance, P_side_instance, Class

Partial reference to the Reflexive Link superclass.

Number, Rnum, Domain, Population

Partial reference to both the Reflexive Link and Binary Link superclasses.

Identifiers

1> Number + Rnum + Domain + Population

Links are numbered uniquely within an Association Population.

2> A_side_instance + P_side_instance + Class + Rnum + Domain + Population

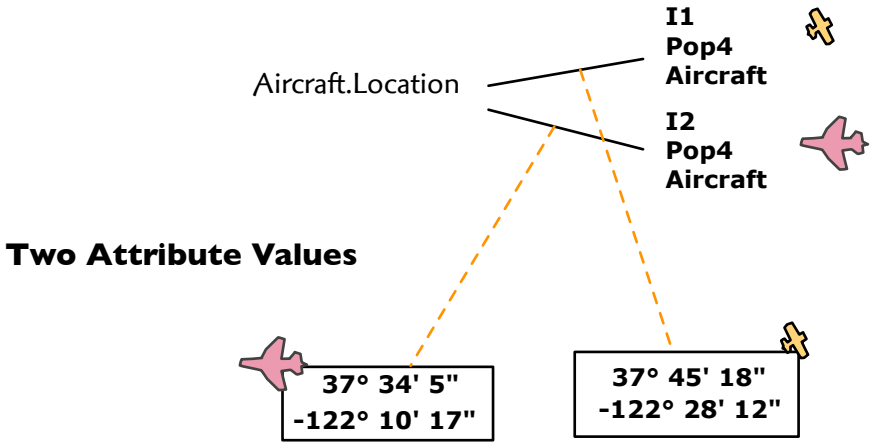
There may only be only one Link between the same two Instances participating in the same Association.

Relationships

None.

Attribute Value

An Instance of a Class must carry one data value for each Attribute defined on that Class. A value supplied for an Instance's Attribute is an Attribute Value.



Departing slightly from Shlaer-Mellor, but closely following C.J. Date[1], a value can have arbitrarily complex structure as defined by a Type. So an Attribute Value will be broken down into one or more component values.

In keeping with Shlaer-Mellor, however, the action language should constrain the ability to parse or otherwise manipulate the components of a complex value. Date prescribes that specific operations be defined for each Type (no arbitrary parsing or poking around inside of attribute values). Matrix operations will probably be supplied in a plug-in service domain. This topic is outside the scope of this description, but my hope is to accommodate Date's formula for getting types and tables to play together nicely in the action language metamodel.

Attributes

Instance

Instance.Number

Attribute

Attribute.Name

[1] Databases, Types and the Relational Model - The Third Manifesto, C.J.Date, Hugh Darwen, Addison-Wesley, 2006, ISBN 978-0321399427

Class, Domain

Partial reference to both the Instance and Attribute

Population

Partial reference to the Instance

Identifiers

1> Instance + Population + Attribute + Class + Domain

The many to many rule for referential attributes yields this identifier.

Relationships

R304
Instance SUPPLIES VALUE FOR zero, one or many Attribute
Attribute REQUIRES VALUE FOR zero, one or many Instance

In miUML there is no such thing as an "optional" Attribute. Each Attribute of an Instance's defining Class MUST be assigned a value. Thus this relationship represents a cross product. Each instance of Attribute is related to each instance of Instance to yield an Attribute Value.

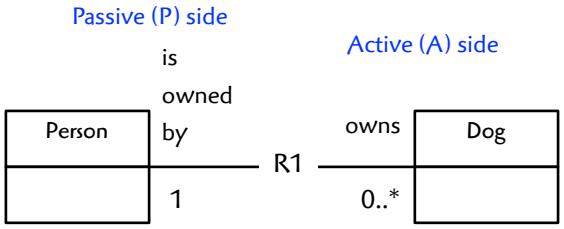
If there are no Instances defined for a particular Class, then that Class's Attributes will not require any values. If the Class has no Attributes defined, then no Instance of that Class will have any Attribute Values.

The cross product relationship is taken from the Shlaer-Mellor method.

Binary Link

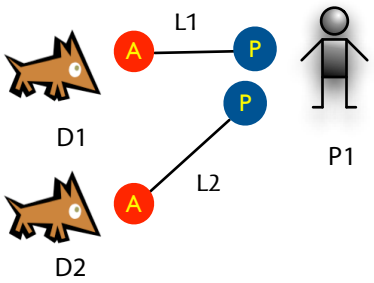
A Binary Link is an instance of a Binary Association. This kind of Link is created to represent an Instance related to another Instance in the same or different Class, or even to itself in the same Class.

User Model



Links

(in a given Association Population)



Attributes

Number

Link.Number

Rnum + Domain + Population

Partial reference to the Link superclass. The Rnum + Domain must also match the same Binary Association as the one referenced by the superclass.

Identifiers

1> Number + Rnum + Domain + Population

This identifier is established by the superclass.

Relationships

R315

Binary Link IS A Non-Reflexive or an Asymmetric Link

The "binary" in Binary Link refers to the dual counterpart Perspectives defined by verb phrases on each side of a Binary Association. Since, by definition, a Symmetric Association has a single Unary Perspective, it cannot be instantiated as a Binary Link. Only Non-Reflexive or Reflexive, Asymmetric Links have dual active/passive Perspectives.

R316

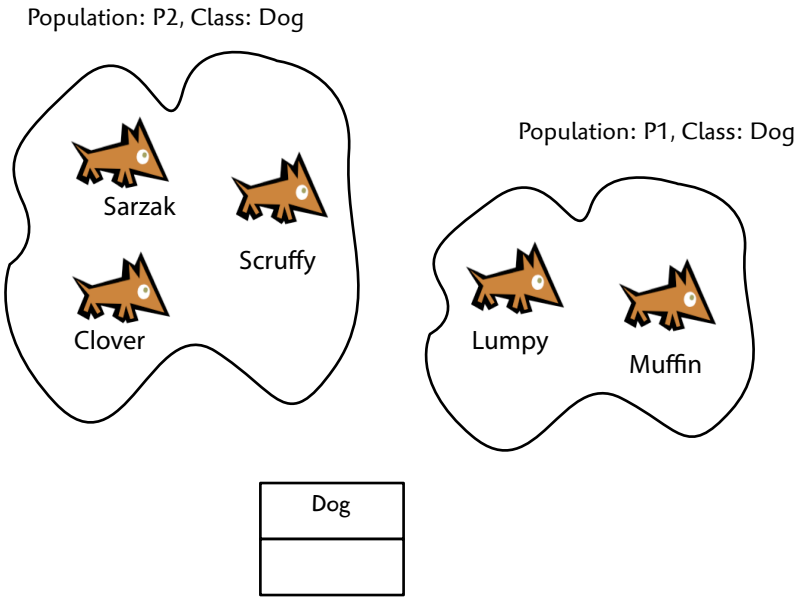
Binary Link INSTANTIATES one Binary Association
Binary Association IS INSTANTIATED BY zero, one or many Binary Link

Instantiation refers to the relationship of M1 (user model) entities to M0 (user data) entities in the miUML metamodel. The Binary Association instance is the M1 element with respect to the Binary Link instance M0 element. The Binary Association can exist without being instantiated, but a Binary Link cannot exist independently of its specification, a Binary Association.

Class Population

Since multiple scenarios might be defined for the same Domain, it makes sense that there may be many separate instance populations defined for a given Class. The population of a Class in a specific Population is represented as a Class Population.

Two Class Populations



In the original Shlaer-Mellor notation (where miUML has its roots planted) the Class Population and Instance classes could have been merged into a single many-association class. This is explicitly forbidden in the OMG UML standard. That fact alone wouldn't stop me from making many-association classes available in miUML, of course. (A mapping function can always be built to transform miUML to generic UML as needed).

But even in Shlaer-Mellor models , I had a tendency to break many-associations down a) to reduce reader confusion and b) to provide more association extension points for future analysis (as described in my class modeling book). Furthermore, novice modelers tended to use the many form when singular was appropriate due to multiplicity concept confusion.

That said, b) and the OMG arguments are more compelling than a). There are better ways to train novice users than by handicapping the modelint language for their benefit.

So I'm on the fence here - any opinions or other arguments to consider? Now would be a good time to express them!

Attributes

Population

Population.Name

Class

Class.Name

Domain

Population.Domain and Class.Domain (constrained to be identical)

Identifiers

1> Population + Class + Domain

Relationships

R302
Population INCLUDES zero, one or many Class
Class IS INCLUDED IN zero, one or many Population

A Class Population is defined if at least one Instance will be created for a Class in a given Population.

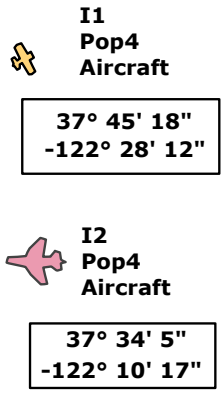
A Class may be populated with a different set of M0 instance data in each Population. A null Population could be defined with no instance data and consequently no Class Populations. It might be helpful to name such a scenario so that you don't need special initialization rules for the empty case. More to the point, there are no reasons to disallow an empty Population.

Field Value

The internal structure of an Attribute Value is broken down into one or more Field Values as specified in the Type Subsystem. In the simplest, and most common case, an Attribute Value will have only one Field.

Two Attribute Values broken down by Field Value

Value	Field	Type	Instance	Attribute	Class	Domain	Pop
37	Degrees	Latitude	2	Location	Aircraft	ATC	4
34	Minutes	Latitude	2	Location	Aircraft	ATC	4
5	Seconds	Latitude	2	Location	Aircraft	ATC	4
37	Degrees	Latitude	1	Location	Aircraft	ATC	4
45	Minutes	Latitude	1	Location	Aircraft	ATC	4
18	Seconds	Latitude	1	Location	Aircraft	ATC	4
-122	Degrees	Longitude	2	Location	Aircraft	ATC	4
10	Minutes	Longitude	2	Location	Aircraft	ATC	4
17	Seconds	Longitude	2	Location	Aircraft	ATC	4
-122	Degrees	Longitude	1	Location	Aircraft	ATC	4
28	Minutes	Longitude	1	Location	Aircraft	ATC	4
12	Seconds	Longitude	1	Location	Aircraft	ATC	4



Attributes

Field, Type

Partial reference to the Field. Constrained as described in the relationship description below.

Instance, Attribute, Class, Population

Partial reference to the Attribute Value

Domain

Partial reference to both the Field and the Attribute Value.

Identifiers

1> Field + Type + Instance + Attribute + Class + Domain + Population

The many to many rule for referential attributes yields this identifier.

Relationships

R308
Attribute Value BREAKS DOWN INTO TYPED one or many Field
Field IS A TYPED COMPONENT OF zero, one or many Attribute Value

An Attribute Value cannot carry any data unless it has at least one Field. More importantly, this is dictated by the Type Subsystem which requires at least one Field in every Attribute's Type. If an Attribute's Type is structured with multiple Fields, then the associated Attribute Value will require multiple data items, one per Field.

A Type is defined independent of whether or not it is used in a Domain. Consequently, a Field may be defined, but never used by any typed model elements. Even if a Field is declared as part of an Attribute's Type, that Attribute might not be populated to yield a corresponding Attribute Value.

If there are multiple Attribute Values corresponding to an Attribute of the same Type, a Field will be associated with multiple Attribute Values.

Constraint: The Fields associated with an Attribute Value are determined by the corresponding Attribute's Attribute Type.

R310
Field Value IS A Real Value, Integer Value or String Value

All Types ultimately decompose into commonly implementable single values. The miUML metamodel strives to be platform independent, so there is no mention of language level implementation types such as "double", "unsigned", "float" and so forth. Real and Integer are mathematically defined and can be implemented various ways. String is admittedly implementation inspired, but is commonly implemented in one way or another in most programming languages.

Enumeration Type values will be populated as String Values. Boolean Type values will be populated as Integer Values. Platform specific marking can be used to convert these types into whatever is appropriate for a given implementation.

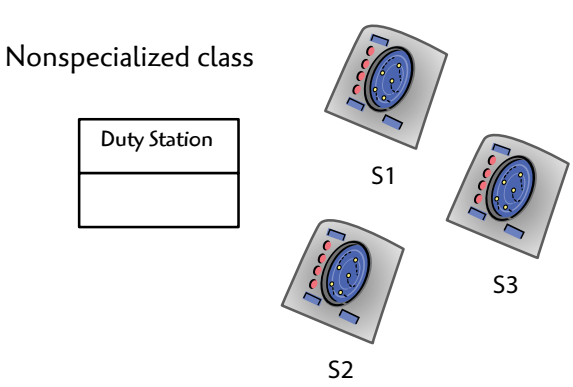
Instance

WIth explicit context, the term "instance" often refers to any M0 data value corresponding to an M1 model element counterpart. You could refer to an "instance of an association" to describe a Link, for example. Absent context, "instance" implicitly refers to Class. The term "object" is informally, and sometimes incorrectly, used synonymously with "instance".

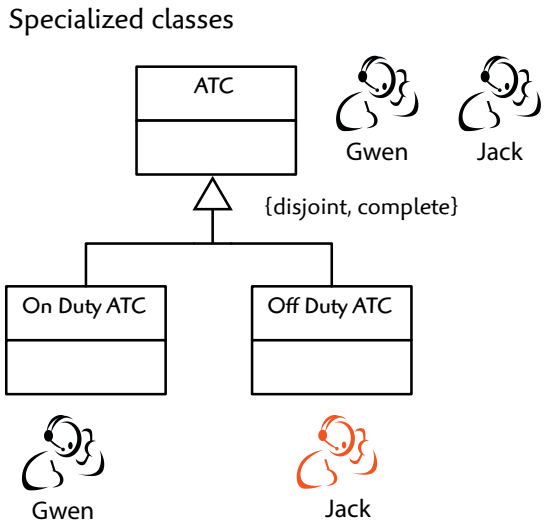
All this is fine for casual conversation, but the metamodel demands a bit more precision. Here, the term Instance refers exclusively to the instantiation of a single Specialized, or Non-specialized Class. Instance is NOT synonymous with Object for Specialized Classes.

Let's say that we have a Non-Specialized Class (a class not involved in any Generalization relationship) named "Duty Station". The three Instances of "Duty Station" we might create, "S1", "S2" and "S3" are three complete Objects.

Instances and Objects



Three instances, three objects



Four instances, two objects

Now let's introduce three Specialized Classes "Controller" subclassed as "On Duty Controller" and "Off Duty Controller" (considered Specialized Classes even though they, themselves are not superclasses). Controller objects "Gwen", "Owen" and "Jack" are each represented by two distinct Instances, one in the superclass and the other in one of the subclasses. So a Specialized Class will instantiate Objects each having multiple Instances while a Non Specialized Class will have one Instance per Object.

Objects and generalization relationships are not modeled in this version of the Population Subsystem. They will appear in the next release.

Attributes

Number

Assigned purely to distinguish one Instance from another.

Type: Nominal

Population, Domain, Class

The Instance is a member of this Class Population.

Identifiers

1> Number + Population + Domain + Class

Instances are numbered uniquely within each Class Population.

Relationships

R303
Instance IS A MEMBER OF exactly one Class Population
Class Population HAS MEMBER one or many Instance

A Class may be instantiated many times in the same Population, so there can be many Instance members of a Class Population.

Since a Class Population is defined as the inclusion of a Class in a given Population, it follows that a Class Population must have at least one Instance.

Shlaer-Mellor modelers will recognize this as an M-Mc:Mc relationship with Class Population and Instance compressed into a single class and this relationship omitted.

Integer Value

The integer data value stored as part or all of an Attribute Value.

Attributes

Field, Type, Instance, Attribute, Class, Domain, Population

Full reference to the Field Value superclass.

Value

The data stored in the Field.

Type: Integer

Identifiers

1> Field + Type + Instance + Attribute + Class + Domain + Population

Same as the superclass identifier.

Relationships

None.

Link

A Link is an instance of an Association. Each Link has two points of connection where each refers to an Instance of a Class. Depending on the type of Association, an Instance may be linked to itself.

Please note that a Link is applicable only to Association relationships. Generalization relationships will be addressed in the next release of the Population Subsystem.

Attributes

Number

Assigned purely to distinguish one Link from another.

Type: Nominal

Population, Domain, Rnum

The Link is a member of this Association Population.

Identifiers

1> Number + Population + Domain + Rnum

Links are numbered uniquely within each Association Population.

Relationships

R307
Link IS A MEMBER OF exactly one Association Population
Association Population HAS MEMBER one or many Link

A Link is created as an instantiation of an Association in a given Population.

Since an Association may be instantiated many times in the same Population, there can be many Links in an Association Population.

Since an Association Population is defined as the inclusion of an Association in a given Population, it must have at least one Link.

Shlaer-Mellor modelers will recognize this as an M-Mc:Mc relationship with Association Population and Link compressed into a single class and this relationship omitted.

R309

Link IS A Reflexive or Binary Link

A Link either connects Instances in the same Class or across different Classes. This relationship serves primarily to establish a numbering system for naming all types of Links uniquely within each Association.

Non Reflexive Link

If a Binary Association connects two distinct Classes, it may be populated with Non Reflexive Links. One Class will be referenced on the active side of the Association with the other Class on the passive side. An example is illustrated in the Binary Link class description.

Attributes

Number

Binary_Link.Number

A_side_instance, A_side_class

The instance on the active side of the Binary Association.

P_side_instance, P_side_class

The instance on the passive side of the Binary Association.

Rnum

A Binary Association.

Domain

Partial reference to the Population, Binary Association, and both participating Instances.

Population

Both Instances are members of this Population.

Identifiers

1> Number + Rnum + Domain + Population

Links are numbered uniquely within an Association Population.

2> A_side_instance + A_side_class + P_side_instance + P_side_class + Rnum + Domain + Population

There may only be one Link between the same two Instances participating in a Binary Association.

RELATIONSHIPS

R306

Instance REFERENCES FROM ACTIVE SIDE zero, one or many Instance
Instance REFERENCES FROM PASSIVE SIDE zero, one or many Instance

A Non Reflexive Link consists of two Instances referring to one another across a Binary Association. There are necessarily two Instances since each is in a different Class (the definition of non-reflexive). Consider a Class named "A". An Instance in A (Ai) may refer to any number of Instances in other Classes assuming the corresponding Binary Associations and side multiplicities are respected. Here are some cases to consider:

Class A is is not attached to any Binary Association (viewed from either an Active or Passive Perspective). In this case, no instance Ai is referenced or references through a Non Reflexive Link.

Class A participates in a single Binary Association with Class B where Class A is on the active side with a multiplicity of 0..* and B is on the passive side with a multiplicity of 1. In this case, an Ai may reference any number of instances Bi. Each possible configuration of multiplicities on the Binary Association will similarly constrain the potential for zero, one or many references between the active and passive sides.

Class A may participate in many Binary Associations with Classes Br, each on the opposite side of a distinct Binary Association.

No two instances Ai and Bi may simultaneously participate in more than one Non Reflexive Link between each other on the same Binary Association.

Shlaer-Mellor modelers may note that this constraint precludes the many association class feature in the OOA96 standard. This constraint does conform with the OMG UML standard, however.

Population

As described in the introduction to this subsystem, the set of user data at the M0 (value) layer (objects, instances, links, attribute values and current states) constitutes a model's Population. During testing or simulation, a Population may be created to define a scenario. For deployment, a persistent Population may be loaded at initialization. Or a Population may serve as a saved snapshot of a runtime model session.

Validation of a Population against a given M1 user model is established through the associations between Population Subsystem model elements and non-Population model elements. The non-Population elements in this case are Association, Perspective, Class, etc. which relate to Population subsystem elements such as Instance and Link. If a Population is out of date with respect to its M1 layer, it won't be possible to load the Population without errors or going through some sort of migration process.

Consider an example where you have a file of data values intended to initialize your Air Traffic Control (ATC) models with names, stations, control zones, etc. But after creating this file, you make a structural change to the ATC model. An attribute is removed, let's say. If you try to load the old population file against this updated model, an error will be triggered when no Attribute object is found for the corresponding data items in the file.

The point here is that the M1 model instantiated into the M2 metamodel validates the M0 population data.

Attributes

Name

This can describe a scenario, test, or other purpose for a persistent set of instances. A name like "Busy Airport 1" could be an example in the ATC example.

Type: Long_name

Domain

The Population is loaded into this Domain. It represents the M1 layer user model.

Identifiers

1> Name + Domain

A Population's Name must be unique for a Domain.

RELATIONSHIPS

R301

Population DEFINES SCENARIO FOR exactly one Domain
Domain HAS SCENARIO DEFINED BY zero or many Population

Since a Population defines M0 elements loaded into a particular user model, there can only be one Domain into which it fits. How about multiple versions of the same Domain? Since configuration management is outside the scope of the miUML metamodel (that's a whole other domain itself!), we assume that there is only one version at a time present of an existing Domain. (It's your responsibility to ensure that the correct Domains and Populations are checked out).

A Domain may have defined and be compatible with any number of Populations, each describing a different user scenario. In the ATC example, there may be a minimal scenario with zero aircraft flying around and all the air traffic controllers idle, and another with multiple aircraft handled by overwhelmed staff.

Real Value

The real number data value stored as part or all of an Attribute Value.

Attributes

Field, Type, Instance, Attribute, Class, Domain, Population

Full reference to the Field Value superclass.

Value

The data stored in the Field.

Type: Real

Identifiers

1> Field + Type + Instance + Attribute + Class + Domain + Population

Same as the superclass identifier.

Relationships

None.

Reflexive Link

A Reflexive Link connects an Instance of a Class to either the same or a different Instance within the same Class. Regardless of type, a Link has two points of reference, each corresponding to an Instance, and possibly the same Instance. On an Asymmetric Link, these Instance(s) correspond nicely to Active and Passive Perspectives. Thus, we can discriminate the Instance on the active or passive side, which again, could be the same Instance.

But on a Symmetric Link, the Active and Passive Perspectives are named identically, e.g. "Territory borders Territory". In such a case the references will be assigned arbitrarily. Thus we can still select an Instance on the active or passive sides. This does not pose a problem since a Symmetric Link is, by definition, symmetric. If the perspective direction mattered, a Binary Association would have been used instead. In the territory example "Brazil" borders "Venezuela" means the same in reverse. A special identity constraint, however, is imposed on the Symmetric Link class (described there).

Attributes

Number

Link.Number

A_side_instance, P_side_instance

The Instances on the active and passive sides of the Association. If the Association is symmetric, these are assigned arbitrarily. These may both be the same Instance unless the Association is constrained to preclude cycles.

Class

Both Instances belong to this Class.

Rnum

An Association.

Domain

Partial reference to the Population, Association and both Instances.

Population

Both Instances are members of this Population.

Identifiers

1> Number + Rnum + Domain + Population

Links are numbered uniquely within an Association Population.

2> A_side_instance + P_side_instance + Class + Rnum + Domain + Population

There is at most one Link between the same two Instances participating in an Association.

Relationships

R305
Instance REFERENCES FROM SAME CLASS ON ACTIVE SIDE zero, one or many Instance
Instance REFERENCES FROM SAME CLASS ON PASSIVE SIDE zero, one or many Instance

Any Association that connects to and from the same Class is reflexive. A Reflexive Link consists of two Instances in the same Class referring to one another across a reflexive Association. Consider a Class named "C". An Instance in C (Ci) may refer to any number of Instances in C including itself assuming the corresponding Associations and side multiplicities are respected. Here are some cases to consider:

Class C is is not attached to any reflexive Association. In this case, no instance Ci is referenced or references through a Reflexive Link.

Class C participates in a single reflexive Association with an active side with a multiplicity of 0..* and a passive side with a multiplicity of 1. If Ci is on the passive side, it may reference any number of instances including Ci. Other possible multiplicities on the reflexive Association willconstrain the potential for zero, one or many references between the active and passive sides.

Class C may participate in many reflexive Associations on C. Consequently two instances Ci and Cj may be linked reflexively multiple times simultaneously, but only once for each reflexive Association. If cycles are allowed in each reflexive Association, Ci could be linked to itself multiple times, again, only once per Association at the same time.

String Value

The text data value stored as part or all of an Attribute Value.

Attributes

Field, Type, Instance, Attribute, Class, Domain, Population

Full reference to the Field Value superclass.

Value

The data stored in the Field.

Type: String

Identifiers

1> Field + Type + Instance + Attribute + Class + Domain + Population

Same as the superclass identifier.

Relationships

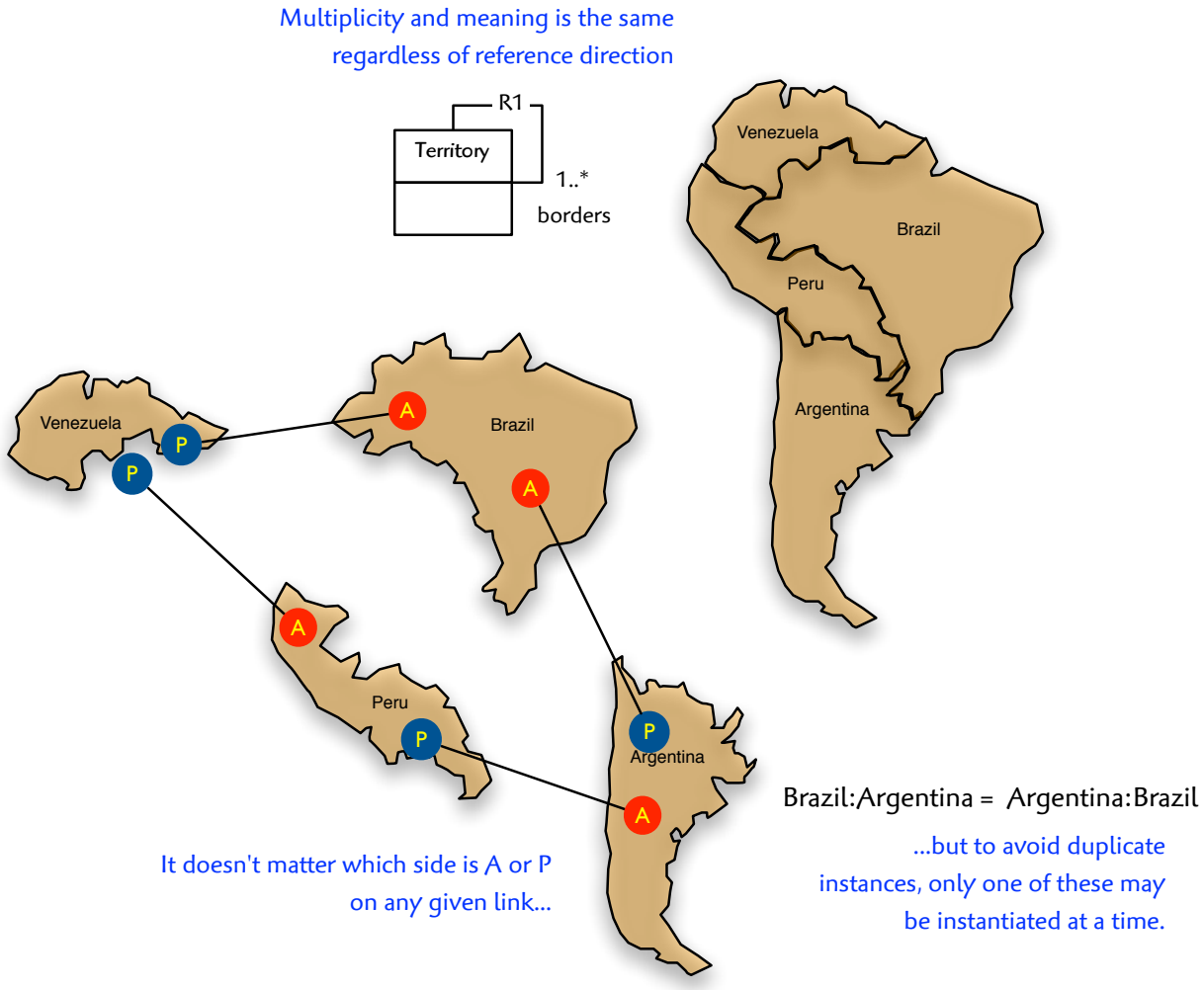
None.

Symmetric Link

This type of Link is symmetric since each side corresponds to an identical Perspective.

Consider two example Instances "Brazil" and "Argentina" linked to one another on the Unary Association "borders 1..*". We can say that "Brazil borders Argentina" and that "Argentina borders Brazil" with equivalent meaning. There are still two sides to the Link, but the meaning is the same regardless of which way you look. There are two perspectives, stated only once in the Relationship Subsystem since they are identical in both name and multiplicity. Thus the active and passive sides are assigned arbitrarily, but with an important identity constraint - see the Identifiers section below.

Symmetry, Perspectives and a special identity constraint



Standard UML does not support this kind of symmetry. When converting miUML to standard UML just copy the verb phrase and multiplicity to the opposite side of the reflexive Association.

Attributes

A_side_instance, P_side_instance, Class

Partial reference to the Reflexive Link superclass.

Number, Rnum, Domain, Population

Partial reference to both the Reflexive Link and Binary Link superclasses.

Identifiers

1> Number + Rnum + Domain + Population

Links are numbered uniquely within an Association Population.

2> A_side_instance + P_side_instance + Class + Rnum + Domain + Population

IMPORTANT: No two instances may exist such that [A, P] = [P, A] where A is the A_side_instance and P is the P_side_instance. In the "Territory" example, "Brazil" borders "Argentina" is the same instance as "Argentina" borders "Brazil", so only one may exist at a time.

Relationships

R314
Symmetric Link INSTANTIATES one Unary Association
Unary Association IS INSTANTIATED BY zero, one or many Symmetric Link

Instantiation refers to the relationship of M1 (user model) entities to M0 (user data) entities in the miUML metamodel. The Unary Association instance is the M1 element with respect to the Symmetric Link instance M0 element. The Unary Association can exist without being instantiated, but a Symmetric Link cannot exist independently of its specification, a Unary Association.