DDI Documentation

Release 4.0 dev

DDI

Aug 22, 2018

Table of contents

Warning: this is a development build, not a final product.

CHAPTER 1

About

1.1 DDI Background

DDI Is a metadata standard for describing data files and events sorounding the creation of the material.

Published versions of DDI

Version	Year published	Documentation link
2.1	2003	http://www.ddialliance.org/Specification/DDI-Codebook/2.1/DTD/Documentation/DDI2-1-tree.html
2.5	2012	http://www.ddialliance.org/Specification/DDI-Codebook/2.5/XMLSchema/field_level_documentation.html
3.0	2008	http://www.ddialliance.org/Specification/DDI-Lifecycle/3.0/XMLSchema/Documentation/
3.1	2009	http://www.ddialliance.org/Specification/DDI-Lifecycle/3.1/XMLSchema/FieldLevelDocumentation/
3.2	2012	http://www.ddialliance.org/Specification/DDI-Lifecycle/3.2/XMLSchema/FieldLevelDocumentation/

1.2 History

Adding short text about the history of DDI.

CHAPTER 2

Introduction

2.1 Description of the Model

2.1.1 Introduction

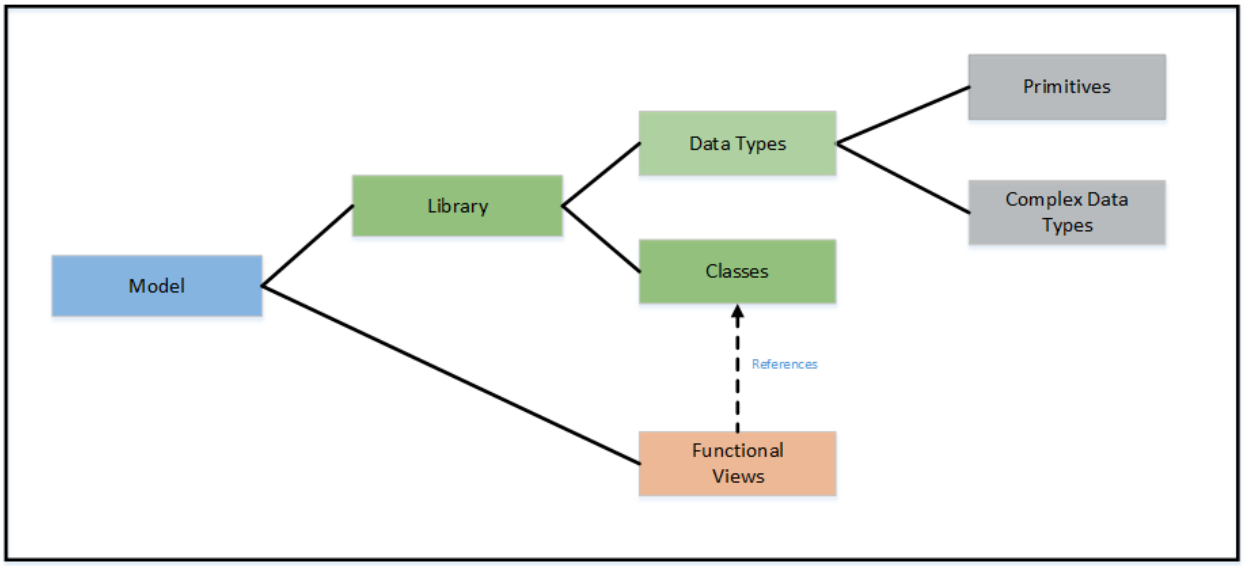
The intent of the Model-Based DDI Specification Class Description is to provide information on the individual classes used in the model, their relationship to each other and their relationship to DDI Lifecycle 3.2 and other standards such as General Statistical Information Model (GSIM). The model based DDI specification consists of two parts – a Library of classes and Functional Views of the model. The Library of classes encompasses the entire DDI-Lifecycle (MD) model. The classes in the Library are the building blocks used to construct the Functional Views. These Functional Views are in essence profiles of the full specification oriented around specific user needs. The model is primarily being developed and surfaced to the user community at http://lion.ddialliance.org/

A Note on Terminology During the development process, the terminology for what is now called classes (to reflect the language used in UML) was previously referred to as 'objects'

2.1.2 Structure of the DDI-Lifecycle (MD) Model

The model contains a Library and Functional Views. The Library is composed of Library Packages which contain other data types (primitives or complex) or classes. The Functional Views contain references to the classes used by the particular Functional View that are needed to meet the needs of the use case or business application.

Figure 1. DDI-Lifecycle (MD) Model and its components



2.1.3 Library of Classes

The Library of Classes encompasses the entire DDI-Lifecycle (MD) model, but without any specific schemas or vocabularies for Functional Views. The classes in the Library contain primitives and complex data types and are the building blocks used to construct the Functional Views. Classes are organized into Library Packages.

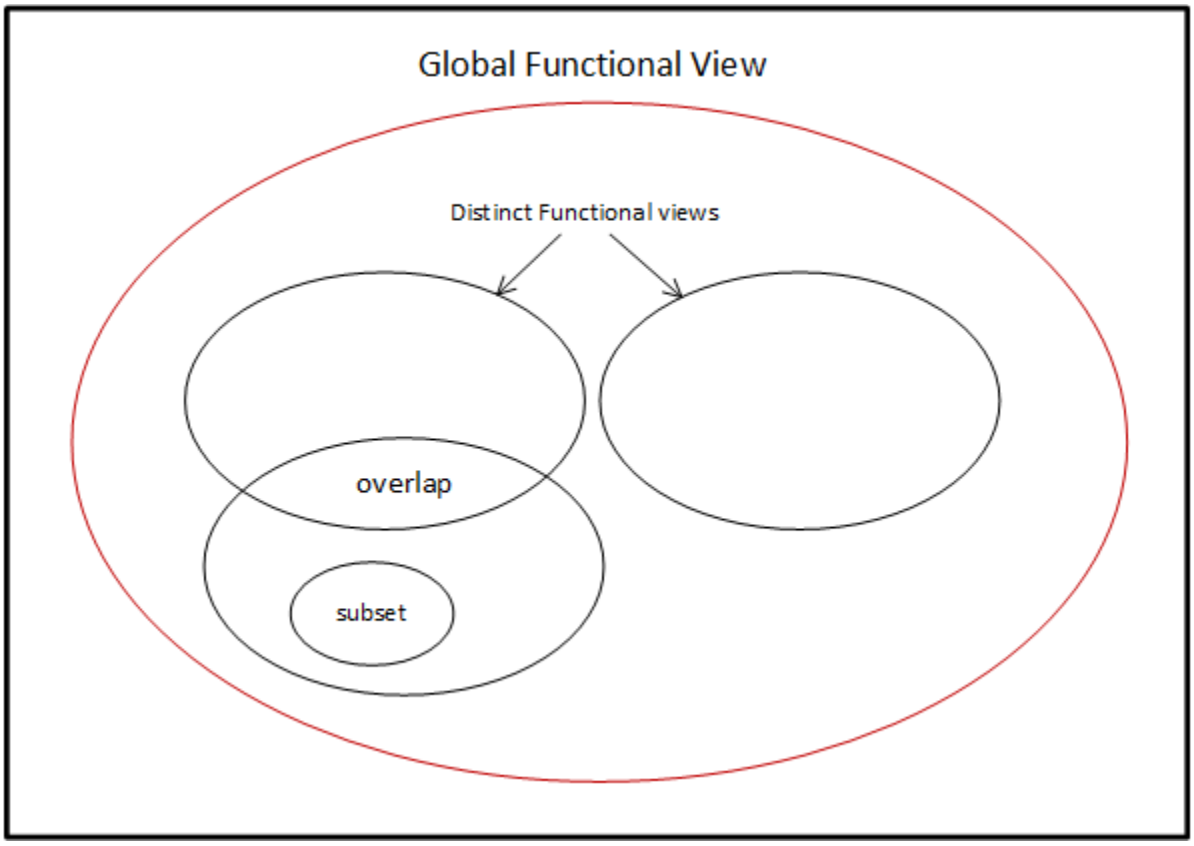
2.1.4 Functional Views

Functional Views are made up of a set of references to the classes in the Library. Functional Views are subsets of the model grouped to support a specific application (for example the description of a questionnaire). The Functional Views are divided into sections. Each section loosely corresponds to a DDI lifecycle business area. Within each business area section there are separate subsections for Functional Views and compositions. Note that Functional Views may include placeholders like an abstract class that need to be substituted before the Functional View can actually be used. Functional Views are always a strict subset of the existing published or (for customization) extended classes. A Functional View identifies a set of classes that are needed to perform a specific task. It primarily consists of a set of references to specific versions of classes. Functional Views are the method used to restrict the portions of the model that are used, and as such they function very much like DDI profiles in DDI 3.*. The subsetting with Functional Views is done on the model and not on the instance level as in DDI Profiles. One may

- restrict the use of non-mandatory properties on a class;
- restrict the cardinality of a class's relationships and properties;
- restrict the use of non-mandatory relationships.

Restrictions may never be made that would violate the mandatory inclusion of a relationship or property. Functional Views may combine classes from any package or set of packages needed. The creation of Functional Views thus has no dependency on the organization of metadata classes within the Library Packaging structure.

Figure 2. Interoperability of Functional Views



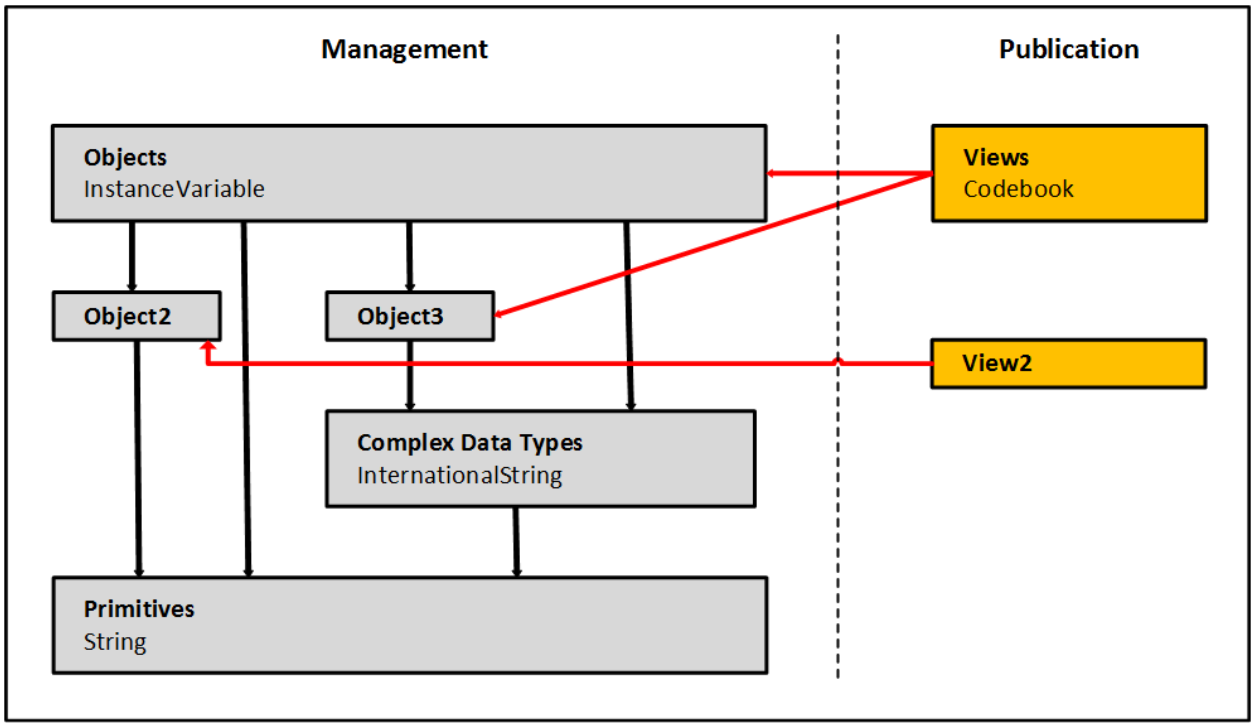
As shown in Figure 2, Each Functional View is a subset of the classes in the Library. Functional Views might be distinct, overlapping in their function or be a subset or superset of another Functional View. Interoperability between two Functional Views is only given for the Library classes which are used in both Functional Views.

A global Functional View could be created which comprehends all classes in the Library and their relationships. It represents all functionality of the class in the Library. Each Functional View would be interoperable to this global Functional View.

2.1.5 Model Constructs and Their Relationships

Figure 3 below shows the basic relationships between the types of constructs in the model. At the lowest level, we have the primitives. These are used directly by classes, and are also used to create complex data types. The complex data types are also used by classes. Classes themselves can relate to other classes, building increasingly complex structures. The classes – along with the primitives and complex data types – form the Class Library. Classes can relate to each other in two ways: a class may have a "direct" relationship (composition, aggregation) with another class, or it may have an inheritance relationship. In this latter case, the DDI model uses additive extension. One class may extend another by inheriting all of its properties and relationships, to which the new class may add additional properties and relationships. This mechanism is used to take more generic classes and alter them for a more specific purpose. Extension is explained more fully below.

Figure 3. DDI-Lifecycle (MD) Organization of the Model

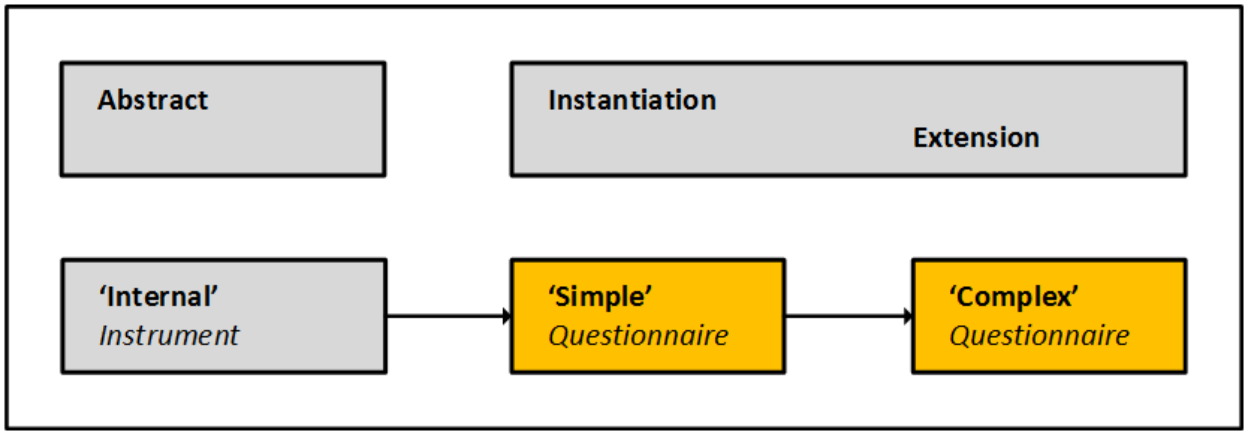


2.1.6 Extension

Extension is the inheritance of one class's properties and relationships from another class. It also has a semantic relationship – an extending class provides a specialized use of the extended class.

Extensions are used within the DDI-published Library Packages to provide relationships between classes as they increase in complexity to meet increasingly complex functionality. Thus, a "simple" version of a questionnaire class might be extended into a more complex class, describing a more complex questionnaire. Some classes exist only for the purpose of extension, and are declared abstract. A Functional View may never include an abstract class. Non-abstract classes may never have direct relationships with abstract classes. Extension is illustrated in Figure 4 below.

Figure 4. Extensions in DDI-Lifecycle (MD)



Here, an abstract class – Instrument, which is any tool used to collect data – is extended by Simple Questionnaire, which is itself extended by Complex Questionnaire. As we proceed through this chain of extension, the classes have increasingly large numbers of properties and relationships.

For example, if an Instrument class has a name property, a description property, and an ID property, these would be inherited by Simple Questionnaire, which might add a relationship to one or more Question classes. The Complex Questionnaire in turn might add a relationship to a Questionnaire Flow class, to add conditional logic to the questionnaire.

The second use of extension in the DDI model is to allow users to add needed metadata fields for the purposes of customization. Thus, a specific user community may decide to have a standard set of additional properties, classes, and relationships and create their own model Library Package which contains classes extending the classes in the DDI-published Library Packages. The creator of the extensions is the owner and maintainer of the extended classes and Library Packages – this is not the business of the DDI Alliance.

Extension in DDI is strictly defined: you are able to add new properties to existing classes, and add new relationships to existing classes. Extension is always done on a class which is referenced and inherited from: that is, a new class is declared which inherits all the properties and relationships of an existing class. New properties and relationships are then declared for it. Extension is always additive extension. There is no concept of refinement – that is handled using Functional Views.

Those creating their own custom Library Packages based on extensions to the DDI model may also declare entirely new classes which are not extension of DDI classes.

Extensions made by those customizing the DDI model are expressed using the same modeling techniques and information that are used for the development of the model published by the DDI Alliance itself. As a result of this, the same tools for the creation of documentation and syntax artefacts (XML schemas, RDF vocabularies) could potentially be used.

2.1.7 Managing the Library

In order to manage the Library effectively, the classes, together with primitives and complex data types, are grouped into Library Packages. The organization of Library Packages is currently flat. As development continues and the number of Library Packages increases the DDI model may be organized in a hierarchy of Library Packages arranged according to the types of constructs.

Library Packages are mutually exclusive and comprehensive. They are organic entities with a logical organization and are labeled in an accessible way so that developers and modelers can easily understand their content. They are stable and should not be changed often.

Provisional Library Organization

- Core
  - Primitives
  - Complex Data Types
  - Identification and versioning
  - Grouping and Comparison
  - Utility classes
- Conceptual
  - Universe, concept, category unit
  - Representations, code lists, classifications
  - Represented and conceptual variables
- Study

  - Study info
  - Study inception
  - Collection
  - Archiving and preservation
  - Access and discovery
- Data
  - Logical data structures
  - Physical data structures
  - Datasets
  - Instance variables (raw and derived variables)
- Process
- Geography
- Instrument and data source
  - Questionnaires, routing
  - Access to administrative data
  - Questions, items
- Methodology
  - Data transformations e.g. formulas

2.1.8 Versioning the Library

The classes within each Library Package as well as Functional Views are versioned. The whole model has a specific release date that acts as part of its identification. The Library Packages are versioned primarily for maintenance purposes

The versioning rule is that if the contents of a versioned class change, it is versioned. This means that versions "trickle up" – a new class is added to a Library Package, which versions the Library Package; the new version of the Library Package can drive a new release of the Model, and so on.

However, if a class does not change, its version does not change, even if the Library Package within which it lives in is versioned. Once published, a class is always available for use within Functional Views, even if it is not the latest version of the class. (If the old version of a class is good enough, it is still available for use in a new version of a Functional View, etc.) Once published, classes are never removed from the Library. All published classes and Functional Views will be available in the model forever

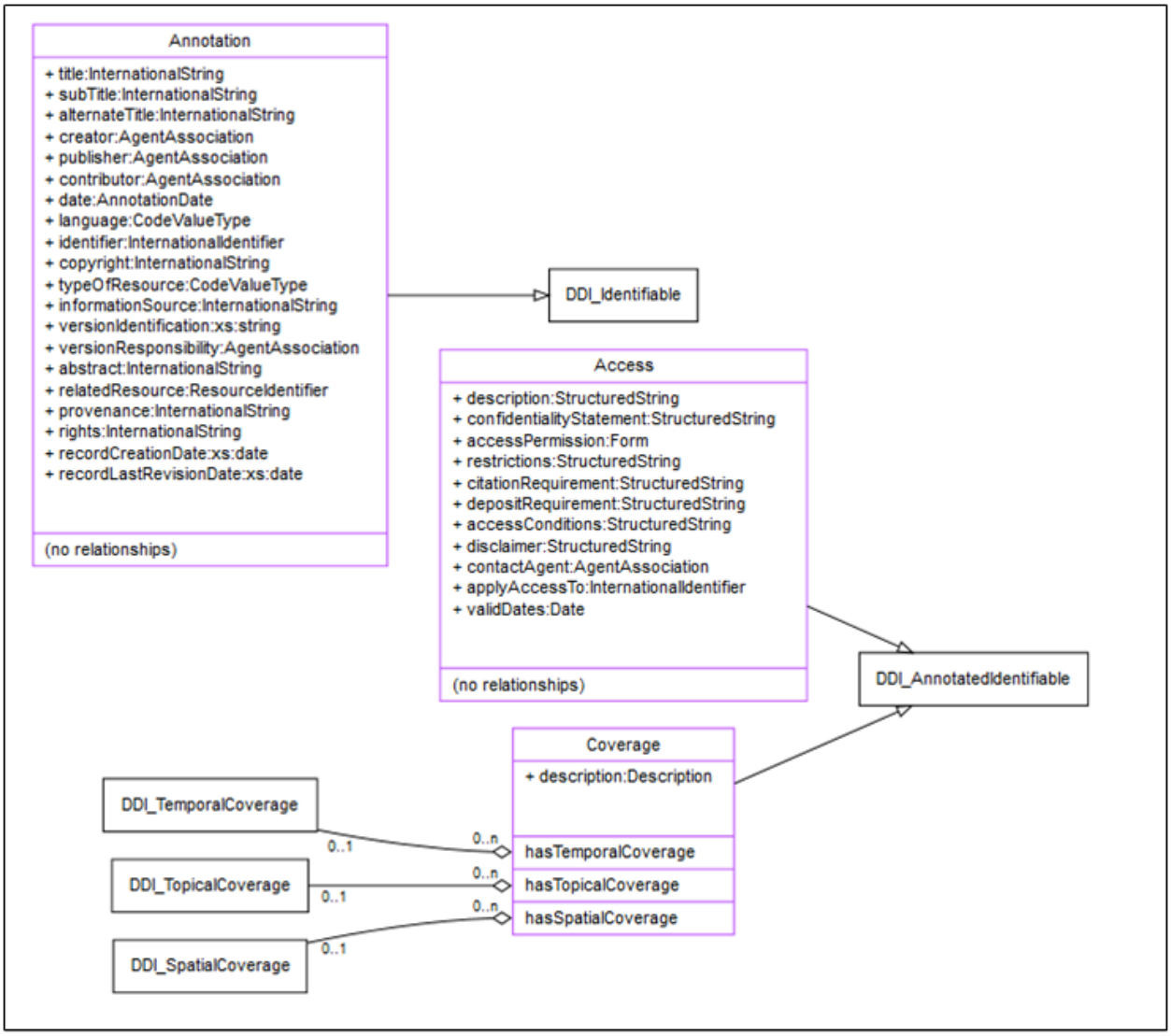
This has the effect of de-coupling the dependencies created by the use of extensions to add new things to the model. Decisions about what release packages consist of are driven by the needs of users and marketing considerations, and not by the chain of dependencies between classes, Library Packages, etc.

It is foreseen that at least initially, the Library will be released alongside sets of useful Functional Views, but incremental releases are possible without causing problems – a new version of the Library is released, but it will always contain all classes already in use.

2.1.9 Example of a Functional View

Figure 5 shows a diagram of the initial Discovery View, which includes the Access, Annotation and Coverage classes. Access and Coverage inherits from the AnnotatedIdentifiable class, while Annotation inherits from the Identifiable class. Coverage has aggregation relationships to TemporalCoverage, TopicalCoverage and SpatialCoverage.

Figure 5. Example Functional View

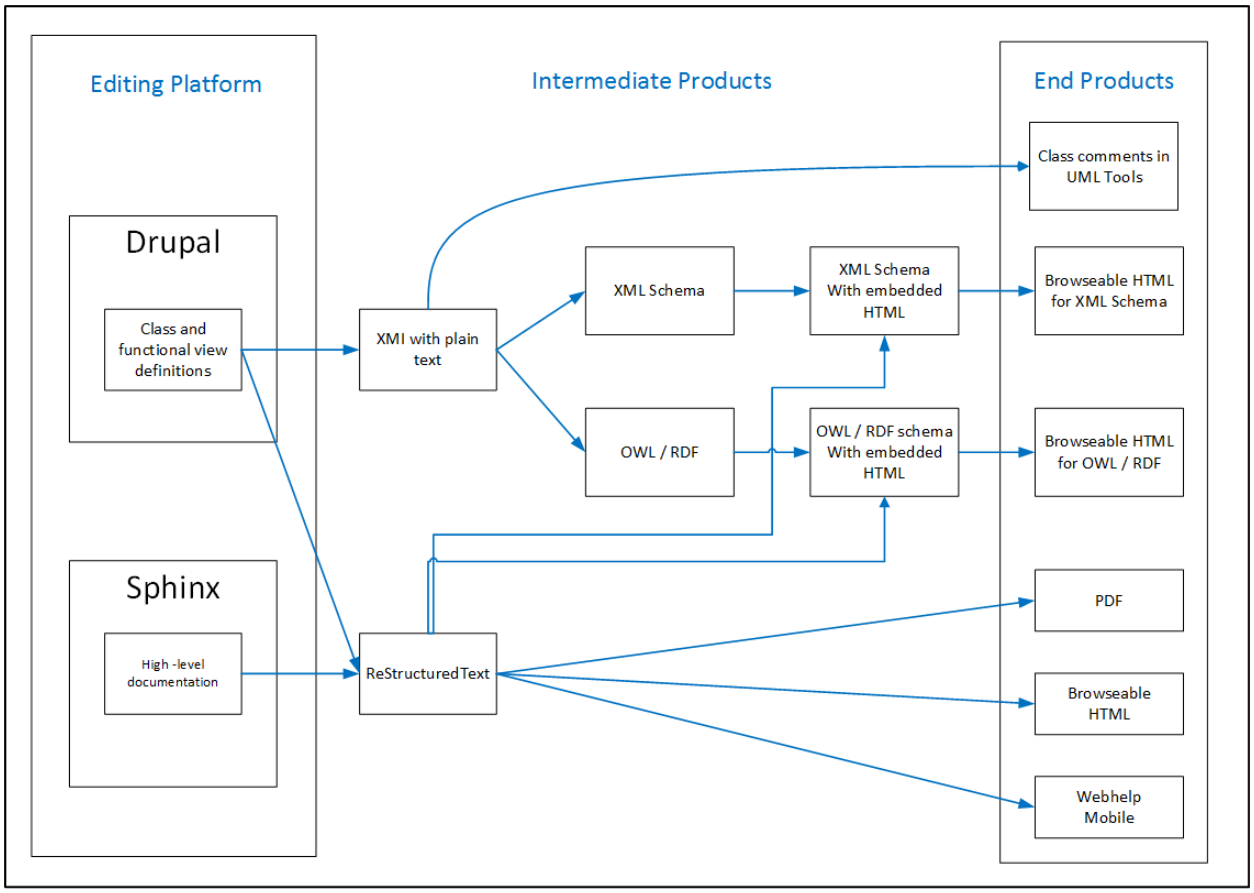


2.2 Production Framework

The model is being developed in Drupal at http://lion.ddialliance.org

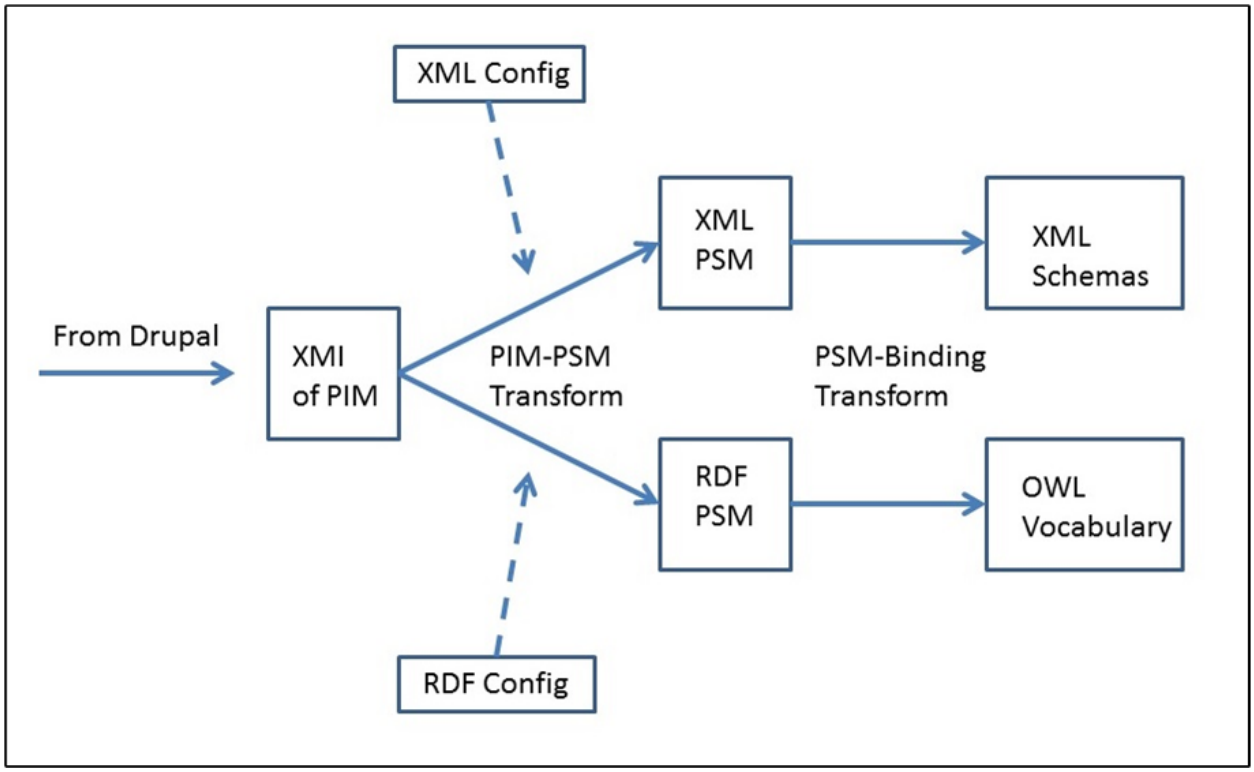
2.2.1 Documentation Flow

Figure 1. Documentation Flow in the Production Process



2.2.2 Bindings Production Flow in the Production Framework

Figure 2. Bindings Flow in the Production Process



The XMI for a portion of the model (as configured in Drupal) is exported as XMI for the platform-independent model (PIM). This is then transformed again, for each binding type (XML or RDF) to be produced, creating the platform-specific model (PSM) specific to that binding. This transformation from PIM to a specific PSM is informed by any needed configuration information. The PSM is optimized for the expressive capabilities of the binding to be produced (RDF and XML have different expressive capabilities). A transformation is then run on each PSM to produce the desired bindings. Each view will be expressed as an XML document type and as an RDF Vocabulary expressed in OWL. There will also be an XML and OWL binding for the library as a whole.

2.3 Design Principles

A set of design principles has been identified during the course of the development of DDI4, The list is shown below:

Principle	Definition
Interoperability and Standards	The model is optimized to facilitate interoperability with other relevant standards
Simplicity	The model is as simple as possible and easily understandable by different stakeholders
User Driven	User perspectives inform the model to ensure that it meets the needs of the international DDI user community
Terminology	The model uses clear terminology and when possible, uses existing terms and definitions
Iterative Development	The model is developed iteratively, bringing in a range of views from the user community
Documentation	The model includes and is supplemented by robust and accessible documentation
Lifecycle Orientation	The model supports the full research data lifecycle and the statistical production process, facilitating replication and the scientific method
Reuse and Exchange	The model supports the reuse, exchange, and sharing of data and metadata within and among institutions
Modularity	The model is modular and these modules can be used independently
Stability	The model is stable and new versions are developed in a controlled manner
Extensibility	The model has a common core and is extensible
Tool Independence	The model is not dependent on any specific IT setting or tool
Innovation	The model supports both current and new ways of documenting, producing, and using data and leverages modern technologies
Actionable Metadata	The model provides actionable metadata that can be used to drive production and data collection processes

Additional lower-level principles have surfaced during initial DDI model development:

Principle	Definition
Remodelling Discouraged	The model leverages existing structures in the specification whenever possible to avoid inefficiencies
Classes Represent Actual Things	The model includes classes that are functional and are used
Separation of Logical and Physical	The model supports a distinction between logical and physical representations
Names are Mutable	The model contains names and labels that may change to encourage accessibility
Common Expressions	The model will only have features that reflect the common expressive capabilities of supported syntaxes/technologies (e.g., no multiple inheritances)

2.4 DDI Base Blocks

2.4.1 Complex Data Types

These are extensions of base type Primitives All complex data types (that is, the set of complex structures which are treated within the Drupal modeling platform as primitives, as for the values of properties) are located in the Complex Data Types package. There is a distinct style of modeling these: each complex data type which has a primary content will have a property named “content” of whatever primitive type is needed. Complex data types will not be extensions of the primitive type of their primary content. [A set of complex structures which are used as datatypes for properties within a class description. There is a distinct style of modeling these: each complex data type which has a primary content will have a property named “content” of whatever primitive type is needed. Complex data types will not be extensions of the primitive type of their primary content. This allows standard structures, such as capturing structured language strings, to be expressed in a single way throughout DDI.]

2.4.2 Relationships

DDI classes are associated via binary relationships. Relationships have cardinality constraints, e.g. 1..1, 1..n, 0..n, and can be of different types, i.e. aggregation, composition, and neither (simple, untyped). Even though most relationships in the model are undirected, in the Drupal they are always defined within one of the classess participating in the relationship, i.e. the source; the class at the other end of the relationship is called the target. Similarly with names: the predicate represented in a relationship name does not impose a direction since the implicit converse predicate is also true in all undirected relationships. It's important to note that this is just a convention used in the Drupal and by no means imposes an actual conceptual direction in the association. For instance, RepresentedVariable has a relationship with ValueDomain. The relationship is defined in ValueDomain, which is the source, and it is named by a predicate, i.e. takesValuesFrom. This seems to imply a direction from RepresentedVariable to ValueDomain. However, the converse predicate, i.e. providesValuesFor, although not represented in the model, is also valid since such a relationship is conceptually undirected. In general, and unless otherwise indicated, all relationships in the Drupal are undirected.

2.4.3 Identification (Identifiable class)

The Identifiable class is core to the functioning of the standard. The purpose of the DDI Identifiable class is to: * Uniquely identify major objects in a persistent manner to ensure accurate reference and retrieval of the object content * Provide context for classes where an understanding of related class types within a packaging structure is essential to the understanding of the class (i.e., a specific classification within a classification scheme) * Manage metadata object change over time * Support the range of object management used by different organizations * Support early and late binding of references * Support interaction with closely related standards, in particular ISO/IEC 11179 and SDMX

Many organizations may have preexisting URI schemes, or have URI patterns imposed on them other organizations or governments. Unlike DDI3.x DDI4 will not require any specific information or pattern to be contained in the URIs of described resource.

To align with both DDI 3.x and ISO/IEC 11179-6, the Identifiable object will continue to be based on a combination of: * Agency Identifier (a unique identifier for the agency managing the object) * Item Identifier (a unique identifier of the object within the context of the agency) * Item Version (a version number of the object to track change over time)

These parts correspond to the agency, id, and version used in DDI 3.x and to the registration authority identifier (RAI), data identifier (DI), and version identifier (VI) constituting the international registration data identifier (IRDI) in ISO/IEC 11179-6

In DDI 3.x, all items had an agency, item id, and version. However, some types of items could inherit a parent item's agency. Some items would inherit a parent item's version. In DDI 4, all items will have their own Agency, Item Identifier, and Item Version specified. This three part structure is the equivalent of a unique persistent identifier for an object, such as described by DOIs and other similar structures. Note that while use of a version number with a DOI is optional, based on local practice, the Version Number in DDI is required due to the need to manage metadata within a dynamic workflow over time

2.4.4 Character Restrictions

In DDI 3.x, regular expressions restricted the Agency Identifier, Item Identifier, and Item Version. This has been removed for DDI 4. The only restrictions are that it is a string without colons and whitespace. Note that these are restrictions on the specific content not the structure of a DDI URN. The restriction on the use of a colon supports the use of this character as a URN separator. This complies with ISO/IEC 11179-6 as it imposes no limitations on the contents of the IRDI fields. In DDI 3.2, versions are restricted to integers that may be separated by periods. This forces implementers to use a specific versioning system. A more flexible system would use a "based on" reference to determine version history. In addition, a "based on" system adds the ability to branch and merge. A Based On system would be backwards compatible with DDI 3.x versioning systems.

2.5 Simple example

For documenting a dataset the content could look something like this.

ID	subj	q1
0	Anders	3
1	Lars	2
2	Krister	3

2.5.1 A DDI Example

```
<DDI>
 <DataFile>
   <path>ct-2015.survey.csv</path>
   <haveLogicalDataset>dataset1</haveLogicalDataset>
 </DataFile>
 <LogicalDataset id="dataset1">
   <hasVariable>ID</hasVariable>
   <hasVariable>subj</hasVariable>
   <hasVariable>q1</hasVariable>
 </LogicalDataset>
 <Variable id="ID">
   <Label>The record id of the collection</Label>
   <Type>Numreric</Type>
 </Variable>
 <Variable id="subj">
   <Label>Person interviewd</Label>
   <Type>String</Type>
 </Variable>
 <Variable id="q1">
   <Label>Relation to cats</Label>
   <hasCodeList>c_list_1</hasCodeList>
 </Variable>
 <CodeList id="c_list_1">
   <hasCode>c2</hasCode>
   <hasCode>c3</hasCode>
 </CodeList>
 <Code id="c2">
   <value>2</value>
   <Label>Yes</Label>
 </Code>
 <Code id="c3">
   <value>3</value>
   <label>No</label>
 </Code>
</DDI>
```

CHAPTER 3

Use Cases

Contents:

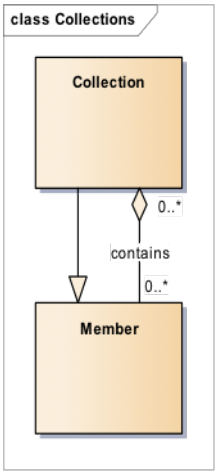
CHAPTER 4

User Guides

4.1 Using the Collection pattern

DDI-Views introduces a generic Collection pattern that can be used to model different types of groupings, from simple unordered sets to all sorts of hierarchies, nesting and ordered sets/bags. A collection is a container, which could be either a set (i.e. unique elements) or a bag (i.e. repeated elements), of Members. Collections can also be extended with richer semantics (e.g. generic, partitive, and instance, among others) to support a variety of DDI 3.x and GSIM structures, such as Node Sets, Schemes, Groups, sequences of Process Steps, etc. Collections together with their related classes provide an abstraction to capture commonalities among a variety of seemingly disparate structures. A Collection consists of zero, one or more Members (Figure 1). A Member could potentially belong to multiple Collections. A Collection is also a Member, which allows for nesting of Collections in complex structures. Members have to belong to some Collection, except in the case of nested Collections where the top level Collection is a Member that doesn't belong to any Collection.

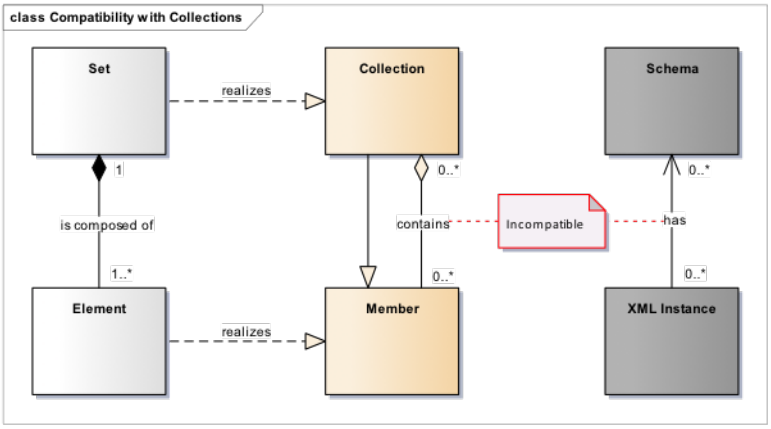
Figure 1. Collections class



This pattern can be used via a special type of association called realizes. DDI-Views uses realizes to say that a class "behaves" like a Collection. For instance, consider a Set that consists of Elements, they implement the Collection

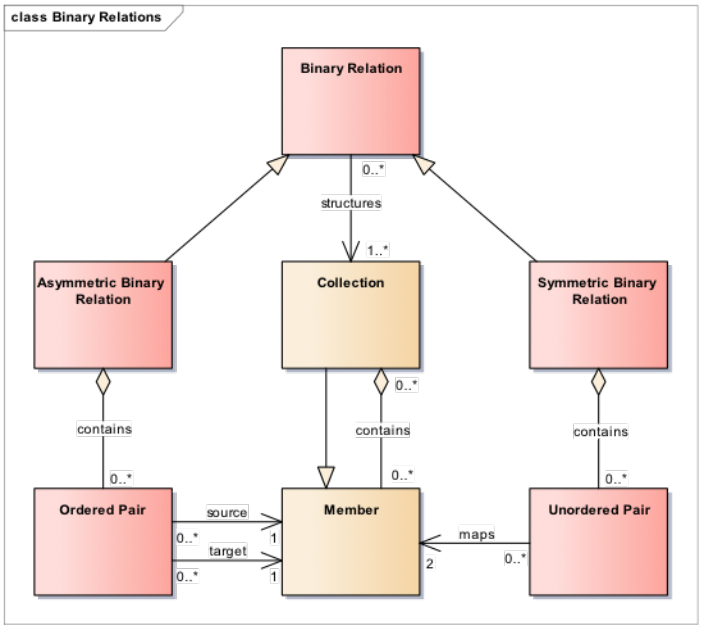
pattern as follows: Set realizes Collection and Element realizes Member. To realize this pattern all classes involved must be associated in a way that is compatible with the pattern. As a rule of thumb, a more restrictive type of association than the one that appears in the pattern is compatible, a looser one is not. For instance, since the collection pattern has an aggregation association (denoted by the empty diamond), classes realizing the Collection pattern need to be related by either an aggregation or a composition, nothing else. In addition, source and target, when applicable, must also match, e.g. the diamond of the aggregation/composition needs to be on the class realizing Collection, not Member. Similar compatibility rules apply to cardinality. Furthermore, all associations must be realized, with the exception of IsA associations, which are usually part of the pattern definition and do not apply to individual realizations in the same way. Renaming associations does not affect compatibility as long as the documentation clearly explains how they map to the association in the pattern. For instance, consider Figure 2.. In this example, a Set class is defined as being composed of at least one Element, i.e. no empty Sets are allowed. In addition, an Element always belong to one and only one Set, which means that deleting the Set will also delete its Elements. Such an association is compatible with the contains association in the Collection pattern and thus Set and Element can realize Collection and Member, respectively. In contrast, Schema and XML Instance cannot realize the pattern: the association is neither an aggregation nor a composition, Schema is not a grouping of XML Instances, and the association points from XML Instance to Schema. None of this is compatible with the Collection pattern, in particular with the semantics of the contains association between Collection and Member.

Figure 2. Compatibility with Collections class



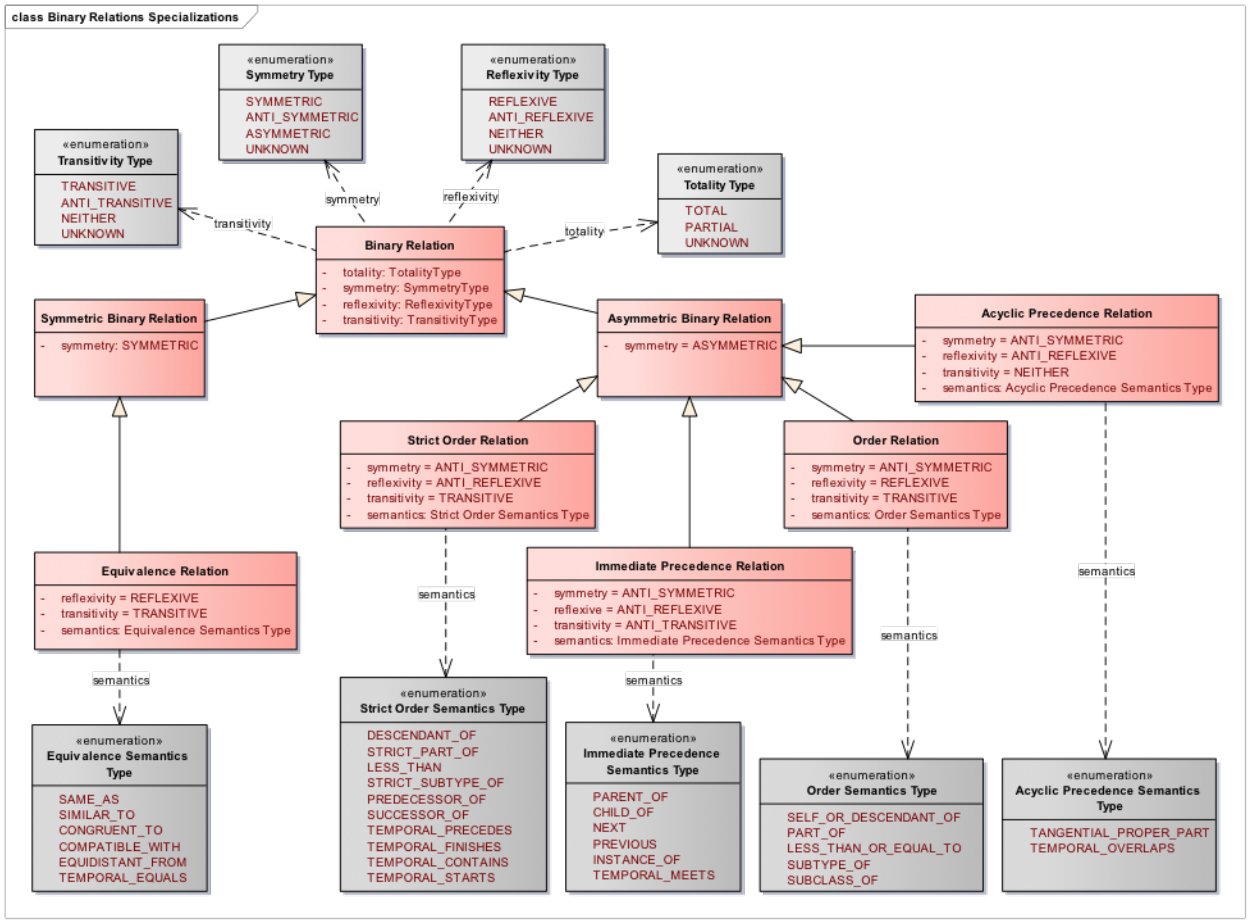
Collections can be structured with Binary Relations, (Figure 3) which are sets of pairs of Members in a Collection. Binary Relations can have different properties, e.g. totality, reflexivity, symmetry, and transitivity, all of which can be useful for reasoning.

Figure 3. Binary Relations



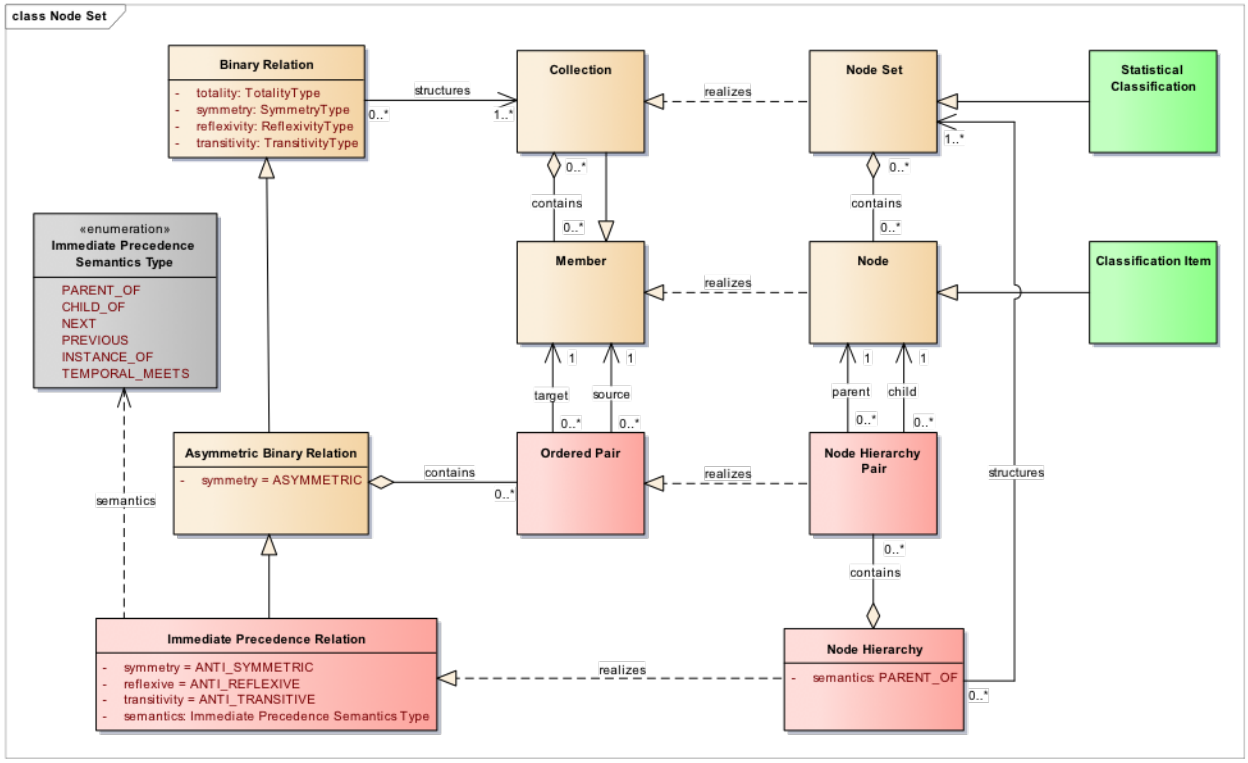
A Binary Relation is said to be symmetric if for any pair of Members a, b in the associated Collection, whenever a is related to b then also b is related to a. Based on this property we define two specializations of Binary Relation: Symmetric Binary Relation, when the property is true, and Asymmetric Binary Relation, when the property is false. Symmetric Binary Relations can be viewed as collections of Unordered Pairs themselves, whereas Asymmetric Binary Relations can be viewed as collections of Ordered Pairs. However, for simplicity, we do not model Relations themselves with the Collection pattern. We can further classify Binary Relations based on additional properties. We say that a Binary Relation is total if all Members of the associated Collection are related to each other. We call it reflexive if all Members of the associated Collection are related to themselves. Finally, we say it is transitive if for any Members a, b, c in the associated Collection, whenever a is related to b and b is related to c then a is also related to c. [Refer to Dan's document on Relations for more details.] These properties can be combined to define subtypes of Binary Relations, e.g. Equivalence Relation, Order Relation, Strict Order Relation, Immediate Precedence Relation, and Acyclic Precedence Relation, among others. Equivalence Relations are useful to define partitions and equivalence classes (e.g. Levels in a Classification). Order Relations can be used to represent lattices (e.g. class hierarchies, partitive relationships), Immediate Precedence Relations can define sequences and trees (e.g. linear orderings, parent-child structures) and Acyclic Precedence Relation can represent directed acyclic graphs (e.g. molecular interactions, geospatial relationships between regions).

Figure 4. Binary Relations specialization



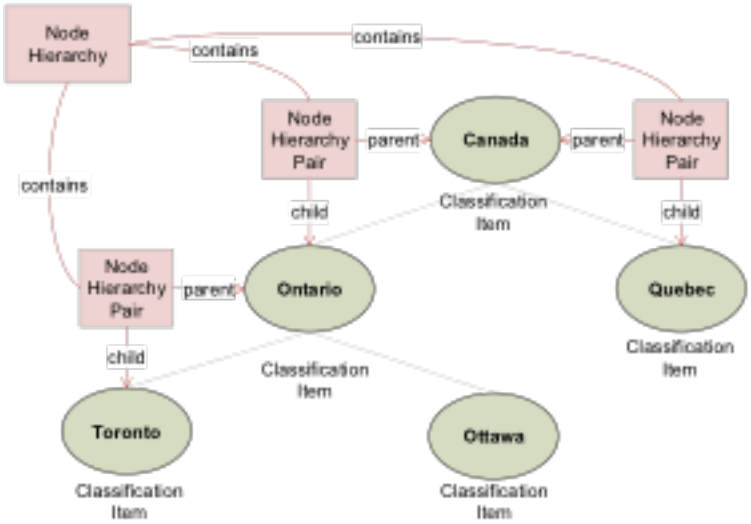
These subtypes can also have various semantics, e.g. Part-Of and Subtype-Of for Order Relations, to support a variety of use cases and structures, such as Node Sets, Schemes, Groups, sequences of Process Steps, etc. Note that some of them include temporal semantics, e.g. Strict Order Relation and Acyclic Precedence Relation. A modeller can use the different semantics types as a guide when trying to decide what type of Binary Relation to realize. For instance, if the new class to be added to the model is a Node Set containing Nodes that will be organized in a parent-child hierarchy, the modeller can define a Node Hierarchy class with PARENT_OF semantics to structure the Node Set. The type of Binary Relation to realize then is Immediate Precedence Relation because it is the one that has the required semantics in its Semantics Type. Alternatively, a modeller familiar with the definitions of the Binary Relation properties, i.e. symmetry, reflexivity and transitivity, could make the choice based on what combination represents the type they are looking for. For instance, a parent-child hierarchy requires the Binary Relation to be ANTI_SIMMETRIC (if a Node is the parent of another, the latter is not the parent of the former), ANTI_REFLEXIVE (a Node cannot be a parent of itself) and ANTI_TRANSITIVE (a Node is not the parent of its children's children). It is easy to see that the only one that satisfies that criteria is the Immediate Precedence Relation. Figure 5 shows an example of the realization of the pattern. We can model Node Hierarchy and Node Hierarchy Pair classes as realizations of Immediate Precedence Relation and Ordered Pair, respectively.

Figure 5. Node Set



Let us illustrate how this model works with a simple instance. Consider a geographical Statistical Classification with Classification Items (Figure 6) representing Canada, its provinces and cities.

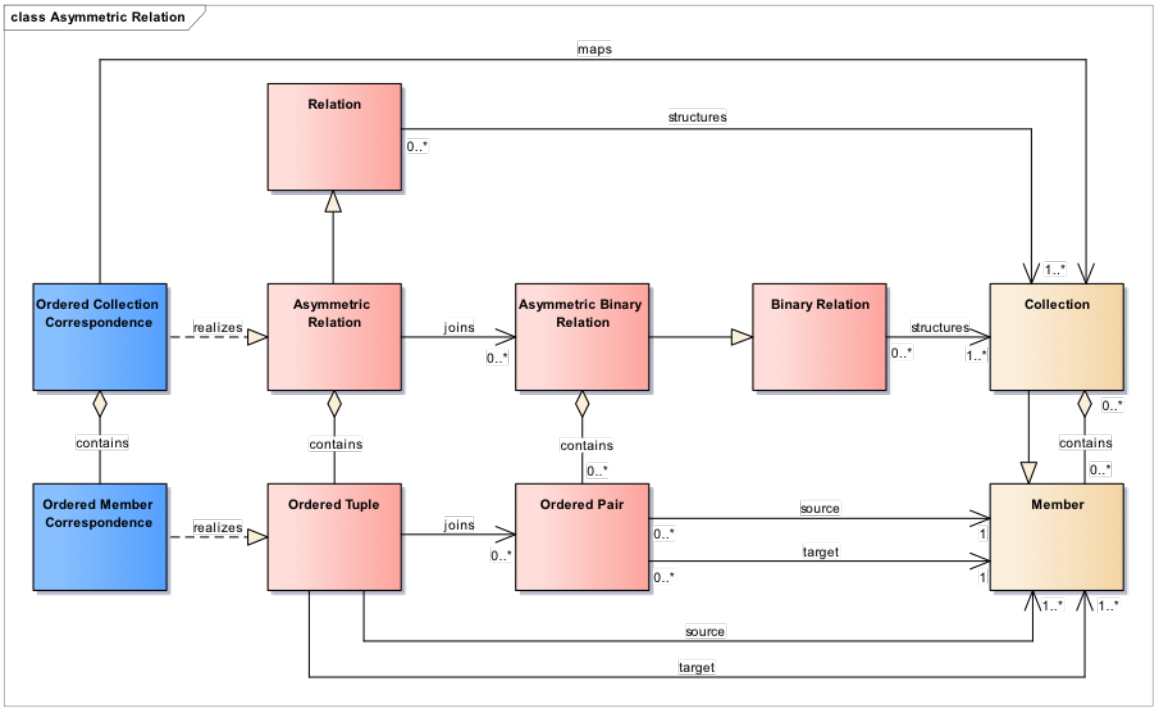
Figure 6. Node Set example



Since Statistical Classifications are Node Sets and Classification Items are Nodes, we can view Classification Items such as Canada, Ontario, Quebec, Toronto, etc. as Members in a Collection structured by a Node Hierarchy Relation. Node Hierarchy Pairs represent the parent-child relationships in the Node Hierarchy Relation. For instance, (Canada, Ontario) is a Node Hierarchy Pair in which Canada is the parent and Ontario is the child. Other Node Hierarchy Pairs are (Canada, Quebec) and (Canada, Toronto). Note that by maintaining the hierarchy in a separate structure, i.e. the Node Hierarchy, Items can be reused in multiple Classifications. For instance, in another geography Statistical Classification provinces grouped into regions, Ontario can be made the child of the Central Region instead of Canada without changing the definition of the Classification Items involved, i.e. Canada, Ontario and Central Region in this

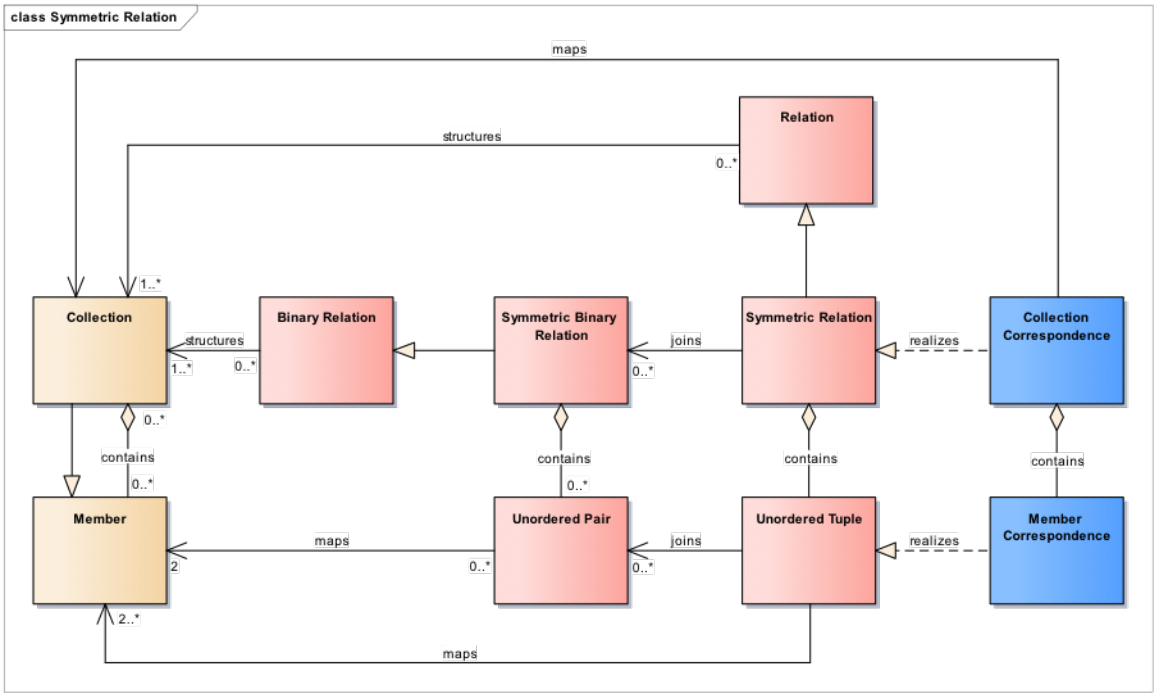
case. Interestingly enough, Binary Relations might not be enough for some purposes. First of all, some structures cannot be reduced to binary representations, e.g. hypergraphs. In addition, a Binary Relation could be too verbose in some cases since the same Member in a Collection could appear multiple times in different pairs, e.g. one-to-many relationships like parent-child and ancestor-descendent. An n-ary Relation provides a more compact representation for such cases. Like Binary Relations they come in two flavours: Symmetric Relation and Asymmetric Relation. Let us consider the asymmetric case (Figure 7) first.

Figure 7. Asymmetric relations



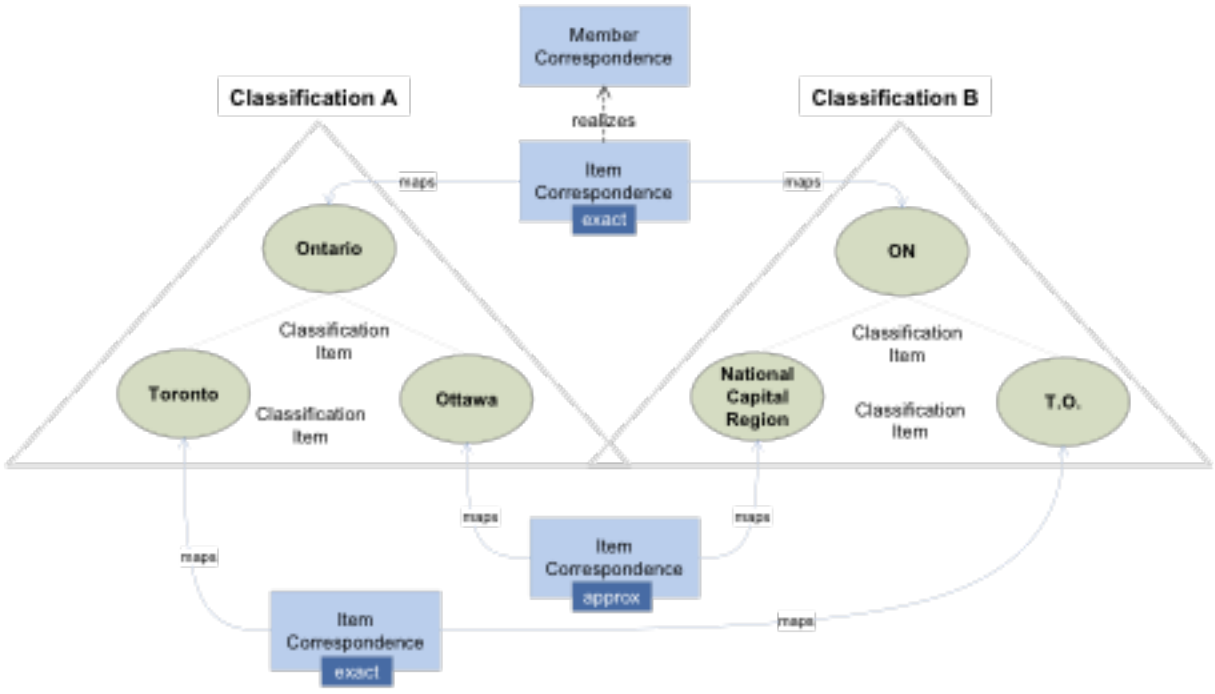
Asymmetric Relations provide an equivalent, yet more compact, n-ary representation for multiple Ordered Pairs that share the same source and/or target Members. In addition, they can be used to model Ordered Correspondences to map Collections and their Members based on some criterion (e.g. similarity, provenance, etc.). Ordered Collection and Member Correspondences realize Asymmetric Relation and Ordered Tuple, respectively. Consider now a geography classification tree like the Canadian example (Figure 6) above. All Node Hierarchy Pairs that have the same parent Member could be represented with a single Node Hierarchy Tuple that realizes the Ordered Tuple in the model. The realization will also rename source as parent and target as child. Although only two cities are shown in the example, Ontario has hundreds of them. Using a Node Hierarchy realizing and Asymmetric Relation, all pairs that have Ontario as parent, e.g. (Ontario, Toronto), (Ontario, Ottawa), (Ontario, Kingston), etc., could be joined into a single n-ary Node Hierarchy Tuple with Ontario as parent and Toronto, Ottawa, Kingston, etc. as children. With this representation we replaced multiple pairs with a single tuple and avoid the repetition of Ontario hundreds of times for each individual pair. Symmetric Relations are similarly structured as Asymmetric Relations (Figure 8). They provide an equivalent, yet more compact, n-ary representation for multiple Unordered Pairs that have some Members in common. In addition, they can be used to model (unordered) Correspondences between Collections and Members.

Figure 8. Symmetric relations



Back to our geography Statistical Classification example (Figure 6), we can have two variants with similar structure as shown in Figure 9.

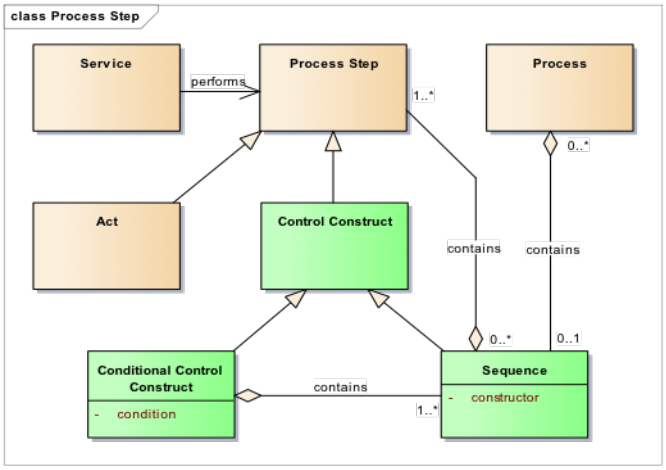
Figure 9. Example showing both Asymmetric and Symmetric relations



4.2 Using the Process pattern

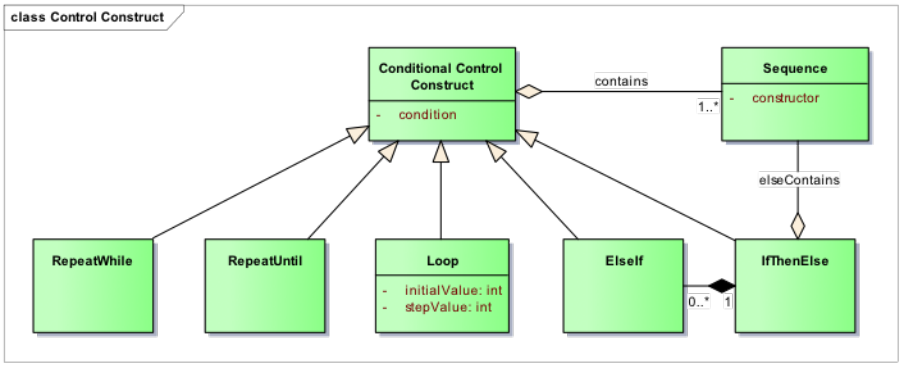
The process pattern consists of classes to describe business functions and workflows that can be mapped to process execution languages like BPEL or BPMN. A Process is a Sequence of Process Steps that perform one or more business functions. Each Process Step can be performed by a Service. There are two types of Process Steps: Acts and Control Constructs. Acts represent actions and are atomic Process Steps, i.e. they cannot be composed of other Process Steps. An Act is similar to an instruction in a programming language and a terminal in the production rules of a formal grammar. A Control Construct describes logical flows between Process Steps.

Figure 1. Process Step class



Process Steps can be nested and thus describe processes at multiple levels of detail. The nesting of Process Steps always terminate in an Act. All nesting of Process Steps occur via Sequences, a specialization of Control Constructs. Control Constructs include Sequence and Conditional Control Construct (Figure 2). The former models linear execution of Process Steps whereas the latter includes three types of iterative constructs: repeatWhile, repeatUntil and Loop. The Sequence at the end of the contains association represents the body of the Conditional Control Construct, which is executed depending on the result of the condition evaluation. The specialized sub-classes determine whether the Sequence is executed in each iteration before the condition is evaluated (RepeatUntil), or after (RepeatWhile, Loop). The Loop also provides a counter with initialValue and stepValue that can be used to specify in the condition how many times the Sequence in the body is executed.

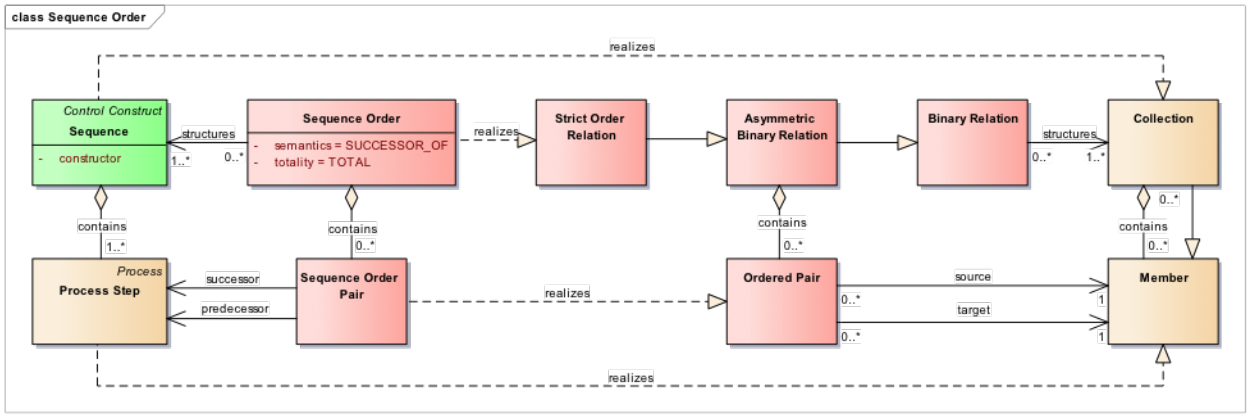
Figure 2. Control Construct class



In addition to the iterative constructs, Conditional Control Constructs includes IfThenElse, which provides a means to specify branching control flows. It contains the condition inherited from the parent class and two associations: contains (also inherited from the parent class), to the Sequence of Process Steps that is executed when the condition is true, and containsElse, to an optional Sequence to be executed if the condition is evaluated to false. Optionally, IfThenElse can also have an associated ElseIf construct to model switch statements. It is important to note that the

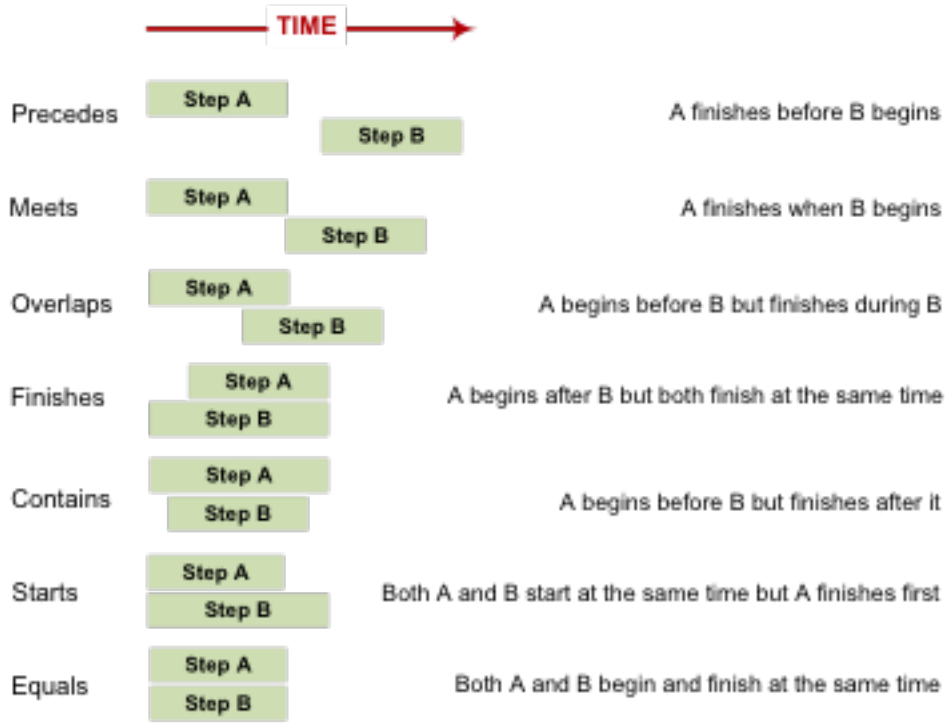
model also covers parallel processing: Conditional Control Constructs can contain multiple Sequences that could be executed in parallel (more on this later). A Sequence can be viewed as a Collection whose Members are Process Steps that can be ordered in three different ways: with a Sequence Order (traditional design-time total ordering), with one or more Temporal Interval Relations (design-time temporal constraint or "fuzzy" ordering), or with a Rule (constructor) to determine ordering at run-time. We discuss Sequence Order first. A Sequence Order realizes Collection pattern as follows (Figure 3).

Figure 3. Sequence Order



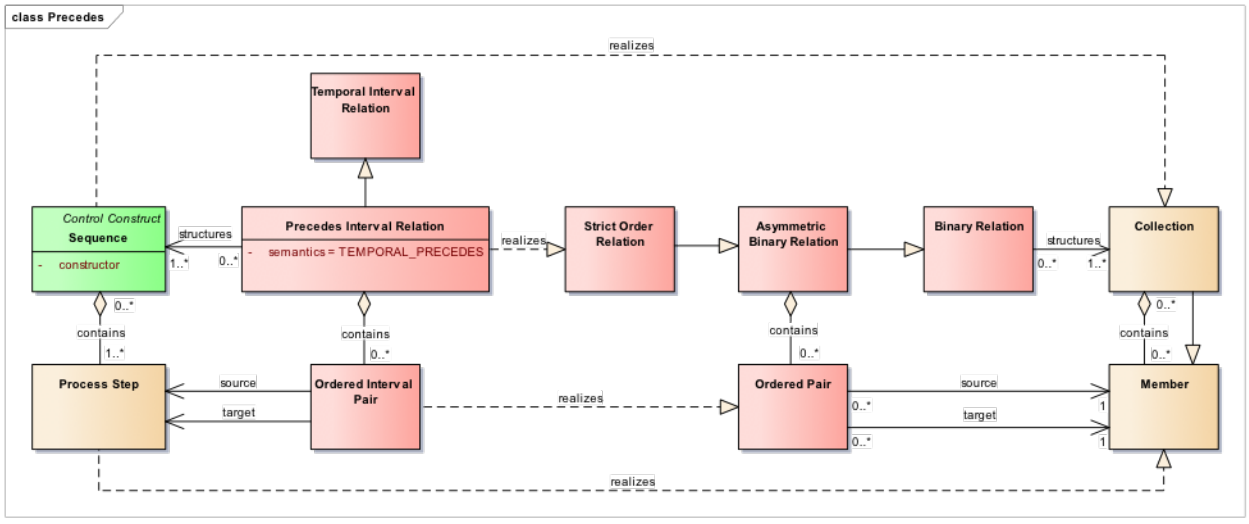
Sequence Order and Sequence Order Pair realize Strict Order Relation and Ordered Pair, respectively. Let us remember from the Binary Relations Specialization diagram [hyperlink to Collections Figure ???] that Strict Order Relation is an Asymmetric Binary Relation that is ANTI_SYMMETRIC, ANTI_REFLEXIVE and TRANSITIVE. In addition, the Sequence Order realization has totality=TOTAL and semantics=SUCCESSOR_OF, which means that it can specify a total order of the Process Steps in the Sequence where the order semantics is given by SUCCESOR_OF. We discuss next Temporal Interval Relations. They provide a mechanism for capturing Allen's interval relations, one of the best established formalisms for temporal reasoning. Temporal Interval Relations can be used to define temporal constraints between pairs of Process Steps, e.g. whether the execution of two Process Steps can overlap in time or not, or one has to finish before the other one starts, etc. This also supports parallel processing. There are thirteen Temporal Interval Relations: twelve asymmetric ones, i.e. precedes, meets, overlaps, finishes, contains, starts and their converses, plus equals, which is the only one that has no converse, or rather, it is the same as its converse. Together these relations are distinct (any pair of definite intervals are described by one and only one of the relations), exhaustive (any pair of definite intervals are described by one of the relations), and qualitative (no numeric time spans are considered). Following Allen's, Temporal Interval Relations are defined as follows.

Figure 4. Allen's Temporal Interval Relations



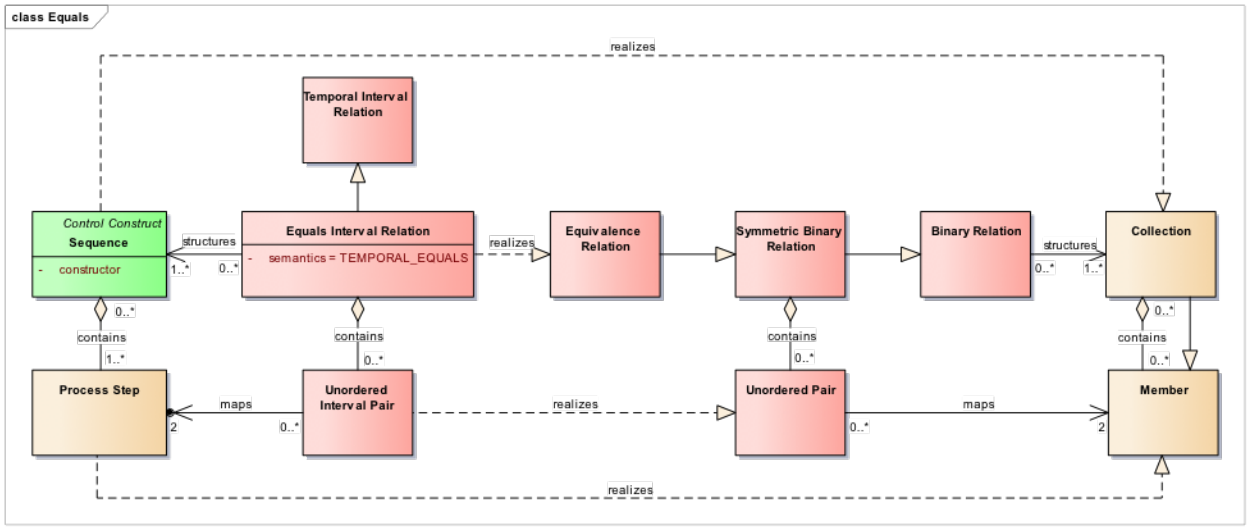
In DDI-Views, each one of the asymmetric Allen's interval relations is a Temporal Interval Relation that realizes different Binary Relations with specific temporal semantics. All asymmetric Temporal Interval Relation contains Ordered Interval Pairs whereas the only symmetric one, i.e. Equals, contains Unordered Interval Pairs (Figure 5). For instance, the Precedes Interval Relation realizes the pattern as follows.

Figure 5. Precedes Interval Relation



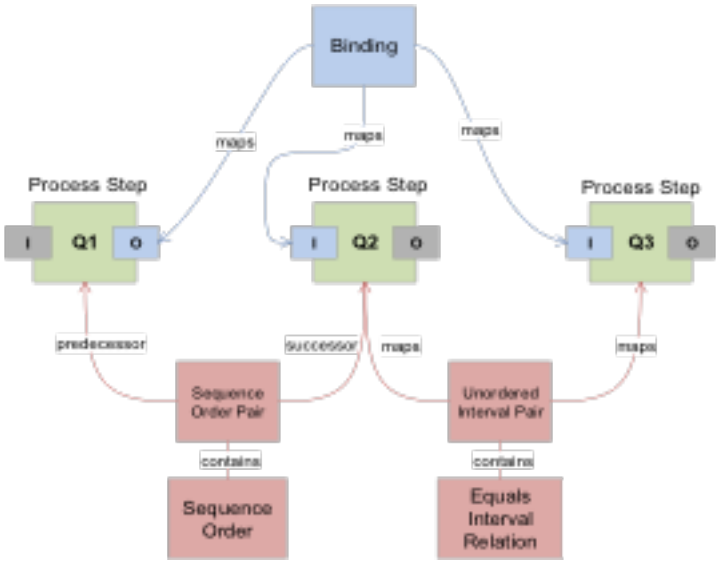
Precedes Interval Relation and Ordered Interval Pair realize Strict Order Relation and Ordered Pair, respectively. If we look back to the Binary Relations Specialization diagram in the previous section we notice that Strict Order Relation has the TEMPORAL_PRECEDES semantics, among others, which means that the Process Step at the end of the source association in the Ordered Interval Pair has to finish before the one at the end of target starts. The Equals Interval Relation (Figure 6) is a slightly different case because it is an equivalence relation rather than an asymmetric one and therefore it contains Unordered Interval Pairs.

Figure 6. Equals Interval Relation



Equals Interval Relation and Unordered Interval Pair realize Equivalence Relation and Unordered Pair, respectively. Equivalence Relation is simply a Symmetric Binary Relation that is REFLEXIVE and TRANSITIVE with some additional semantics, among which we find TEMPORAL_EQUALS, the one required by the Equals Interval Relation. This means that the execution of the two Process Steps at the end of the maps association in Unordered Interval Pair begin and end at the same time. Temporal Interval Relations and Sequence Orders can be combined in the same Sequence. For instance, consider Figure 7:

Figure 7. Combination of Interval Relations



Question Q2 requires the answer of question Q1 so it has to be executed after Q1. That is an example of the traditional sequence ordering given by the Sequence Order Relation. However, note that there is no dependency between Q2 and Q3 since both require only the answer of Q1. Therefore Q2 and Q3 could be executed at the same time, which can be expressed with the Equals Interval Relation.

4.3 The Variable Cascade

The DDI-Lifecycle standard is intended to address the metadata needs for the entire survey lifecycle. This particular document is dedicated to a description of variables as part of the DDI-Lifecycle. It contains a UML (Unified Modeling

Language) class diagram of a model for describing variables, and the model is part of the larger development effort called DDI Moving Forward to express DDI-Lifecycle using UML.

Typical models for describing variables take advantage of much reuse, and the model provided here is no exception. It is reuse that makes metadata management such an effective approach. However, effectiveness is due to other factors as well, and an important one is to keep the number of objects described to a minimum. For finding relevant data is much more complicated as the number of objects rises.

For variables, reusable descriptions are brittle in the sense that if one of the related records to a variable changes, then a new variable needs to be defined. This is especially true when considering the allowed values (the Value Domain) for a variable. Many small variations in value domains exist in production databases, yet these differences are often gratuitous (e.g., simple differences in the way some category is described that do not alter the meaning), differences in representation (e.g., simple changes from letter codes to numeric ones), or differences in the way missing (or sentinel) values are represented.

Gratuitous differences in expressions of meaning are reduced or eliminated by encouraging the usage of URIs (Uniform Resource Identifiers) to point to definition entries in taxonomies of terms and concepts. The principle of "write once – use many", very similar to the idea of reuse, is employed. Pointing to an entry rather than writing its value eliminates transcription errors and simple expression differences, promotes comparability, and ensures interoperability.

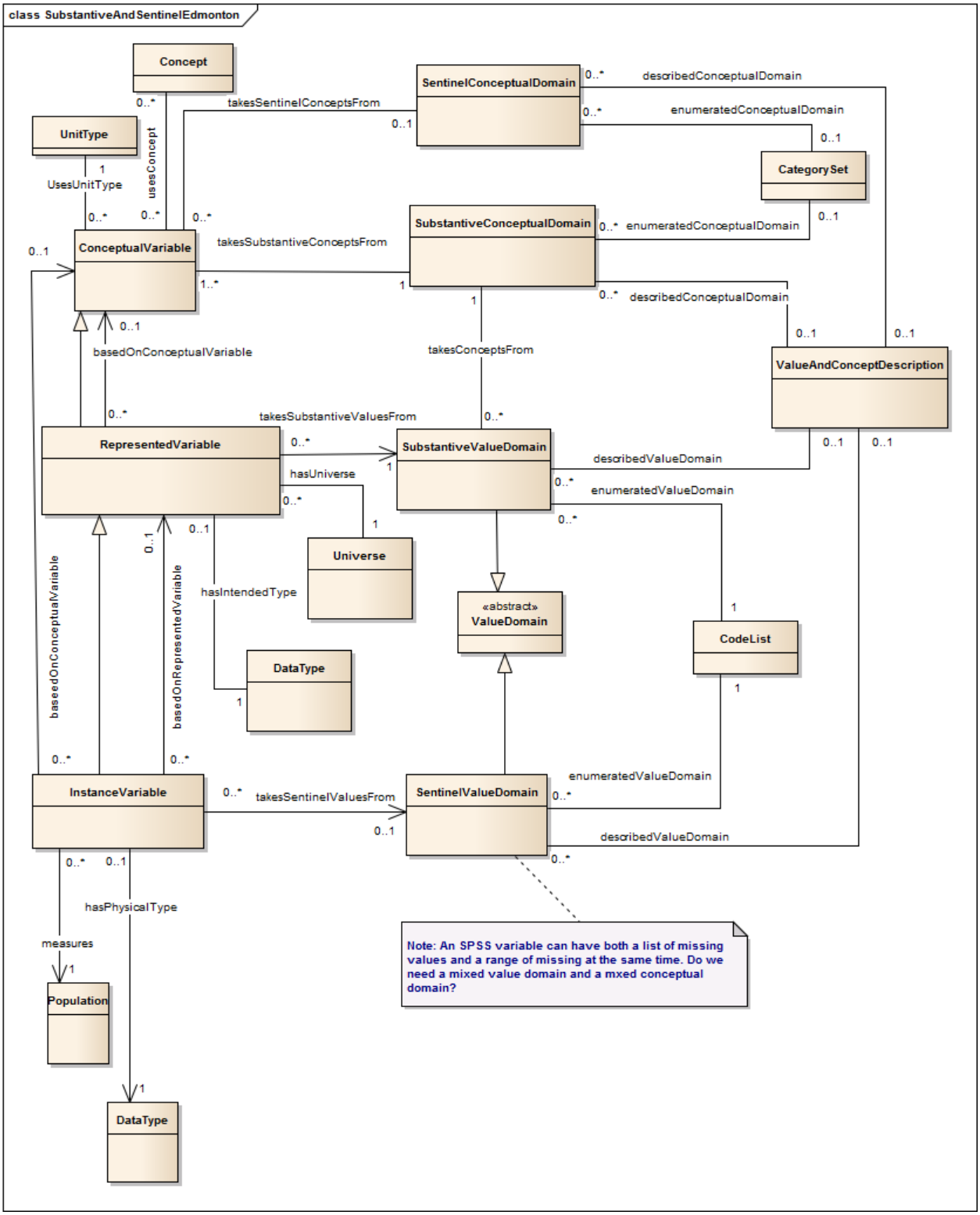
Differences in representations, including codes, are simplified by separating them from the underlying meaning. This is equivalent to the terminological issue of allowing for synonyms and homonyms of terms. Through reuse, all representations with the same meaning are linked to the same concept. This is achieved through the use of Value Domains and Conceptual Domains in the model presented here.

Sentinel values are important for any processing of statistical or experimental data, as there are multiple reasons some data are not obtainable. Typically, these values are added to the value domain for a variable. However, each time in the processing cascade the list of sentinel values changes, the value domain changes, which forces the variable to change as well. Given that each stage of the processing cascade make require a different set of sentinel values due to processing requirements, the number of variables mushrooms. And this variable proliferation is unmanageable and unsustainable.

The model developed here is based on two important standards for the statistical community, GSIM (Generic Statistical Information Model) and ISO/IEC 11179 (Metadata Registries). In fact, the model in the conceptual section of GSIM is based closely on the metamodel defined in ISO/IEC 11179. Both models help to perpetuate the problems described here, if each standard is followed in a conforming way. They reduce redundancy by separating value domains and conceptual domains. However, they do not directly support the use of URIs and they do not separate value domains from sentinel value lists. Also, they do not fully exploit the traceability that inheres in certain relationships between the value and conceptual domains to create a continuum of connected variables that further reduces redundancy.

The purpose of this document is to present a model that significantly reduces the overhead described above. In particular, we separate the sentinel values from the substantive ones. This separation allows us to greatly reduce the number of value domains, and thus variables, that need to be maintained. With this separation, now there are three classes of connected variables in which the represented variable specializes a conceptual variable by adding a value domain, and the instance variable specializes the represented variable by adding a sentinel value domain.

Figure 1. Variable Cascade


class SubstantiveAndSentinelEdmonton
Concept
SentinelConceptualDomain
describedConceptualDomain
takesSentinelConceptsFrom
enumeratedConceptualDomain
UnitType
usesConcept
UsesUnitType
CategorySet
SubstantiveConceptualDomain
enumeratedConceptualDomain
ConceptualVariable
takesSubstantiveConceptsFrom
describedConceptualDomain
basedOnConceptualVariable
takesConceptsFrom
ValueAndConceptDescription
RepresentedVariable
takesSubstantiveValuesFrom
SubstantiveValueDomain
describedValueDomain
hasUniverse
enumeratedValueDomain
Universe
hasIntendedType
«abstract»
ValueDomain
CodeList
DataType
basedOnConceptualVariable
basedOnRepresentedVariable
InstanceVariable
takesSentinelValuesFrom
SentinelValueDomain
enumeratedValueDomain
describedValueDomain
hasPhysicalType
measures
Population
DataType
Note: An SPSS variable can have both a list of missing values and a range of missing at the same time. Do we need a mixed value domain and a mxed conceptual domain?

4.3.1 Example

Detergents

Imagine we are assessing environmental influences at the household level. One question we might ask is "What detergents are used in the home?" In connection with this question we prepared show cards. Each show card lists a series of detergents. There are multiple show cards because products vary by region. Each show card corresponds to a code list. There are overlaps among the code lists but there are differences too.

In our model there is a conceptual variable. It has an enumerated conceptual domain that takes its categories from a category set. Here the category set is an unduplicated list of detergents taken from all the show cards put together. The conceptual domain might be exposure to chemicals in household detergents. The unit type might be households.

In our survey we ask a question corresponding to multiple represented variables, one corresponding to each show card. Each represented variable is measured by an enumerated value domain that takes its values from the show card code list.

All the represented variables here are derived from the same conceptual variable. This is the main point of the example: a conceptual variable under the right conditions can connect multiple represented variables.

Sentinel Values

The represented variable code list in our model excludes sentinel values. Sentinel values were introduced into ISO/IEC 11404 in 2007. ISO/IEC 11404 describes a set of language independent datatypes and defines a sentinel value as follows: a sentinel value is a "signaling" value such as nil, NaN (not a number), +inf and –inf (infinities), and so on. Depending on the study, sentinel values may include missing values. ISO 21090 is a health datatypes standard based on ISO/IEC 11404. ISO 21090 identifies 15 "nullflavors" that correspond to the concept of sentinel values introduced in ISO 11404 .

In our model the instance variable adds a sentinel value domain to the represented variable from which it is derived. In the process it grows the code list it derived from the represented variable to include a set of sentinel values. These sentinel values reference a category set of sentinels called the sentinel conceptual domain. The sentinel values included in any instance variable need not cover all the members of the sentinel conceptual domain. Instead they may refer just to a subset.

During data acquisition, we ask a question that allows don't know and refused, which an interviewer may ask or not, depending on the skip logic. We create an instance variable corresponding to this question based on a represented variable. The instance variable includes sentinel values. Note that the value domain of the represented variable need not be an enumerated value domain. Instead it can be a described value domain. We choose to include three sentinel values corresponding to three ISO 21090 nullflavors:

Nullflavor	Description
UNK	A proper value is applicable, but not known. Corresponds to Refused.
ASKU	Information was sought but not found Corresponds to Don't Know.
NA	No proper value is applicable in this context (e.g., last menstrual period for a male)

Subsequently, we prepare the collected data for processing by SAS. SAS has its own set of sentinel values. For each sentinel category the data acquisition instance variable accounts for, SAS has its own set of sentinel values. As a consequence, the answers that correspond to sentinel values are different, and in the process of constructing a SAS file, a second instance variable is created for the purposes of data processing.

However, both the data acquisition instance variable and the data processing instance variable are derived from the same represented variable. That is the point of this example.

CHAPTER 5

Packages

5.1 NewObjectsForSimpleInstruments

New objects to enter the simple instrument view Contents

5.2 Identification

Objects to support Identification and Versioning Contents

5.2.1 AnnotatedIdentifiable

Used to identify objects for purposes of internal and/or external referencing. Elements of this type are versioned. Provides administrative metadata about the object in addition to what is provided by Identifiable, including more details on the versioning of the object. Most objects except for the ComplexDataTypes will inherit AnnotatedIdentfiable.

Extends

Identifiable

Properties

Name	Type	Cardinality
versionResponsibility	xs:string	0..1
versionRationale	xs:string	0..1
versionDate	xs:dateTime	0..1
isUniversallyUnique	xs:boolean	1..1
isPersistent	xs:boolean	1..1
localId	LocalId	0..n
basedOnObject	BasedOnObject	0..1

versionResponsibility

Contributor who has the ownership and responsiblity for the current version.

versionRationale

The reason for making this version of the object.

versionDate

The date and time the object was changed.

isUniversallyUnique

Usually the combination of agency and id (ignoring different versions) is unique. If isUniversallyUnique is set to true, it indicates that the id itsef is universally unique (unique across systems and/or agencies) and therefore the agency part is not required to ensure uniqueness. Default value is false.

isPersistent

Usually the content of the current version is allowed to change, for example as the contibutor is working on the object contents. However, when isPersistent is true, it indicates the there will be no more changes to the current version. Default value is false.

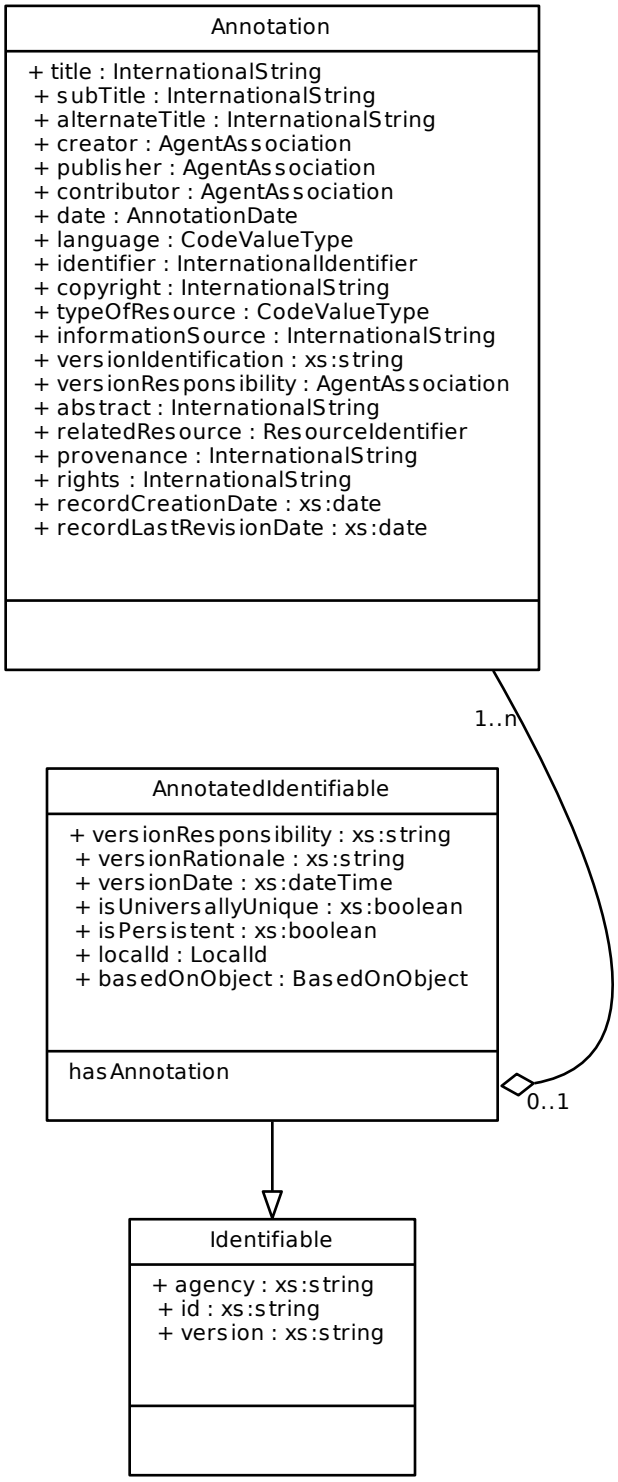
localId

This is an identifier in a given local context that uniquely references an object, as opposed to the full ddi identifier which has an agency plus the id. For example, localId could be a variable name in a dataset.

basedOnObject

The object/version that this object version is based on.

Graph


Annotation
+ title : InternationalString
+ subTitle : InternationalString
+ alternateTitle : InternationalString
+ creator : AgentAssociation
+ publisher : AgentAssociation
+ contributor : AgentAssociation
+ date : AnnotationDate
+ language : CodeValueType
+ identifier : InternationalIdentifier
+ copyright : InternationalString
+ typeOfResource : CodeValueType
+ informationSource : InternationalString
+ versionIdentification : xs:string
+ versionResponsibility : AgentAssociation
+ abstract : InternationalString
+ relatedResource : ResourceIdentifier
+ provenance : InternationalString
+ rights : InternationalString
+ recordCreationDate : xs:date
+ recordLastRevisionDate : xs:date
1..n
AnnotatedIdentifiable
+ versionResponsibility : xs:string
+ versionRationale : xs:string
+ versionDate : xs:dateTime
+ isUniversallyUnique : xs:boolean
+ isPersistent : xs:boolean
+ localId : LocalId
+ basedOnObject : BasedOnObject
hasAnnotation
0..1
Identifiable
+ agency : xs:string
+ id : xs:string
+ version : xs:string

5.2.2 Identifiable

Used to identify objects for purposes of internal and/or external referencing. Elements of this type are versioned. Most objects except for the ComplexDataTypes will inherit from Identifiable or the more specialised AnnotatedIdentfiable.

Properties

Name	Type	Cardinality
agency	xs:string	1..1
id	xs:string	1..1
version	xs:string	1..1

agency

This is the registered agency code with optional sub-agencies separated by dots. For example, diw.soep, ucl.qss, abs.essg.

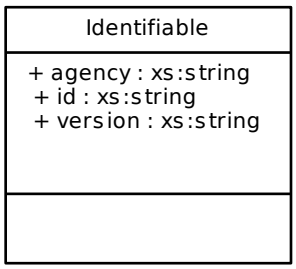
id

The ID of the object. This must conform to the allowed structure of the DDI Identifier and must be unique within the declared scope of uniqueness (Agency or Maintainable).

version

The version number of the object. The version number is incremented whenever the non-administrative metadata contained by the object changes.

Graph



5.3 Discovery

Contains objects used to provide annotation and coverage information for DDI publications (views, object sets, data files, etc.) Contents

5.3.1 Access

Describes access to the annotated object. This item includes a confidentiality statement, descriptions of the access permissions required, restrictions to access, citation requirements, depositor requirements, conditions for access, a disclaimer, any time limits for access restrictions, and contact information regarding access.

Extends

AnnotatedIdentifiable

Properties

Name	Type	Cardinality
description	StructuredString	0..1
confidentialityStatement	StructuredString	0..1
accessPermission	Form	0..n
restrictions	StructuredString	0..1
citationRequirement	StructuredString	0..1
depositRequirement	StructuredString	0..1
accessConditions	StructuredString	0..1
disclaimer	StructuredString	0..1
contactAgent	AgentAssociation	0..n
applyAccessTo	InternationalIdentifier	0..n
validDates	Date	0..1

description

A description of the content and purpose of the access description. May be expressed in multiple languages and supports the use of structured content.

confidentialityStatement

A statement regarding the confidentiality of the related data or metadata.

accessPermission

A link to a form used to provide access to the data or metadata including a statement of the purpose of the form.

restrictions

A statement regarding restrictions to access. May be expressed in multiple languages and supports the use of structured content.

citationRequirement

A statement regarding the citation requirement. May be expressed in multiple languages and supports the use of structured content.

depositRequirement

A statement regarding depositor requirements. May be expressed in multiple languages and supports the use of structured content.

accessConditions

A statement regarding conditions for access. May be expressed in multiple languages and supports the use of structured content.

disclaimer

A disclaimer regarding the liability of the data producers or providers. May be expressed in multiple languages and supports the use of structured content.

contactAgent

The agent to contact regarding access including the role of the agent.

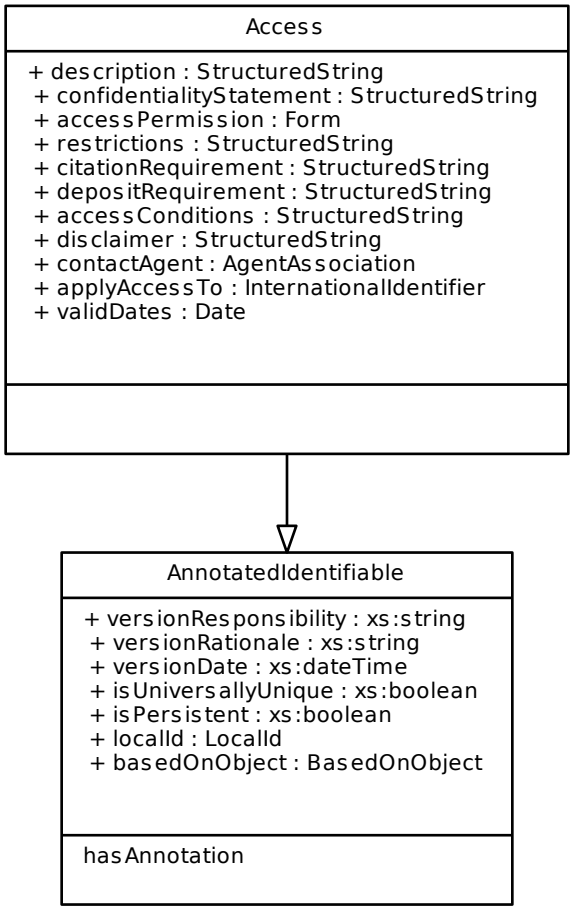
applyAccessTo

Identification for an object covered by the access description. This may be any annotated object (collection, publication, identifiable object).

validDates

The date range or start date of the access description.

Graph



5.3.2 Annotation

Provides annotation information on the object to support citation and crediting of the creator(s) of the object.

Extends

Identifiable

Properties

Name	Type	Cardinality
title	InternationalString	0..1
subTitle	InternationalString	0..n
alternateTitle	InternationalString	0..n
creator	AgentAssociation	0..n
publisher	AgentAssociation	0..n
contributor	AgentAssociation	0..n
date	AnnotationDate	0..n
language	CodeValueType	0..n
identifier	InternationalIdentifier	0..n
copyright	InternationalString	0..n
typeOfResource	CodeValueType	0..n
informationSource	InternationalString	0..n
versionIdentification	xs:string	0..1
versionResponsibility	AgentAssociation	0..n
abstract	InternationalString	0..1
relatedResource	ResourceIdentifier	0..n
provenance	InternationalString	0..n
rights	InternationalString	0..n
recordCreationDate	xs:date	0..1
recordLastRevisionDate	xs:date	0..1

title

Full authoritative title. List any additional titles for this item as AlternativeTitle.

subTitle

Secondary or explanatory title.

alternateTitle

An alternative title by which a data collection is commonly referred, or an abbreviation for the title.

creator

Person, corporate body, or agency responsible for the substantive and intellectual content of the described object.

publisher

Person or organization responsible for making the resource available in its present form.

contributor

The name of a contributing author or creator, who worked in support of the primary creator given above.

date

A date associated with the annotated object (not the coverage period). Use typeOfDate to specify the type of date such as Version, Publication, Submitted, Copyrighted, Accepted, etc.

language

Language of the intellectual content of the described object. Strongly recommend the use of language codes supported by xs:language which include the 2 and 3 character and extended structures defined by RFC4646 or its successors.

identifier

An identifier or locator. Contains identifier and Managing agency (ISBN, ISSN, DOI, local archive). Indicates if it is a URI.

copyright

The copyright statement.

typeOfResource

Provide the type of the resource. This supports the use of a controlled vocabulary. It should be appropriate to the level of the annotation.

informationSource

The name or identifier of source information for the annotated object.

versionIdentification

Means of identifying the current version of the annotated object.

versionResponsibility

The agent responsible for the version. May have an associated role.

abstract

An a abstract (description) of the annotated object.

relatedResource

Provide the identifier, managing agency, and type of resource related to this object.

provenance

A statement of any changes in ownership and custody of the resource since its creation that are significant for its authenticity, integrity, and interpretation.

rights

Information about rights held in and over the resource. Typically, rights information includes a statement about various property rights associated with the resource, including intellectual property rights.

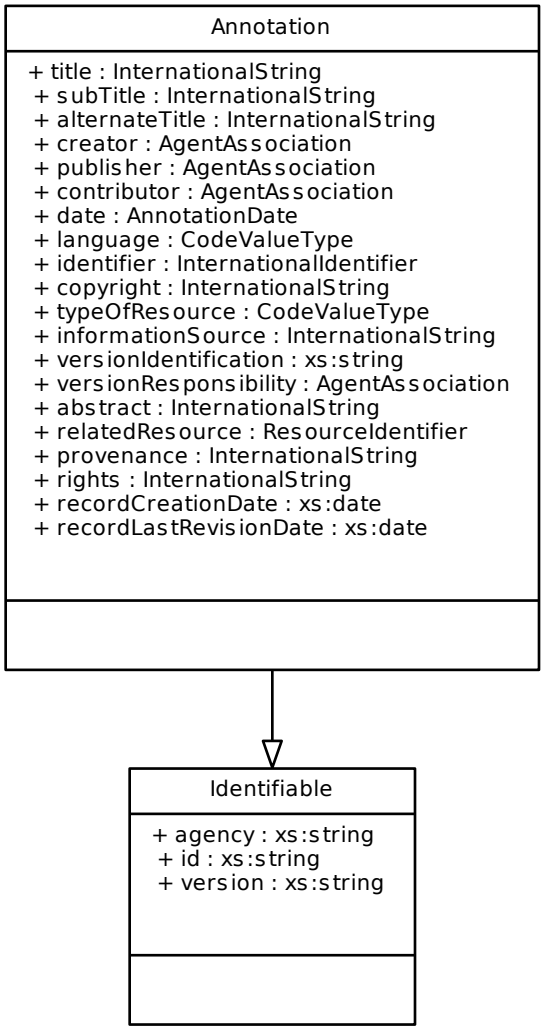
recordCreationDate

Date the record was created

recordLastRevisionDate

Date the record was last revised

Graph



5.3.3 BoundingBox

A type of Spatial coverage describing a rectangular area within which the actual range of location fits. A BoundingBox can be described by 4 numbers, or two x,y coordinates - the maxima of the north, south, east, and west coordinates found in the area.

Extends

AnnotatedIdentifiable

Properties

Name	Type	Cardinality
eastLongitude	xs:decimal	1..1
westLongitude	xs:decimal	1..1
northLatitude	xs:decimal	1..1
southLatitude	xs:decimal	1..1

eastLongitude

The easternmost coordinate expressed as a decimal between the values of -180 and 180 degrees

westLongitude

The westernmost coordinate expressed as a decimal between the values of -180 and 180 degrees

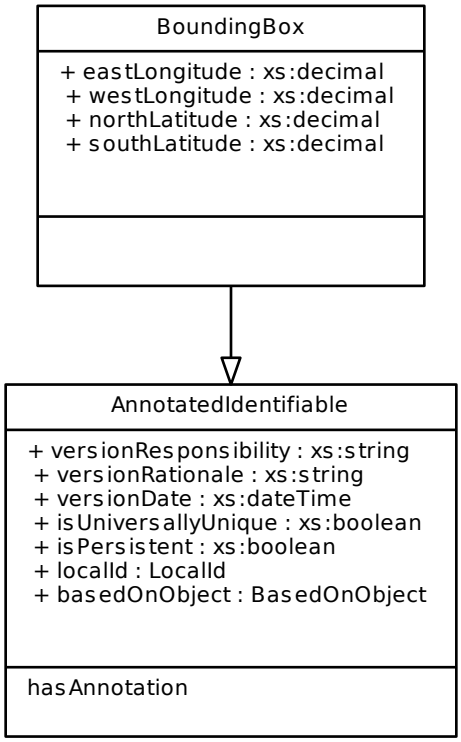
northLatitude

The northernmost coordinate expressed as a decimal between the values of -90 and 90 degrees.

southLatitude

The southermost latitude expressed as a decimal between the values of -90 and 90 degrees

Graph



5.3.4 Coverage

Coverage information for an annotated object. Includes coverage information for temporal, topical, and spatial coverage.

Extends

AnnotatedIdentifiable

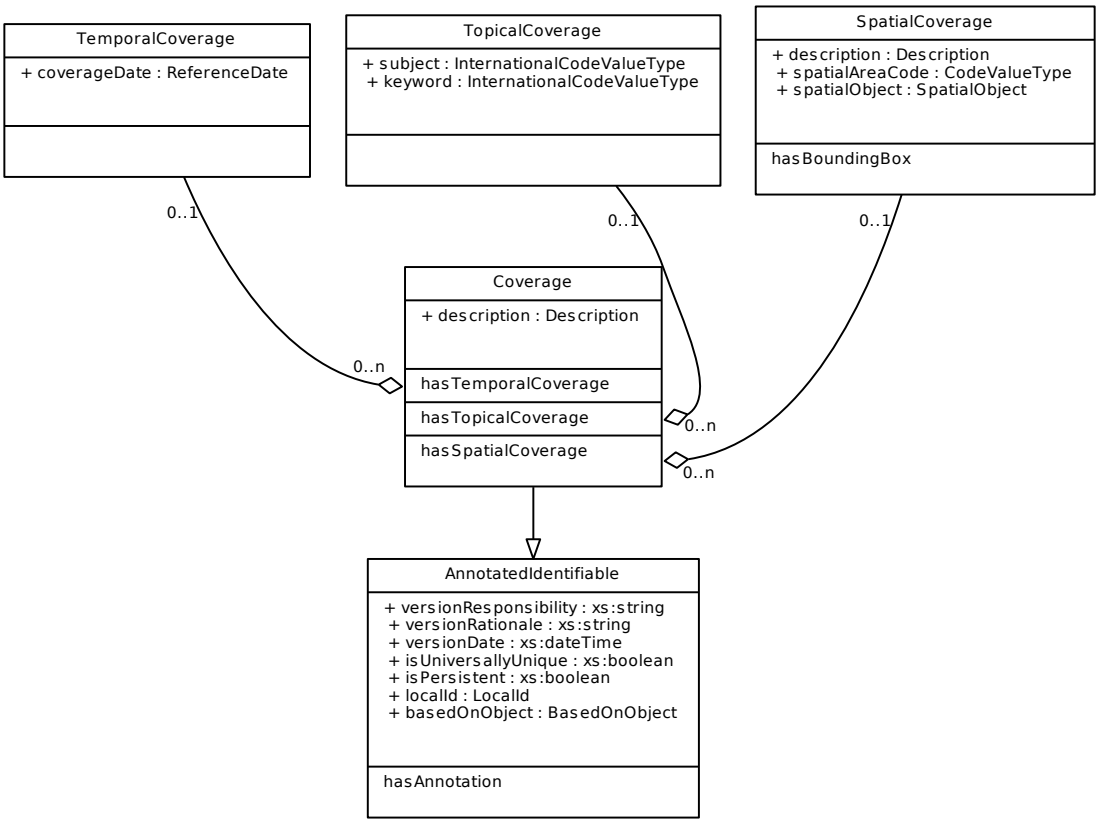
Properties

Name	Type	Cardinality
description	Description	0..n

description

A generic description including temporal, topical, and spatial coverage that is the equivalent of dc:coverage (the refinement base of dcterms:spatial and dcterms:temporal. Use specific coverage content for detailed information.

Graph



5.3.5 SpatialCoverage

A description of spatial coverage (geographic coverage) of the annotated object. Spatial coverage is described using a number of objects that support searching by a wide range of systems (geospatial coordinates, geographic classification systems, and general systems using dcterms:spatial.

Extends

AnnotatedIdentifiable

Properties

Name	Type	Cardinality
description	Description	0..1
spatialAreaCode	CodeValueType	0..n
spatialObject	SpatialObject	0..1

description

A textual description of the spatial coverage to support general searches.

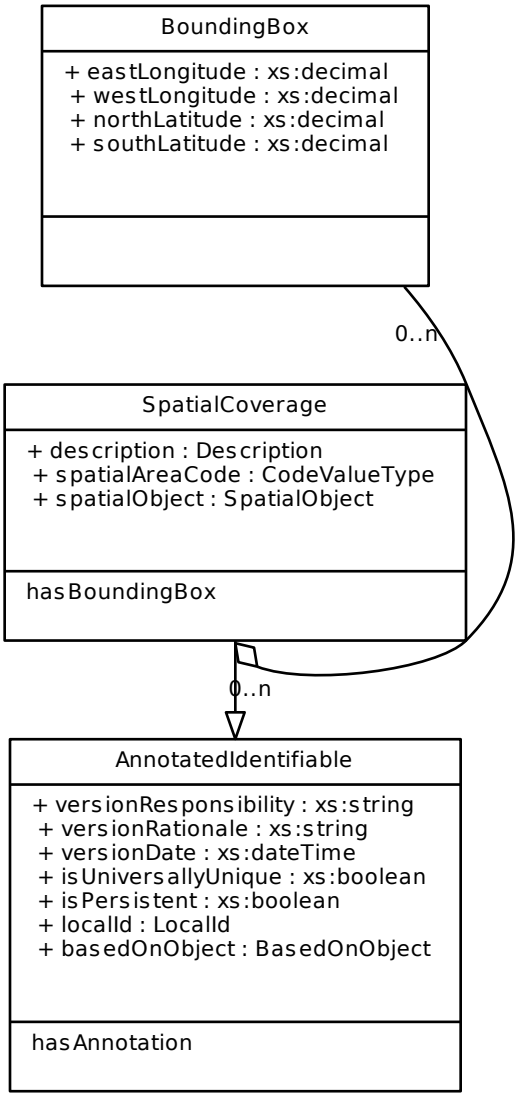
spatialAreaCode

Supports the use of a standardized code such as ISO 3166-1, the Getty Thesaurus of Geographic Names, FIPS-5, etc.

spatialObject

Indicates the most discrete spatial object type identified for a single case. Note that data can be collected at a geographic point (address) and reported as such in a protected file, and then aggregated to a polygon for a public file.

Graph



5.3.6 TemporalCoverage

Describes the date or time period covered by the annotated object. Allows for the use of a specifying the type of coverage date as well as associated subjects or keywords.

Extends

AnnotatedIdentifiable

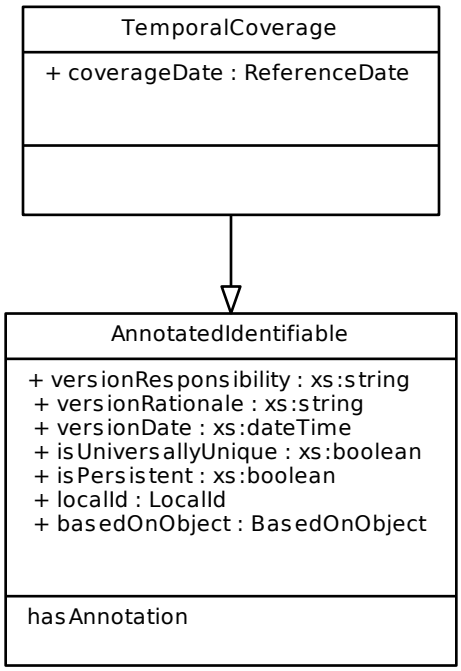
Properties

Name	Type	Cardinality
coverageDate	ReferenceDate	0..n

coverageDate

A date referencing a specific aspect of temporal coverage. The date may be typed to reflect coverage date, collection date, referent date, etc. Subject and Keywords may be associated with the date to specify a specific set of topical information (i.e. Residence associated with a date 5 years prior to the collection date).

Graph



5.3.7 TopicalCoverage

Describes the topical coverage of the module using Subject and Keyword. Note that upper level modules should include all the members of lower level modules. Subjects are members of structured classification systems such as formal subject headings in libraries. Keywords may be structured (e.g. TheSoz thesauri) or unstructured and reflect the terminology found in the document and other related (broader or similar) terms.

Extends

AnnotatedIdentifiable

Properties

Name	Type	Cardinality
subject	InternationalCodeValueType	0..n
keyword	InternationalCodeValueType	0..n

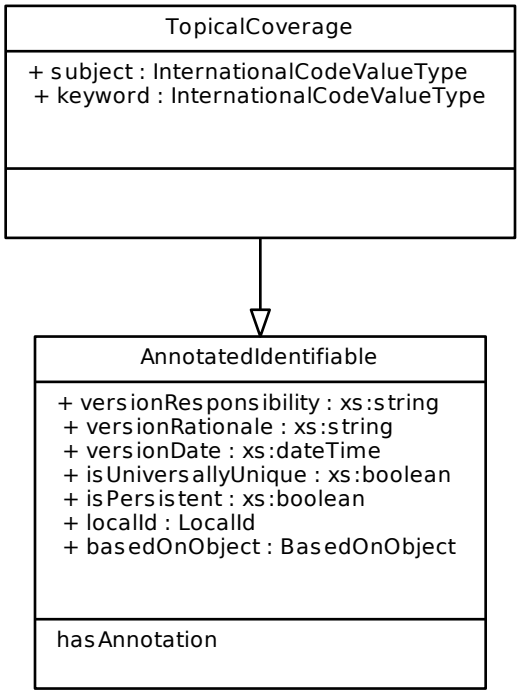
subject

A subject that describes the topical coverage of the content of the annotated object. Subjects are members of structured classification systems such as formal subject headings in libraries. Uses and InternationalCodeValue and may indicate the language of the code used.

keyword

A keyword that describes the topical coverage of the content of the annotated object. Keywords may be structured (e.g. TheSoz thesauri) or unstructured and reflect the terminology found in the document and other related (broader or similar) terms. Uses and InternationalCodeValue and may indicate the language of the code used.

Graph



5.4 Primitives

Base data types, a restricted set of object only available as properties of other objects Contents

5.5 Processing

Contents

5.5.1 Command

Provides the following information on the command The content of the command, the programming language used, the pieces of information (InParameters) used by the command, the pieces of information created by the command (OutParameters) and the source of the information used by the InParameters (Binding).

Extends

AnnotatedIdentifiable

Properties

Name	Type	Cardinality
programLanguage	CodeValueType	0..1
commandContent	xs:string	0..1

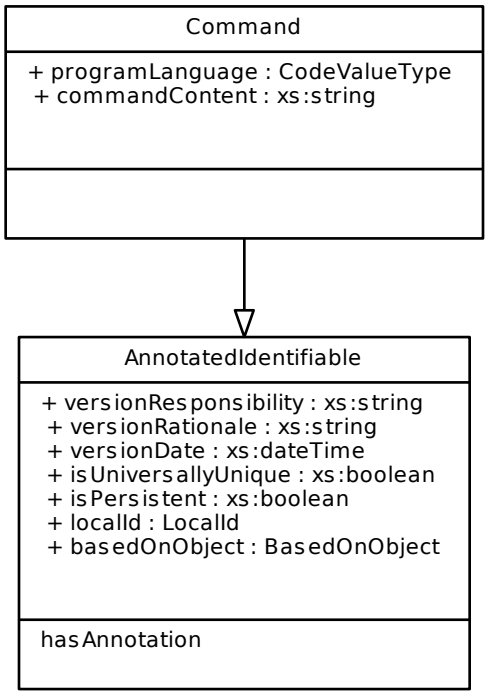
programLanguage

Designates the programming language used for the command. Supports the use of a controlled vocabulary.

commandContent

Content of the command itself expressed in the language designated in Programming Language.

Graph



5.5.2 CommandFile

Identifies and provides a link to an external copy of the command, for example, a SAS Command Code script. Designates the programming language of the command file, designates input and output parameters, binding information

between input and output parameters, a description of the location of the file , and a URN or URL for the command file.

Extends

AnnotatedIdentifiable

Properties

Name	Type	Cardinality
programLanguage	CodeValueType	0..1
location	InternationalString	0..1
uri	xs:anyURI	0..1

programLanguage

Designates the programming language used for the command. Supports the use of a controlled vocabulary.

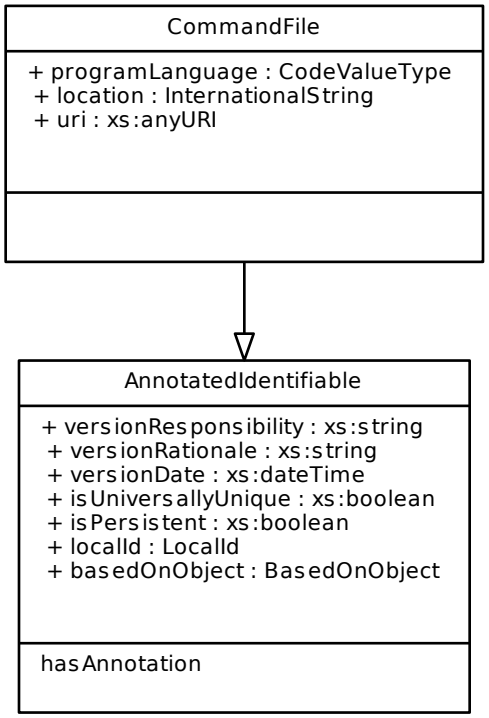
location

A description of the location of the file. This may not be machine actionable. It supports a description expressed in multiple languages.

uri

The URL or URN of the command file.

Graph



5.5.3 GenerationInstruction

Processing instructions for recodes, derivations from multiple question or variable sources, and derivations based on external sources. Instructions should be listed separately so they can be referenced individually.

Extends

Act

Properties

Name	Type	Cardinality
externalInformation	AccessRights	0..n
description	StructuredString	0..1
commandCode	CommandCode	0..n
isDerived	xs:boolean	0..1

externalInformation

Reference to an external source of information used in the coding process, for example a value from a chart, etc.

description

A description of the generation instruction. May be expressed in multiple languages and supports the use of structured content.

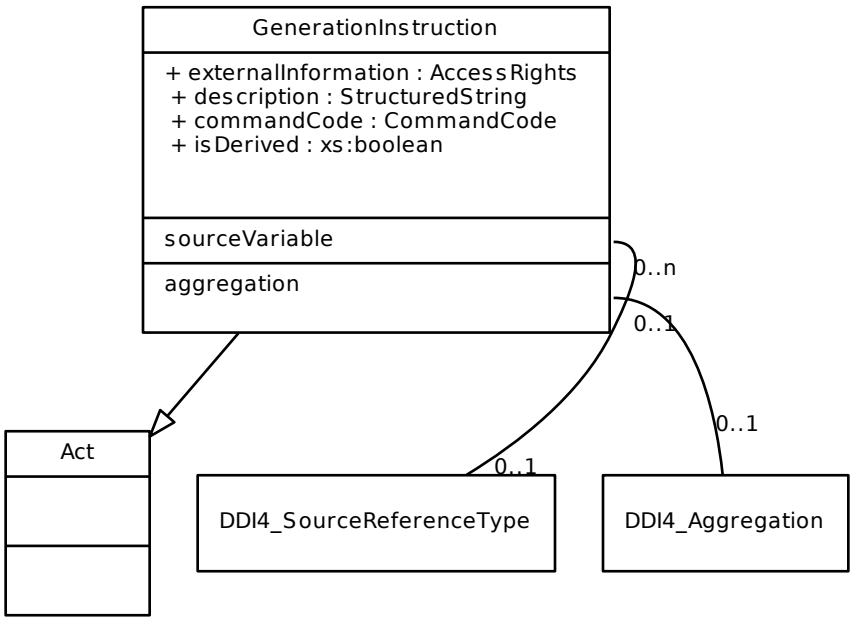
commandCode

Structured information used by a system to process the instruction.

isDerived

Default setting is “true”, the instruction describes a derivation. If the instruction is a simple recode, set to “false”.

Graph



5.5.4 Parameter

A parameter is a structure that specifically identifies a source of input or output information so that it can be use pragmatically.

Extends

AnnotatedIdentifiable

Properties

Name	Type	Cardinality
alias	xs:NMTOKEN	0..1
defaultValue	Value	0..1
isArray	xs:boolean	0..1

alias

If the content of the parameter is being used in a generic set of code as an alias for the value of an object in a formula (for example source code for a statistical program) enter that name here. This provides a link from the identified parameter to the alias in the code.

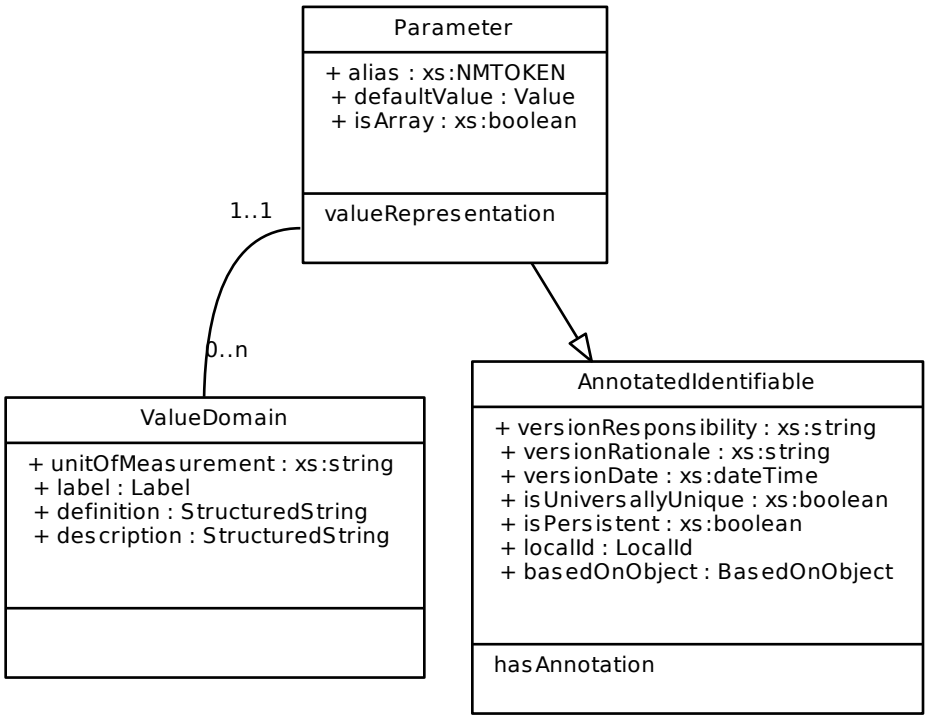
defaultValue

Provides a default value for the parameter if there is no value provided by the object it is bound to or the process that was intended to produce a value.

isArray

If set to "true" indicates that the content of the parameter is a delimited array rather than a single object and should be processed as such.

Graph



5.5.5 ProcessingInstruction

Extends

ControlConstruct

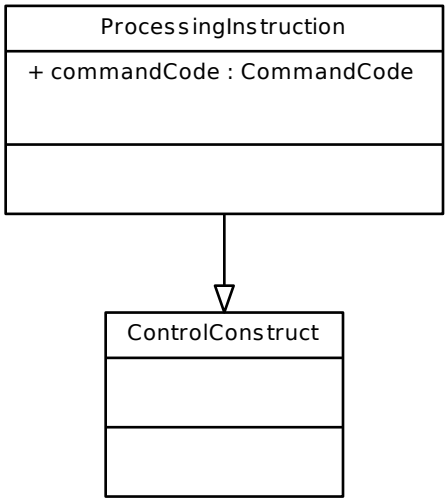
Properties

Name	Type	Cardinality
commandCode	CommandCode	0..n

commandCode

Structured information used by a system to process the instruction.

Graph



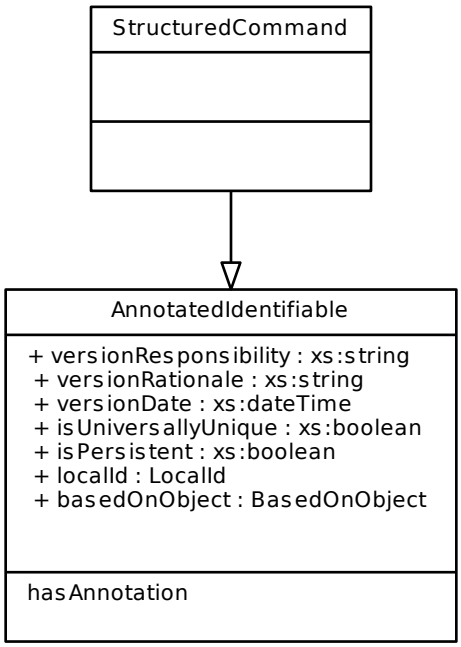
5.5.6 StructuredCommand

This type structures an empty stub which is used as the basis for extensions added using external namespaces such as MathML. The DDI 3.0 extension methodology is used here - a new module is declared, and the StructuredCommand element is used as the head of a substitution group to insert whatever XML is needed to express the command.

Extends

AnnotatedIdentifiable

Graph



5.6 Utility

Contents

5.6.1 Note

A note related to one or more identifiable objects. Note is designed to be an inherent part of the DDI. (Unlike XML comments or other types of system-level annotations, which may be removed during processing.) DDI recommends placing the note within the maintainable object containing the objects this note relates to in order to assist tracking of note items within a study. Each note may indicate who is responsible for the note, its type using a controlled vocabulary, the subject of the note, a head and note content, a set of key/value pairs and language specification for the overall note. In addition each note must be related to one or more identifiable objects.

Extends

AnnotatedIdentifiable

Properties

Name	Type	Cardinality
typeOfNote	CodeValueType	0..1
noteSubject	CodeValueType	0..1
relationship	Relationship	0..n
responsibility	xs:string	0..1
header	InternationalString	0..1
noteContent	StructuredString	0..1
proprietaryInfo	StandardKeyValuePair	0..1
xml:lang	xs:language	0..1

typeOfNote

Specifies the type of note. Supports the use of a controlled vocabulary.

noteSubject

The subject of the note.

relationship

Reference to one or more identifiable objects which the note is related to.

responsibility

The person or agency responsible for adding the note.

header

A brief label or heading for the note contents.

noteContent

The content of the note. Note should contain content except when it is a production flag that is fully explained by its "type". If the note provides system specific information in a structured way using XHTML formating, DDI strongly recommends the use of local extensions or the Key/Value pair structure in ProprietaryInfo whenever possible.

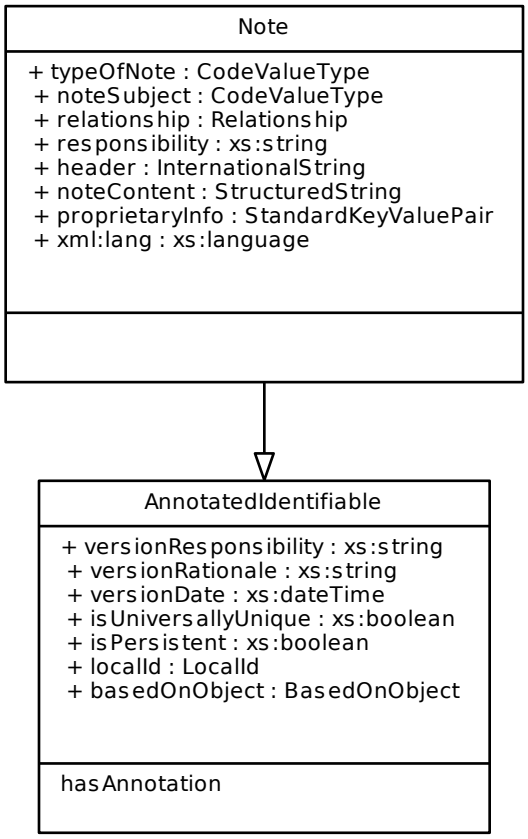
proprietaryInfo

A set of actions related to the object as described by a set of name-value pairs. This would commonly be used in a case where additional information needs to be recorded regarding the content of a new element or attribute that has not yet been added to the schema, for example when a bug for a missing object has been filed and the user wishes to record the content prior to correction in the schema. Ideally this should be handled by local extensions of the schema as described in Part 2 of the formal documentation. However, the structure in Note allows for an unanticipated need for an extension at run time by providing a means of capturing system specific information in a structured way.

xml:lang

Indicates the language of content. Note that xmlang allows for a simple 2 or 3 character language code or a language code extended by a country code , for example en-au for English as used in Australia.

Graph



5.7 SimpleDiscovery

Contents

5.8 Representations

Logical Data Description covers the logical content of a dataset - the “variable cascade”. Contents

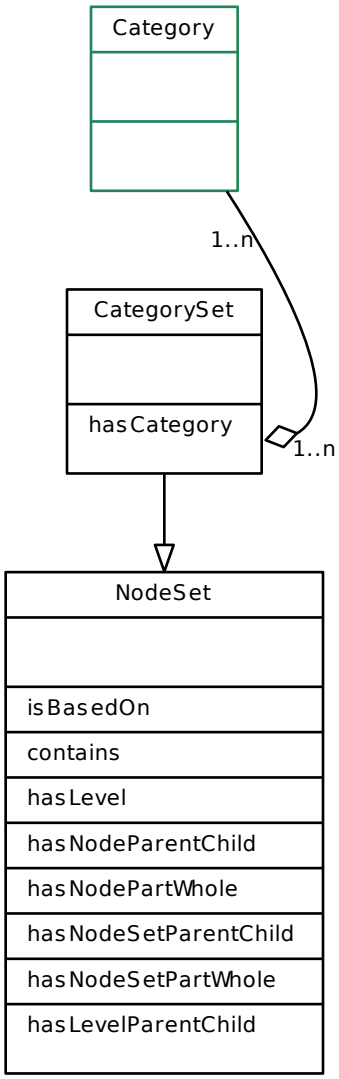
5.8.1 CategorySet

A Category Set is a type of Node Set which groups Categories.

Extends

NodeSet

Graph



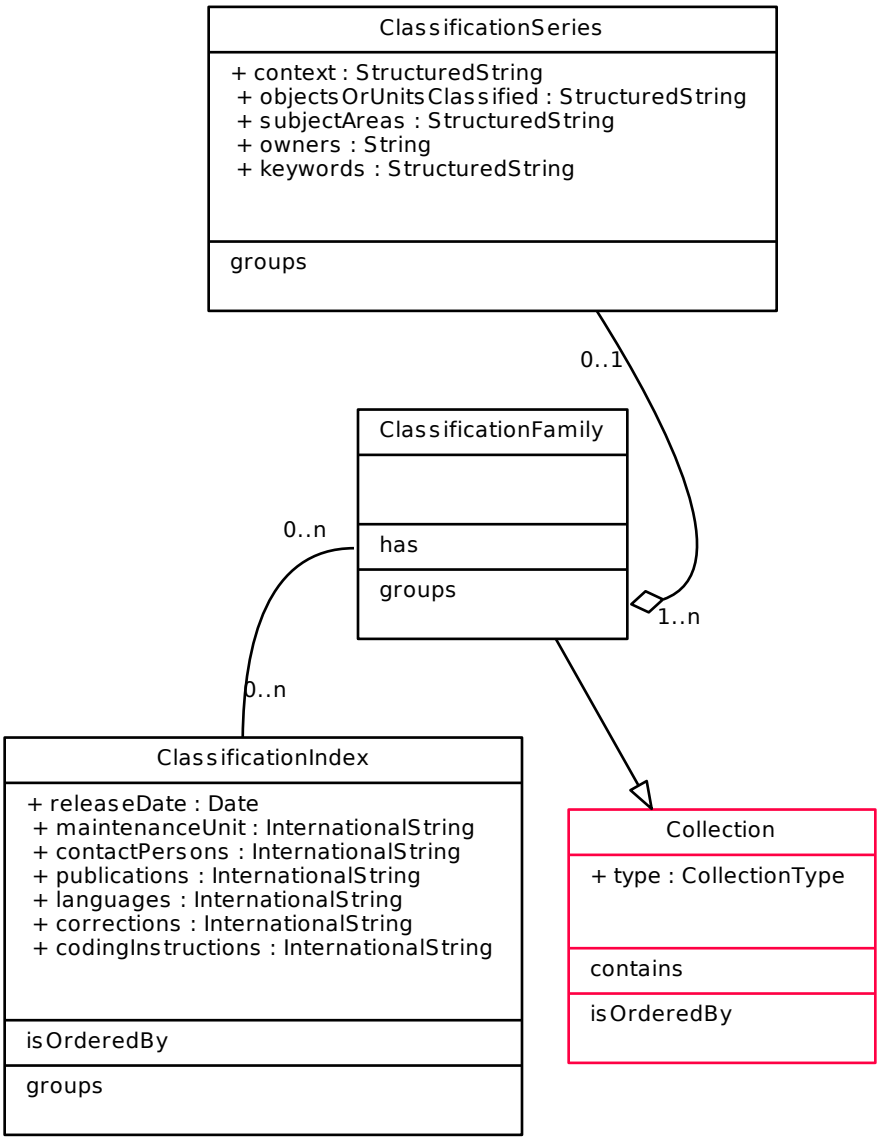
5.8.2 ClassificationFamily

A Classification Family is a group of Classification Series related from a particular point of view. The Classification Family is related by being based on a common Concept (e.g. economic activity).[GSIM1.1]

Extends

Collection

Graph



5.8.3 ClassificationIndex

A Classification Index is an ordered list (alphabetical, in code order etc) of Classification Index Entries. A Classification Index can relate to one particular or to several Statistical Classifications. [GSIM Statistical Classification Model]

Extends

Collection

Properties

Name	Type	Cardinality
releaseDate	Date	0..1
maintenanceUnit	InternationalString	0..1
contactPersons	InternationalString	0..1
publications	InternationalString	0..n
languages	InternationalString	0..n
corrections	InternationalString	0..n
codingInstructions	InternationalString	0..n

releaseDate

Date when the current version of the Classification Index was released.

maintenanceUnit

The unit or group of persons within the organisation responsible for the Classification Index, i.e. for adding, changing or deleting Classification Index Entries.

contactPersons

Person(s) who may be contacted for additional information about the Classification Index.

publications

A list of the publications in which the Classification Index has been published.

languages

A Classification Index can exist in several languages. Indicates the languages available. If a Classification Index exists in several languages, the number of entries in each language may be different, as the number of terms describing the same phenomenon can change from one language to another. However, the same phenomena should be described in each language.

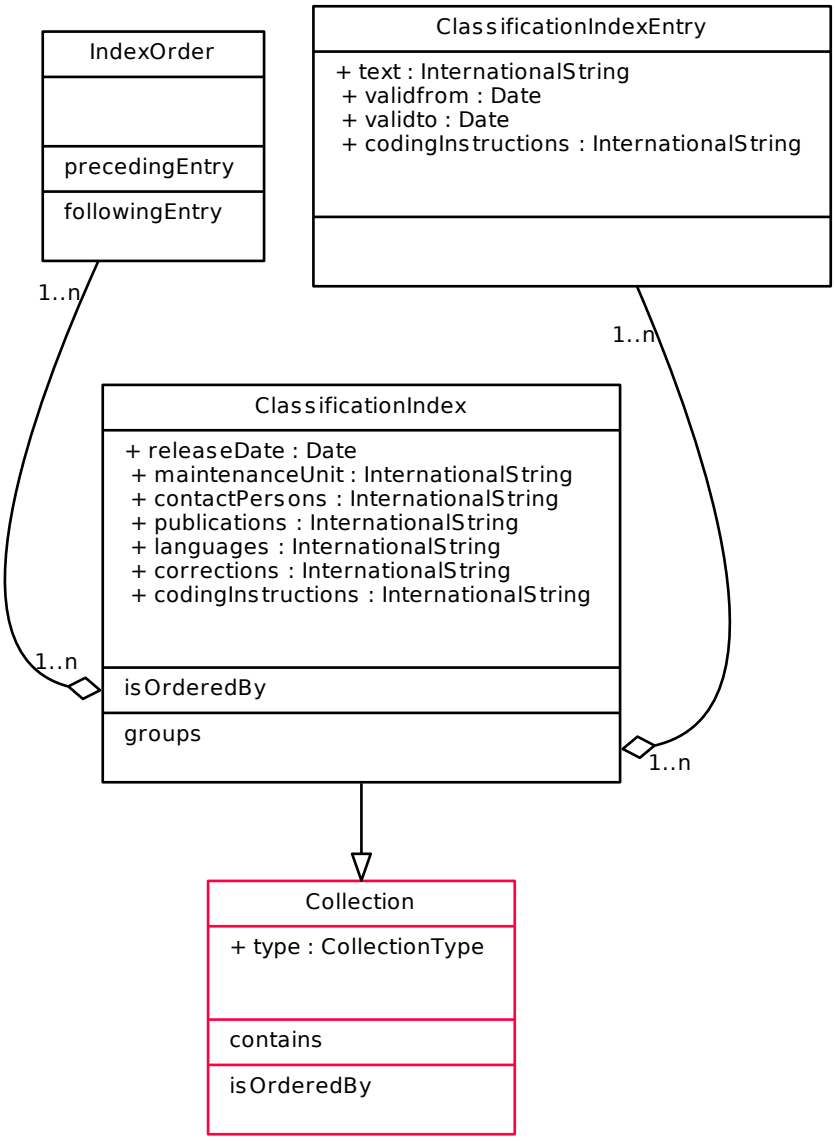
corrections

Verbal summary description of corrections, which have occurred within the Classification Index. Corrections include changing the item code associated with an Classification Index Entry.

codingInstructions

Additional information which drives the coding process for all entries in a Classification Index.

Graph



5.8.4 ClassificationIndexEntry

A Classification Index Entry is a word or a short text (e.g. the name of a locality, an economic activity or an occupational title) describing a type of object/unit or object property to which a Classification Item applies, together with the code of the corresponding Classification Item. Each Classification Index Entry typically refers to one item of the Statistical Classification. Although a Classification Index Entry may be associated with a Classification Item at any Level of a Statistical Classification, Classification Index Entries are normally associated with items at the lowest Level.

Extends

Member

Properties

Name	Type	Cardinality
text	InternationalString	1..n
validfrom	Date	0..1
validto	Date	0..1
codingInstructions	InternationalString	0..n

text

Text describing the type of object/unit or object property.

validfrom

Date from which the Classification Index Entry became valid. The date must be defined if the Classification Index Entry belongs to a floating Classification Index.

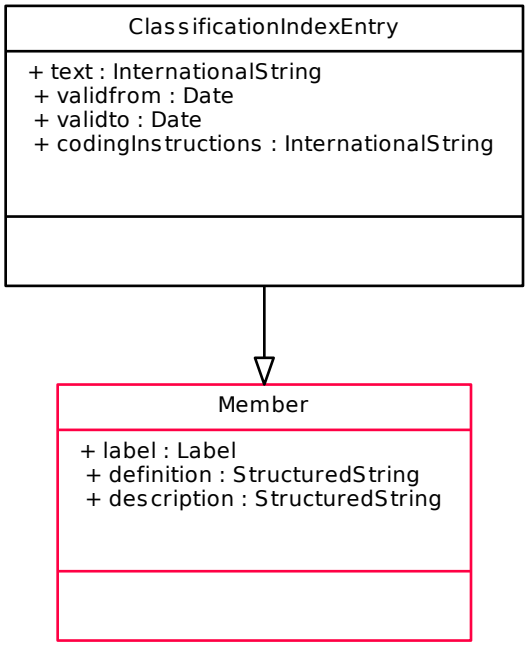
validto

Date at which the Classification Index Entry became invalid. The date must be defined if the Classification Index Entry belongs to a floating Classification Index and is no longer valid.

codingInstructions

Additional information which drives the coding process. Required when coding is dependent upon one or many other factors.

Graph



5.8.5 ClassificationItem

A Classification Item represents a Category at a certain Level within a Statistical Classification.

Extends

Node

Properties

Name	Type	Cardinality
isValid	xs:boolean	0..1
isGenerated	xs:boolean	0..1
explanatoryNotes	StructuredString	0..n
futureNotes	InternationalString	0..n
changeLog	InternationalString	0..1
changeFromPreviousVersion	InternationalString	0..1
validDate	Date	0..1
officialName	Name	1..1

isValid

Indicates whether or not the item is currently valid. If updates are allowed in the Statistical Classification, an item may be restricted in its validity, i.e. it may become valid or invalid after the Statistical Classification has been released.

isGenerated

Indicates whether or not the item has been generated to make the level to which it belongs complete

explanatoryNotes

A Classification Item may be associated with explanatory notes, which further describe and clarify the contents of the Category. Explanatory notes consist of: General note: Contains either additional information about the Category, or a general description of the Category, which is not structured according to the "includes", "includes also", "excludes" pattern. Includes: Specifies the contents of the Category. Includes also: A list of borderline cases, which belong to the described Category. Excludes: A list of borderline cases, which do not belong to the described Category. Excluded cases may contain a reference to the Classification Items to which the excluded cases belong.

futureNotes

The future events describe a change (or a number of changes) related to an invalid item. These changes may e.g. have turned the now invalid item into one or several successor items. This allows the possibility to follow successors of the item in the future.

changeLog

Describes the changes, which the item has been subject to during the life time of the actual Statistical Classification.

changeFromPreviousVersion

Describes the changes, which the item has been subject to from the previous version to the actual Statistical Classification

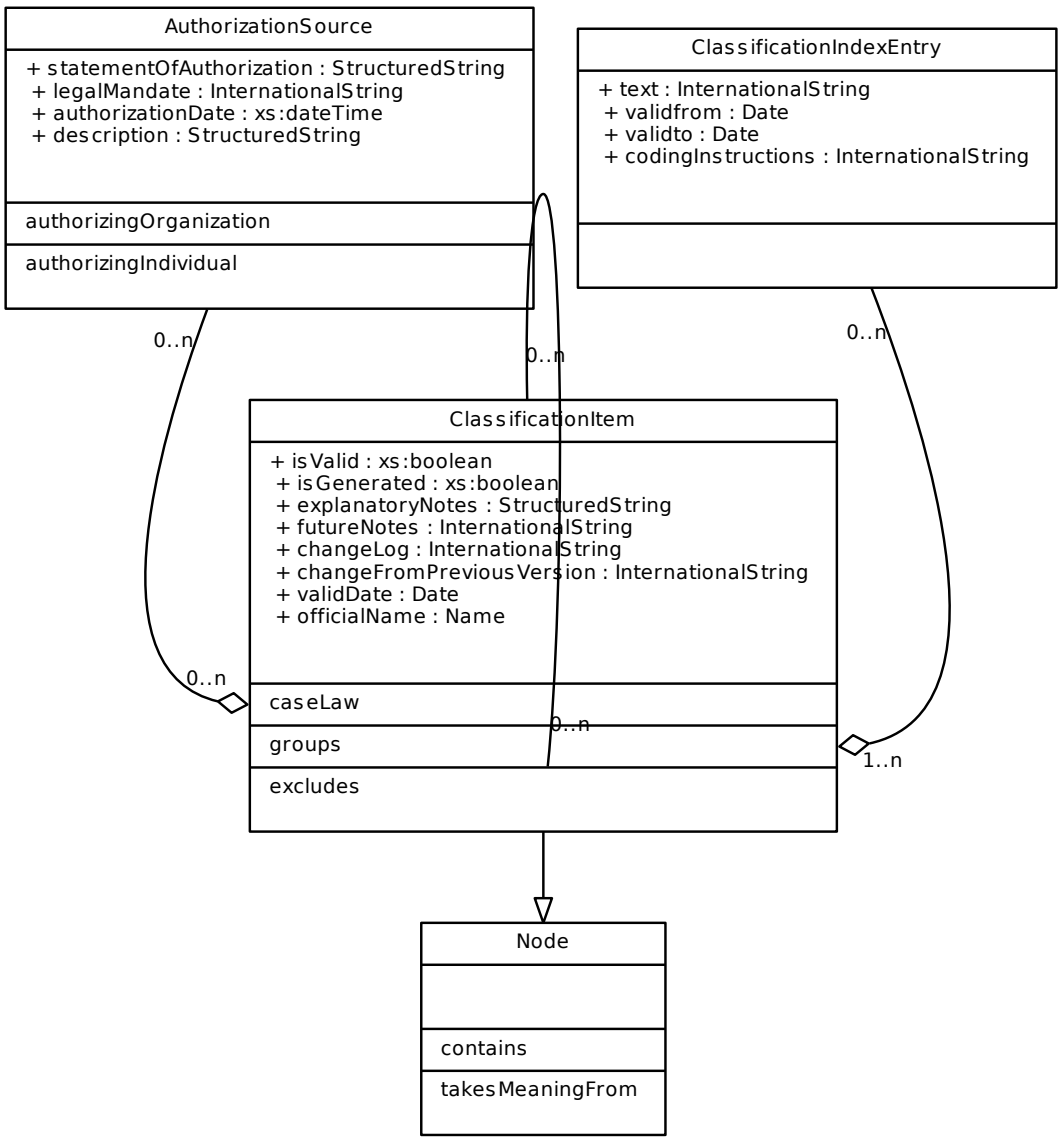
validDate

Dates for which the classification is valid. Date from which the item became valid. The date must be defined if the item belongs to a floating Statistical classification. Date at which the item became invalid. The date must be defined if the item belongs to a floating Statistical classification and is no longer valid

officialName

A Classification Item has a name as provided by the owner or maintenance unit. The name describes the content of the category. The name is unique within the Statistical Classification to which the item belongs, except for categories that are identical at more than one level in a hierarchical classification

Graph



5.8.6 ClassificationSeries

A Classification Series is an ensemble of one or more Statistical Classifications, based on the same concept, and related to each other as versions or updates. Typically, these Statistical Classifications have the same name (for example, ISIC or ISCO).

Extends

Collection

Properties

Name	Type	Cardinality
context	StructuredString	0..1
objectsOrUnitsClassified	StructuredString	1..1
subjectAreas	StructuredString	1..1
owners	String	0..1
keywords	StructuredString	0..n

context

ClassificationSeries can be designed in a specific context.

objectsOrUnitsClassified

A ClassificationSeries is designed to classify a specific type of object/unit according to a specific attribute.

subjectAreas

Areas of statistics in which the ClassificationSeries is implemented.

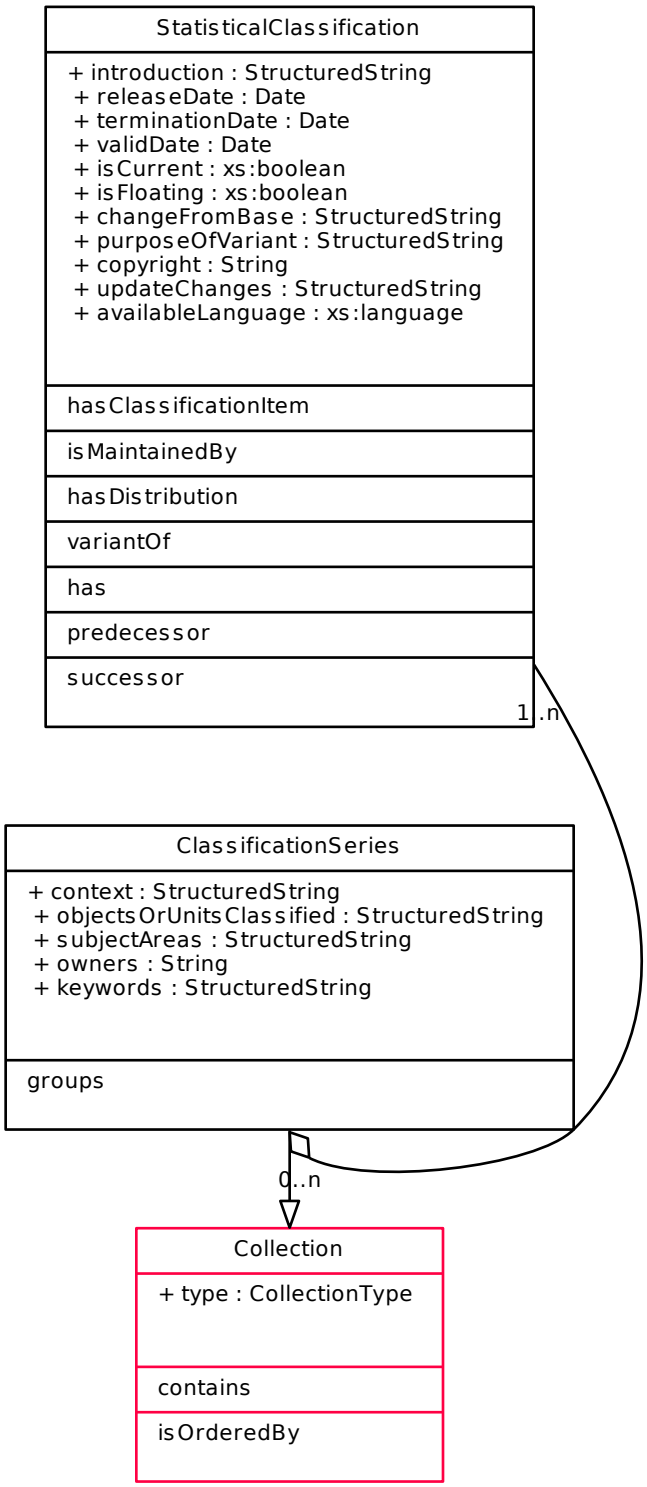
owners

The statistical office or other authority, which created and maintains the StatisticalClassification (s) related to the ClassificationSeries. A ClassificationSeries may have several owners.

keywords

A ClassificationSeries can be associated with one or a number of keywords.

Graph


StatisticalClassification
+ introduction : StructuredString
+ releaseDate : Date
+ terminationDate : Date
+ validDate : Date
+ isCurrent : xs:boolean
+ isFloating : xs:boolean
+ changeFromBase : StructuredString
+ purposeOfVariant : StructuredString
+ copyright : String
+ updateChanges : StructuredString
+ availableLanguage : xs:language
hasClassificationItem
isMaintainedBy
hasDistribution
variantOf
has
predecessor
successor
1..n
ClassificationSeries
+ context : StructuredString
+ objectsOrUnitsClassified : StructuredString
+ subjectAreas : StructuredString
+ owners : String
+ keywords : StructuredString
groups
0..n
Collection
+ type : CollectionType
contains
isOrderedBy

5.8.7 Code

A Designation for a Category.

Extends

Designation

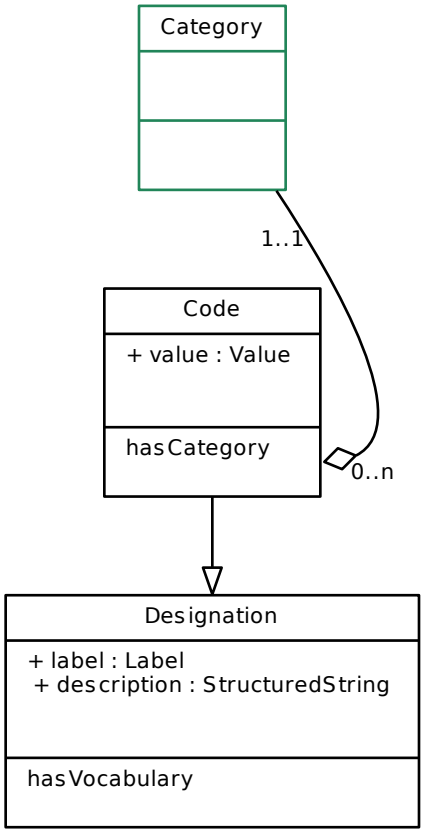
Properties

Name	Type	Cardinality
value	Value	1..1

value

Specified value of the code

Graph



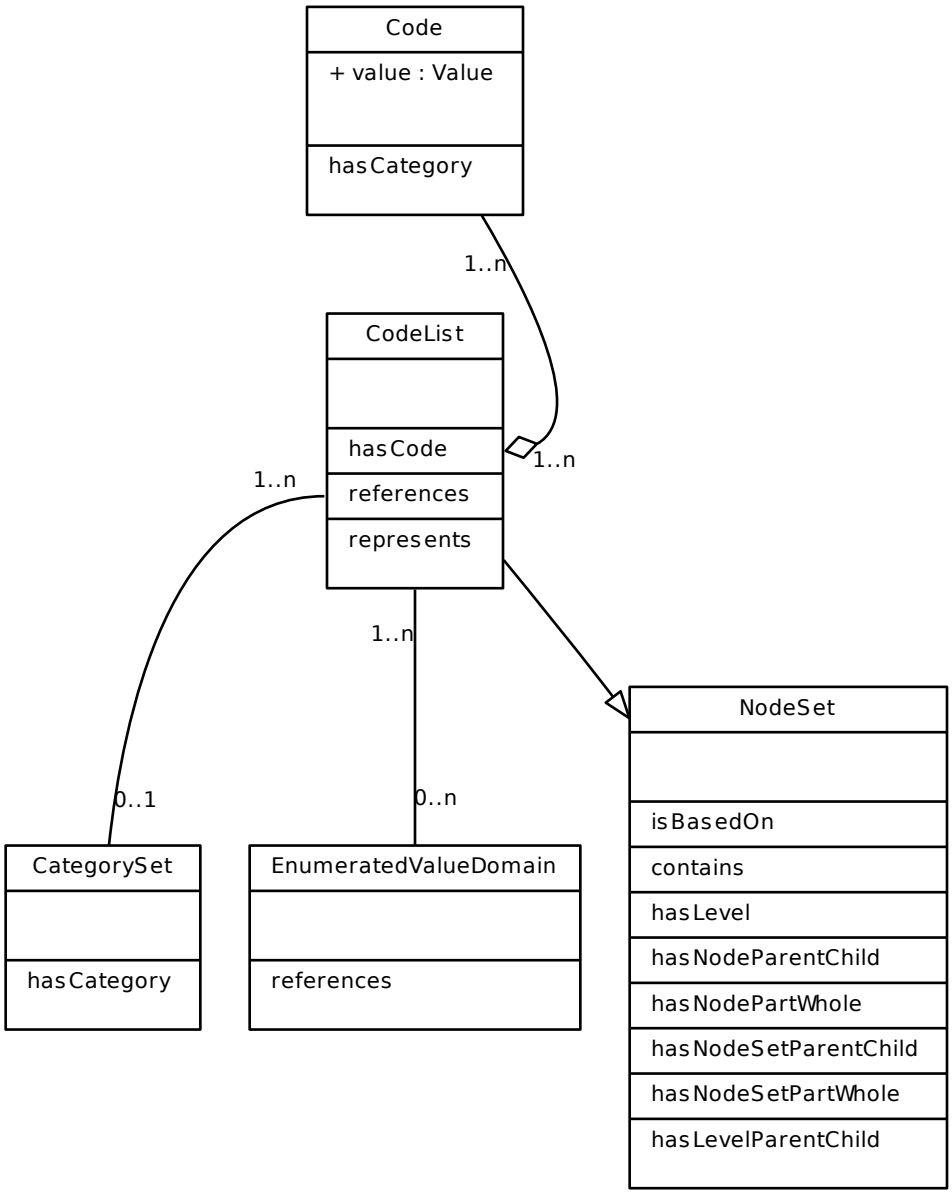
5.8.8 CodeList

A list of Codes and associated Categories. May be flat or hierarchical.

Extends

NodeSet

Graph



5.8.9 CorrespondenceTable

A Correspondence Table expresses a relationship between two NodeSets.

Extends

OrderedCollectionCorrespondence

Properties

Name	Type	Cardinality
owners	String	0..1
maintenanceUnit	String	0..1
contactPersons	String	0..n
publications	StructuredString	0..n
effectivePeriod	Date	0..1

owners

The statistical office, other authority or section that created and maintains the Correspondence Table. A Correspondence Table may have several owners.

maintenanceUnit

The unit or group of persons who are responsible for the Correspondence Table, i.e. for maintaining and updating it.

contactPersons

The person(s) who may be contacted for additional information about the Correspondence Table.

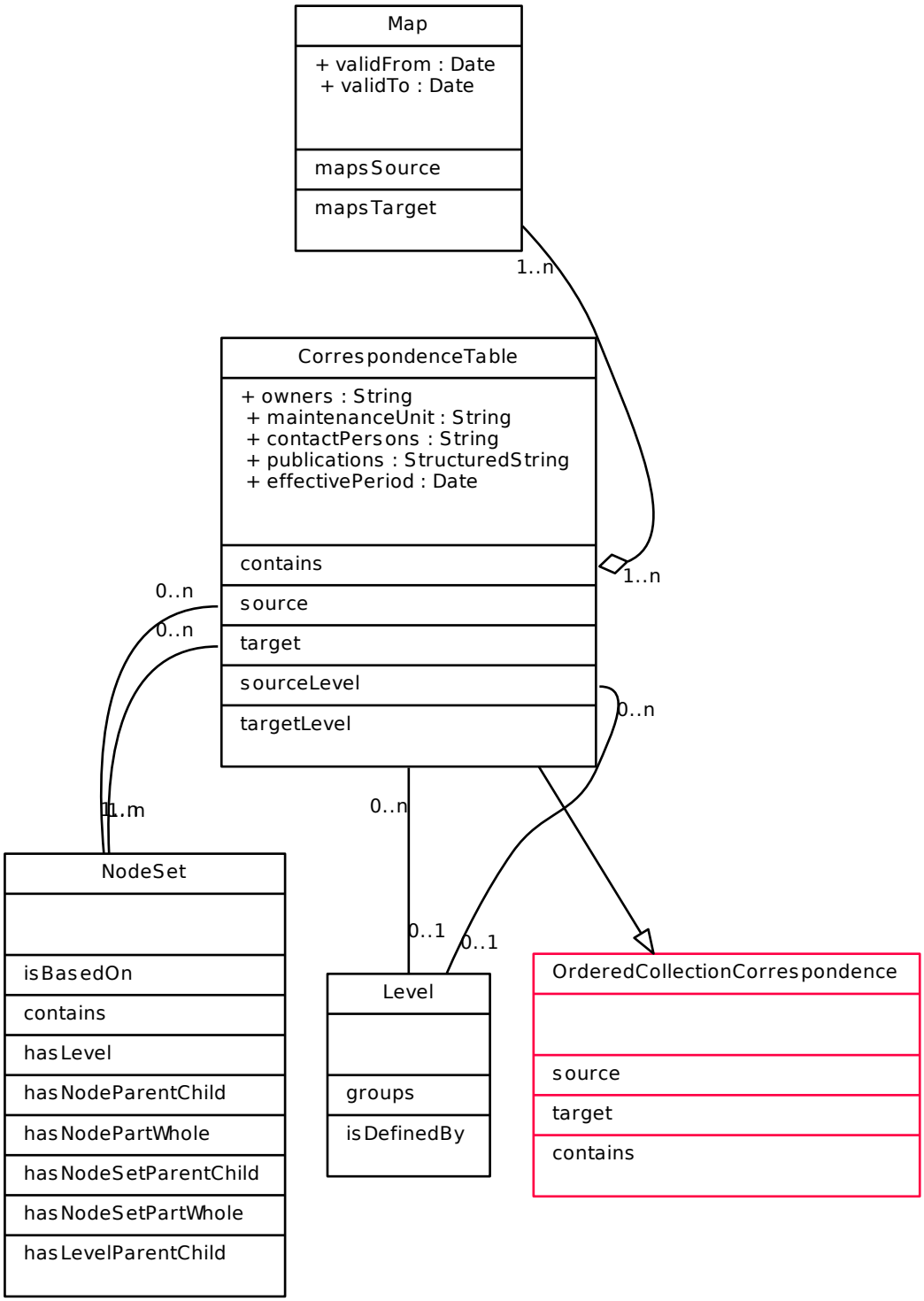
publications

A list of the publications in which the Correspondence Table has been published.

effectivePeriod

Effective period of validity of the CorrespondenceTable. The correspondence table expresses the relationships between the two NodeSets as they existed on the period specified in the table.

Graph


Map
+ validFrom : Date
+ validTo : Date
mapsSource
mapsTarget
1..n
CorrespondenceTable
+ owners : String
+ maintenanceUnit : String
+ contactPersons : String
+ publications : StructuredString
+ effectivePeriod : Date
contains
source
target
sourceLevel
targetLevel
1..n
0..n
0..n
0..n
1..m
0..n
0..1
0..1
NodeSet
isBasedOn
contains
hasLevel
hasNodeParentChild
hasNodePartWhole
hasNodeSetParentChild
hasNodeSetPartWhole
hasLevelParentChild
Level
groups
isDefinedBy
OrderedCollectionCorrespondence
source
target
contains

5.8.10 DataType

Set of distinct values, characterized by properties of those values, and by operations on those values. (From ISO/IEC 11404 - General purpose datatypes)

Extends

AnnotatedIdentifiable

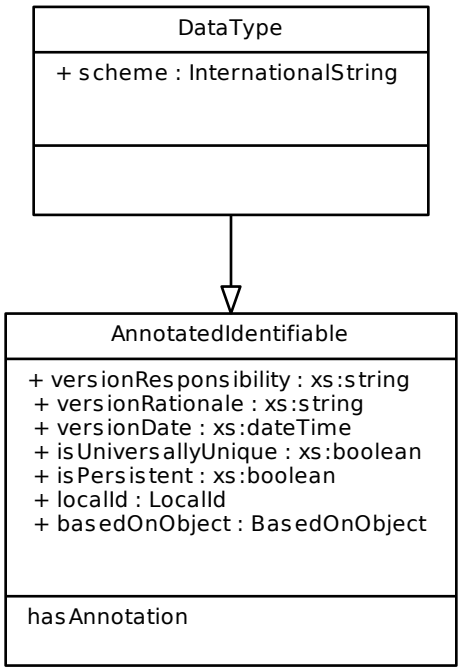
Properties

Name	Type	Cardinality
scheme	InternationalString	1..1

scheme

ISO 11404, Excel, SAS, R, etc.

Graph



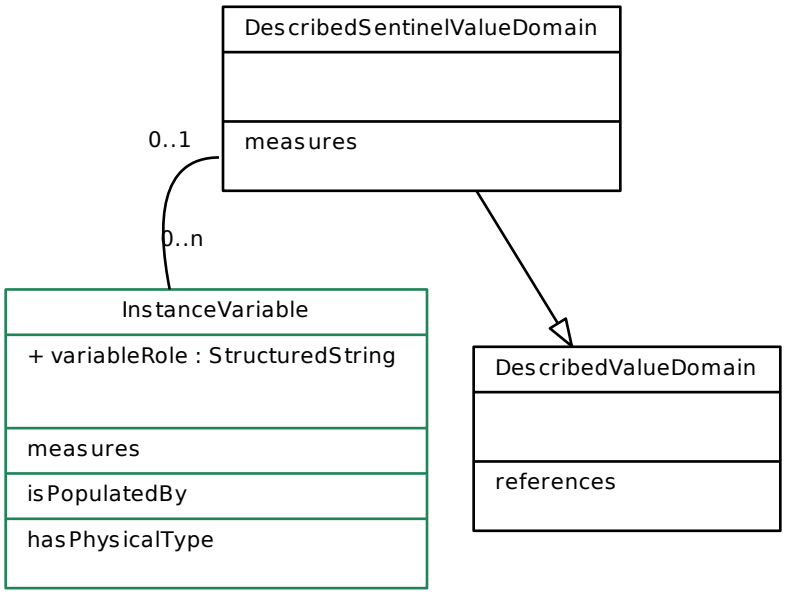
5.8.11 DescribedSentinelValueDomain

A described value domain whose values are used only for the processing of data after capture but before dissemination.

Extends

DescribedValueDomain

Graph



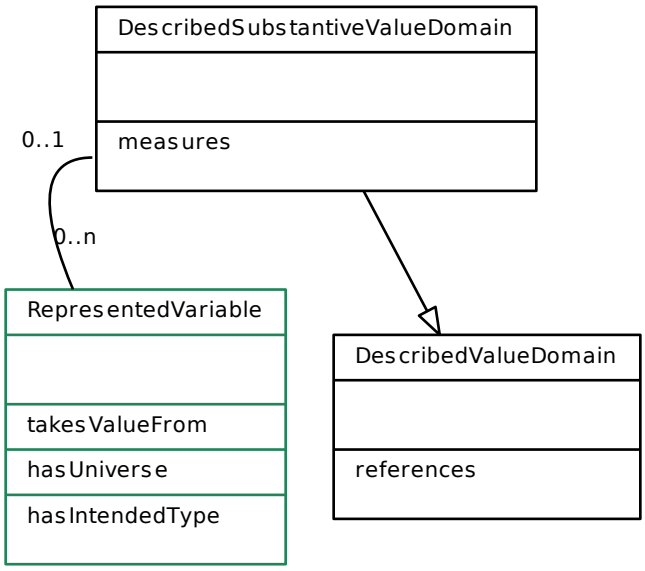
5.8.12 DescribedSubstantiveValueDomain

A described value domain whose values are associated with the scientific questions of interest in a study.

Extends

DescribedValueDomain

Graph



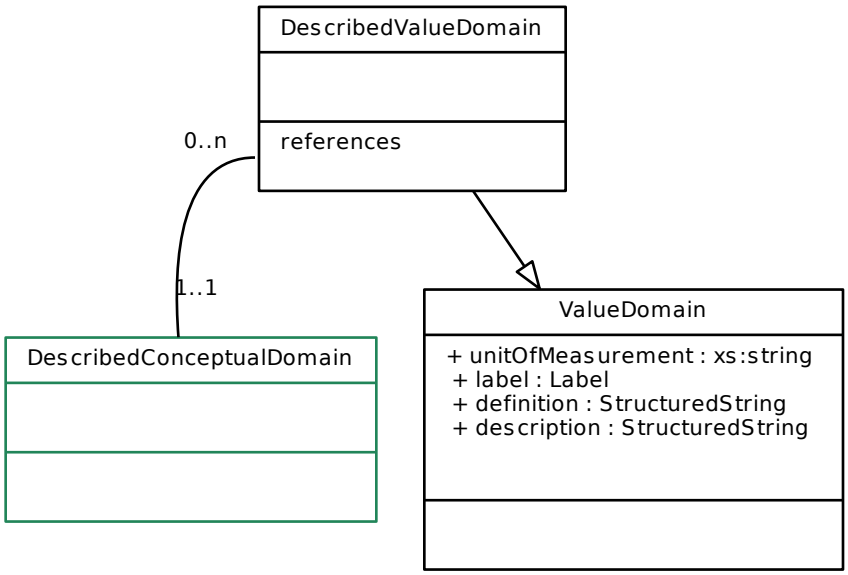
5.8.13 DescribedValueDomain

A Value Domain defined by an expression. [GSIM 1.1]

Extends

ValueDomain

Graph



5.8.14 Designation

The name given to an object for identification.

Extends

AnnotatedIdentifiable

Properties

Name	Type	Cardinality
label	Label	0..n
description	StructuredString	0..1

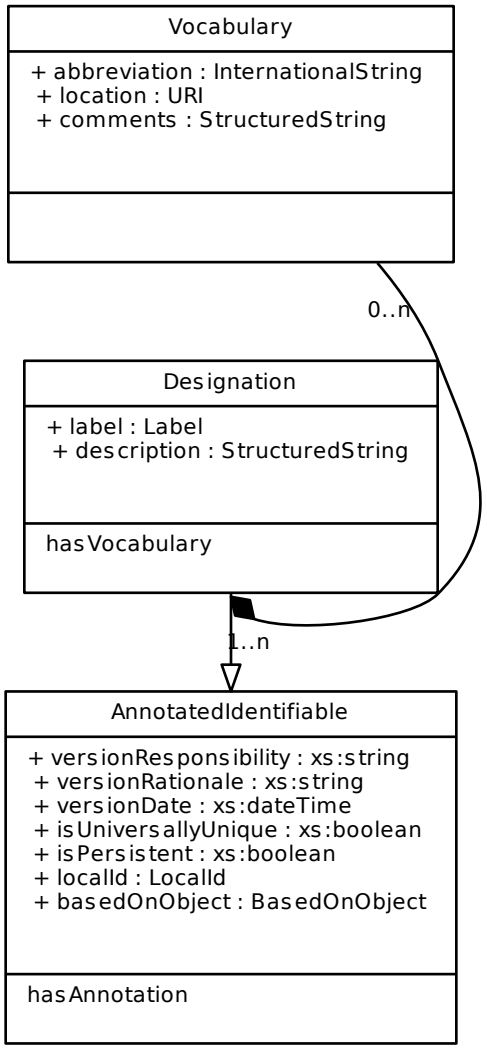
label

A display label for the Designation. May be expressed in multiple languages. Repeat for labels with different content, for example, labels with differing length limitations.

description

A description of the purpose or use of the Designation. May be expressed in multiple languages and supports the use of structured content.

Graph



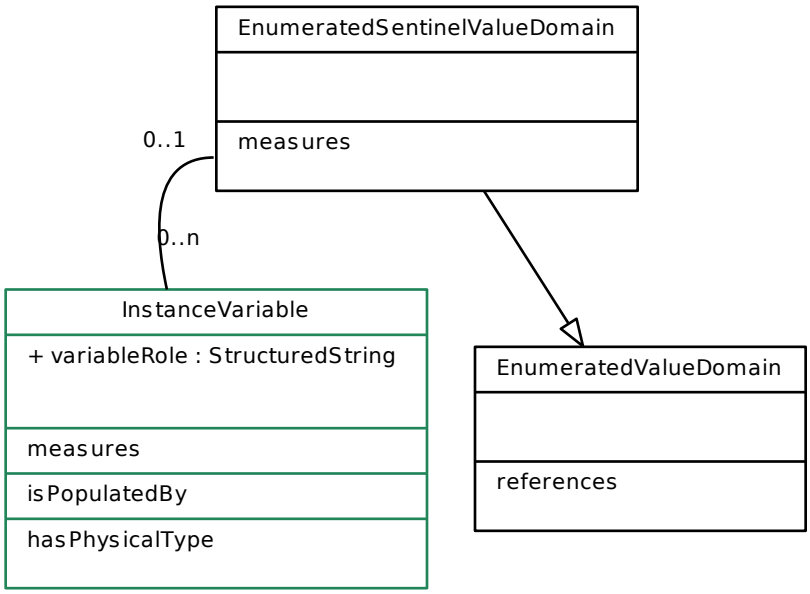
5.8.15 EnumeratedSentinelValueDomain

An enumerated value domain whose values are used only for the processing of data after capture but before dissemination.

Extends

EnumeratedValueDomain

Graph



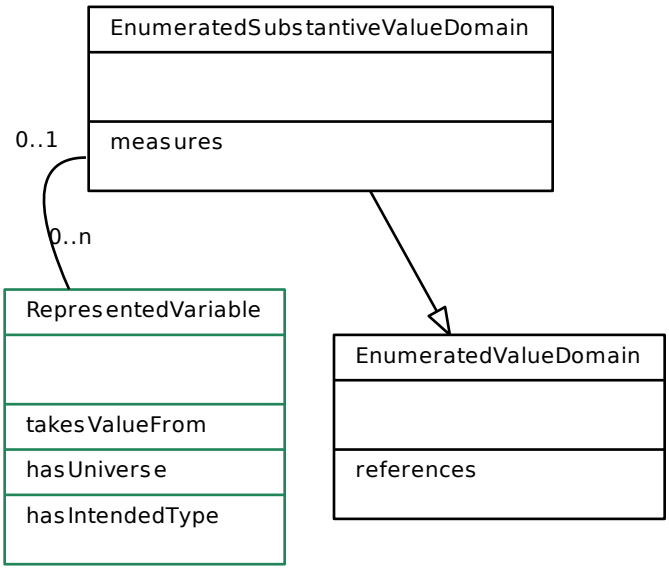
5.8.16 EnumeratedSubstantiveValueDomain

An enumerated value domain whose values are associated with the scientific questions of interest in a study.

Extends

EnumeratedValueDomain

Graph



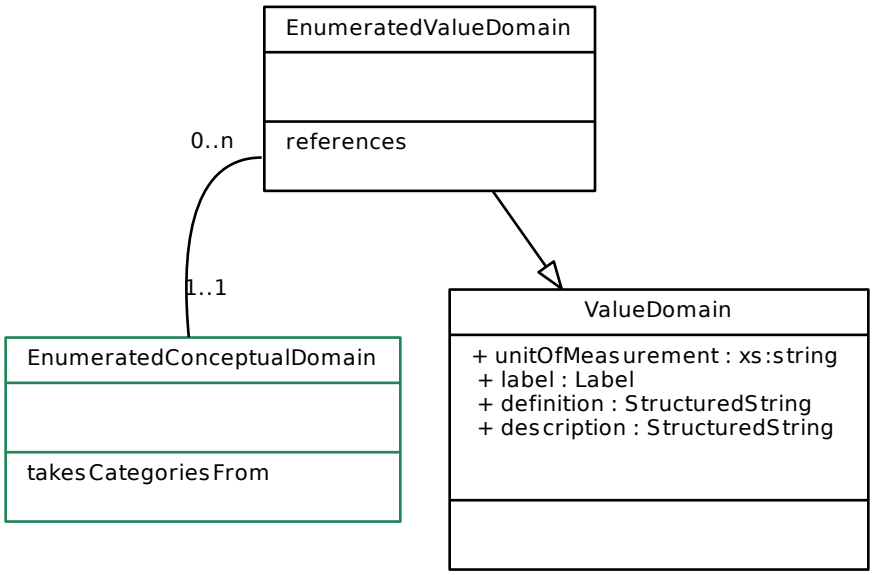
5.8.17 EnumeratedValueDomain

A Value domain expressed as a list of categories and associated codes.

Extends

ValueDomain

Graph



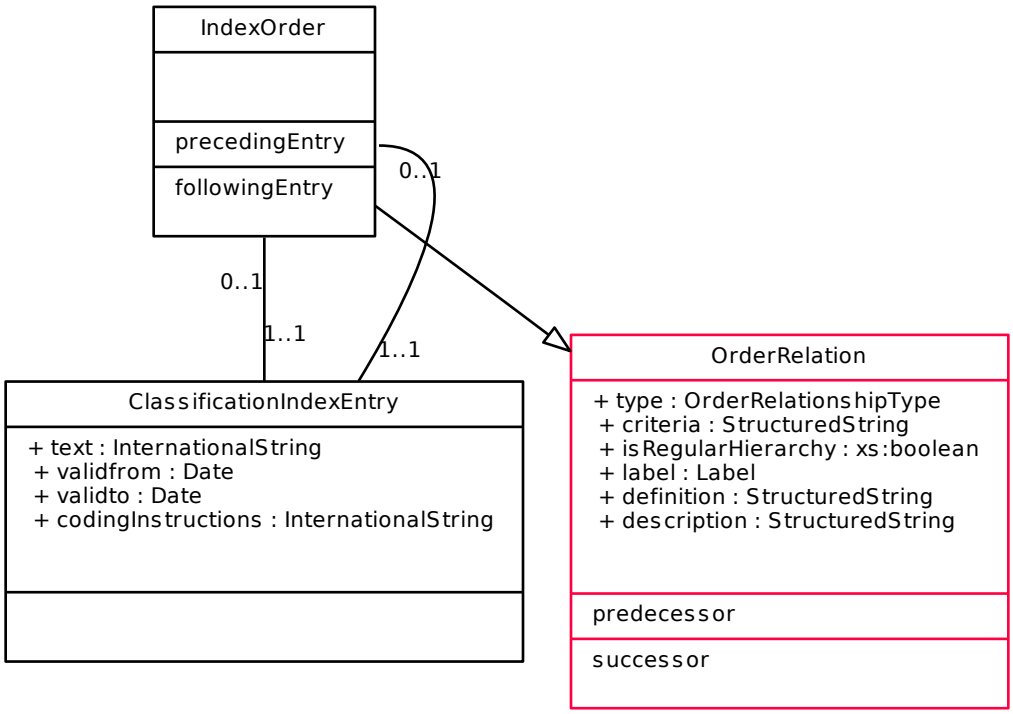
5.8.18 IndexOrder

Indexing order, defined either by predecessor-successor pairs or by a criteria (e.g. alphabetical, in code order, etc.)

Extends

OrderRelation

Graph



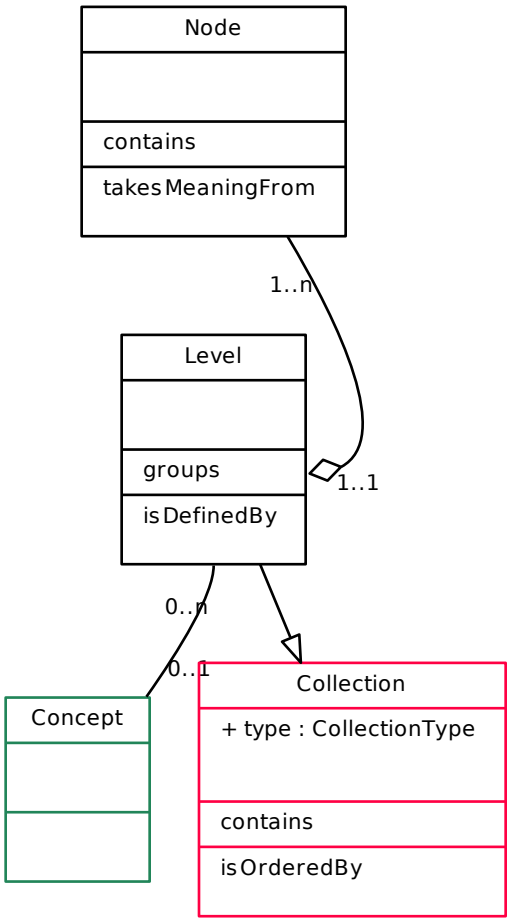
5.8.19 Level

The Level describes the nesting structure of a hierarchical collection.

Extends

Collection

Graph



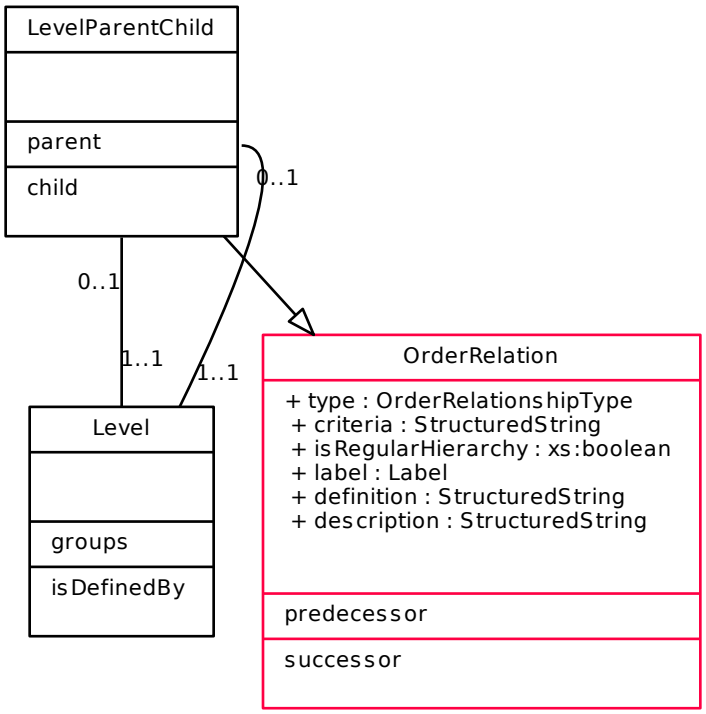
5.8.20 LevelParentChild

Parent-child specialization of OrderRelation between Levels within a NodeSet. The inherited type property is set to "total" to specify that the parent-child relationships among Levels in any given NodeSet define a linear sequence.

Extends

OrderRelation

Graph



5.8.21 Map

Extends

OrderedMemberCorrespondence

Properties

Name	Type	Cardinality
validFrom	Date	0..1
validTo	Date	0..1

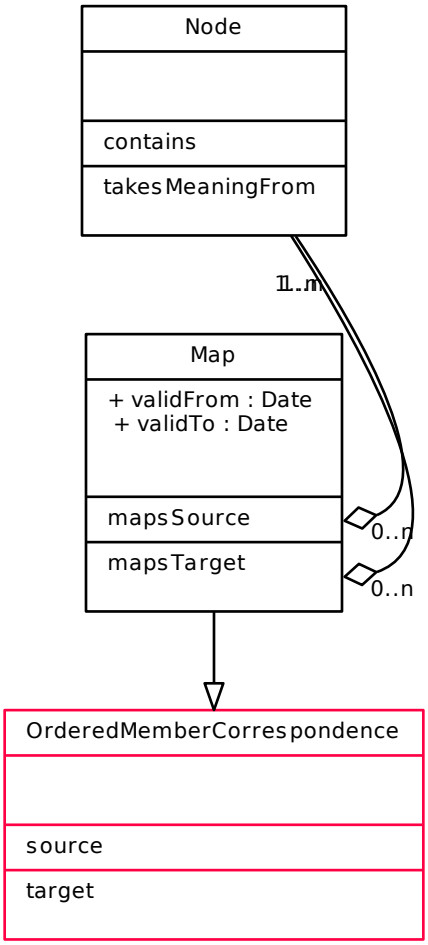
validFrom

Date from which the Map became valid. The date must be defined if the Map belongs to a floating CorrespondenceTable.

validTo

Date at which the Map became invalid. The date must be defined if the Map belongs to a floating Correspondence Table and is no longer valid.

Graph



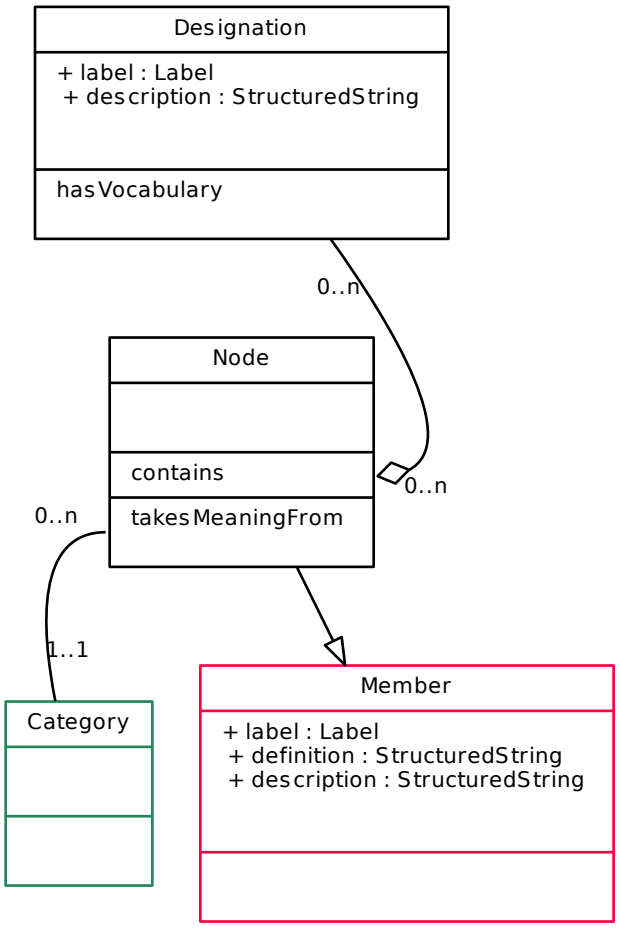
5.8.22 Node

A combination of a category and related attributes.

Extends

Member

Graph



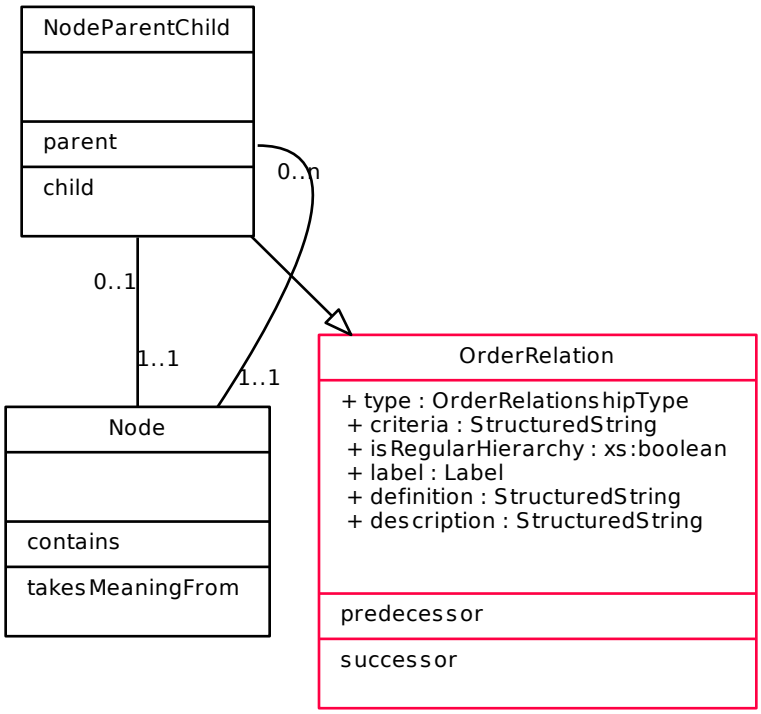
5.8.23 NodeParentChild

Parent-child specialization of OrderRelation between Nodes within a NodeSet. The inherited type property is set to "partial" to specify that the parent-child relationships among Nodes define a tree structure.

Extends

OrderRelation

Graph



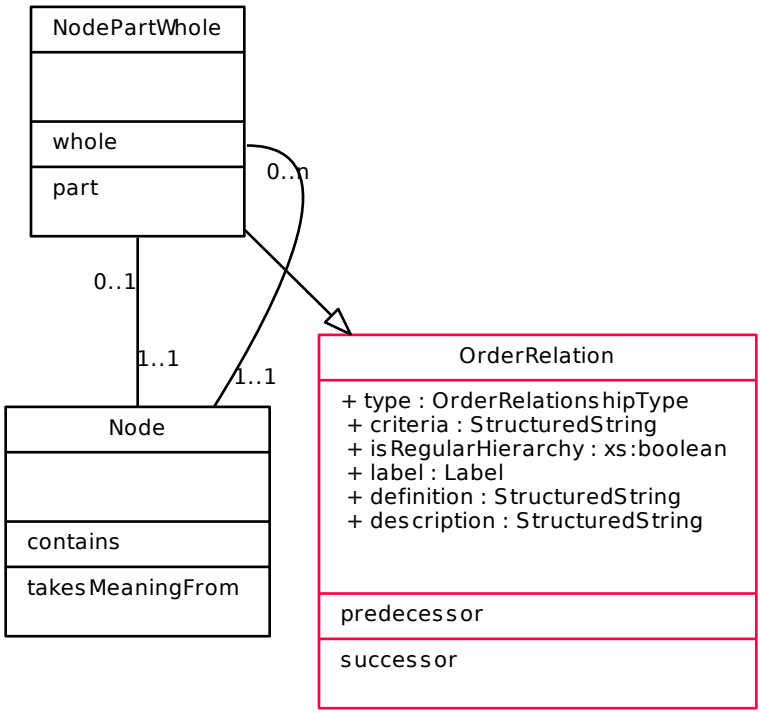
5.8.24 NodePartWhole

Part-whole specialization of OrderRelation between Nodes within a NodeSet. The inherited type property is set to "partial" to specify that the part-whole relationships among Nodes define a tree structure.

Extends

OrderRelation

Graph



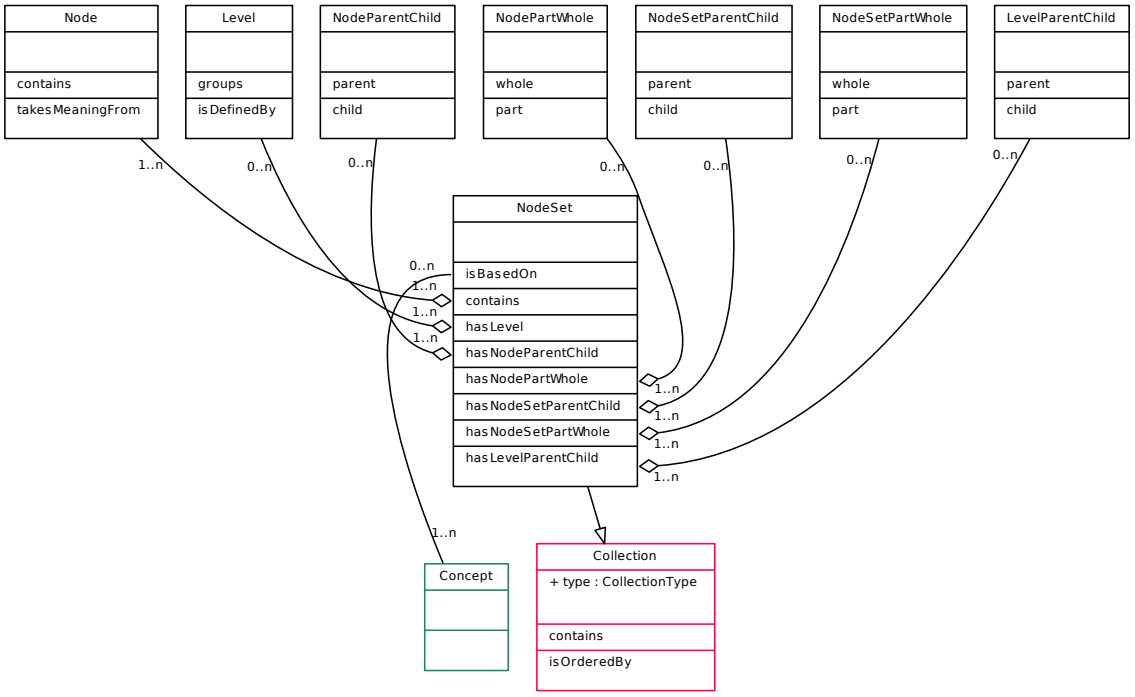
5.8.25 NodeSet

A NodeSet is a set of Nodes, which could be organized into a hierarchy of Levels.

Extends

Collection

Graph



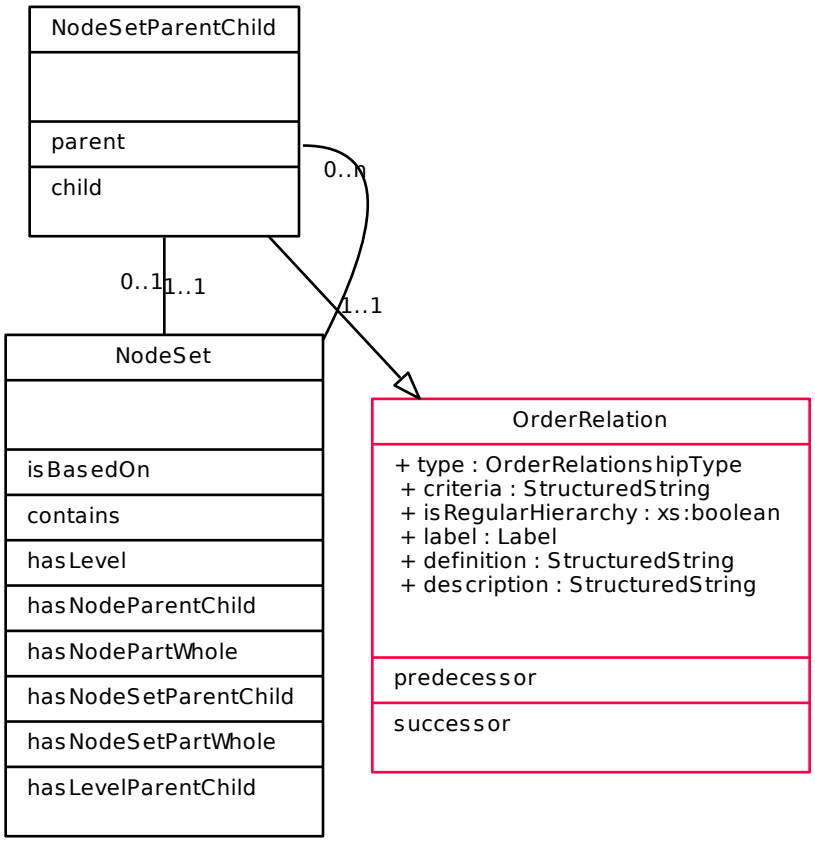
5.8.26 NodeSetParentChild

Parent-child specialization of OrderRelation between NodeSets. The inherited type property is set to "partial" to specify that the parent-child relationships among NodeSets define a tree structure.

Extends

OrderRelation

Graph



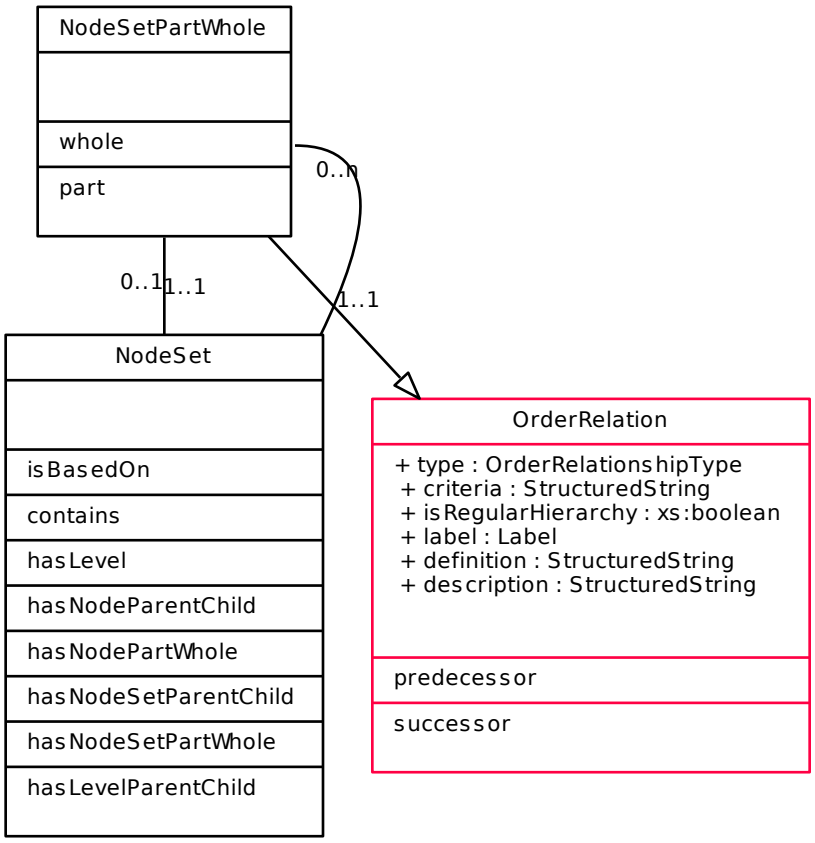
5.8.27 NodeSetPartWhole

Part-whole specialization of OrderRelation between NodeSets. The inherited type property is set to "partial" to specify that the part-whole relationships among NodeSets define a tree structure.

Extends

OrderRelation

Graph



5.8.28 Sign

Something that suggests the presence or existence of a fact, condition, or quality.

Extends

AnnotatedIdentifiable

Properties

Name	Type	Cardinality
value	StructuredString	1..1
label	Label	0..n
description	StructuredString	0..1

value

The text representation

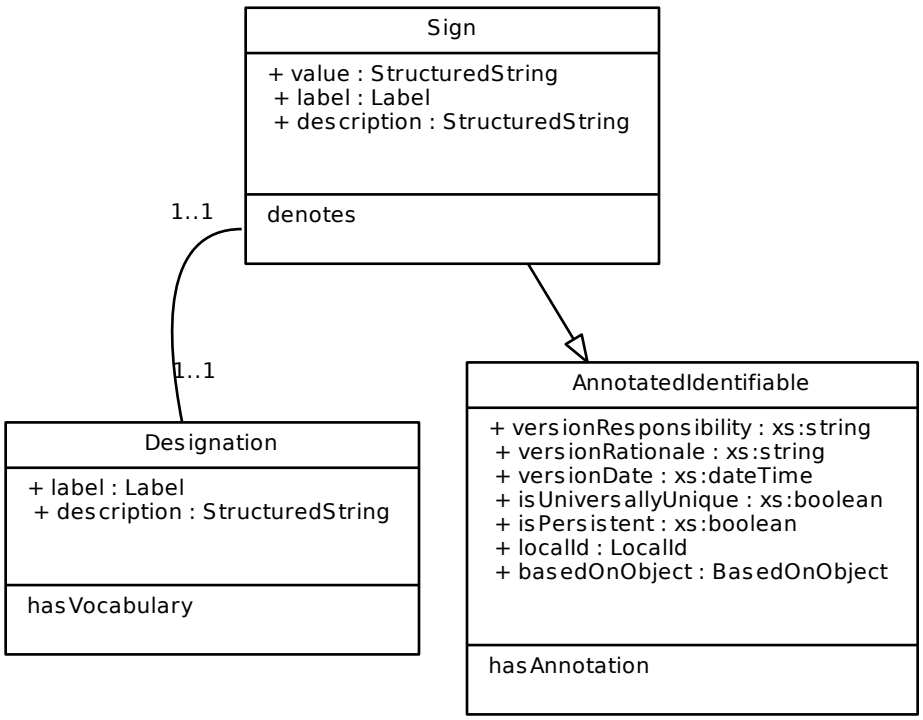
label

A display label for the object. May be expressed in multiple languages. Repeat for labels with different content, for example, labels with differing length limitations.

description

A description of the content and purpose of the object. May be expressed in multiple languages and supports the use of structured content.

Graph



5.8.29 StatisticalClassification

A Statistical Classification is a set of Categories which may be assigned to one or more variables registered in statistical surveys or administrative files, and used in the production and dissemination of statistics. The Categories at each Level

of the classification structure must be mutually exclusive and jointly exhaustive of all objects/units in the population of interest. (Source: GSIM StatisticalClassification)

Extends

NodeSet

Properties

Name	Type	Cardinality
introduction	StructuredString	0..1
releaseDate	Date	0..1
terminationDate	Date	0..1
validDate	Date	0..1
isCurrent	xs:boolean	0..1
isFloating	xs:boolean	0..1
changeFromBase	StructuredString	0..1
purposeOfVariant	StructuredString	0..1
copyright	String	0..n
updateChanges	StructuredString	0..n
availableLanguage	xs:language	0..n

introduction

The introduction provides a detailed description of the Statistical Classification, the background for its creation, the classification variable and objects/units classified, classification rules etc. (Source: GSIM StatisticalClassification

releaseDate

Date the Statistical Classification was released

terminationDate

Date on which the Statistical Classification was superseded by a successor version or otherwise ceased to be valid. (Source: GSIM Statistical Classification)

validDate

The date the statistical classification enters production use.

isCurrent

Indicates if the Statistical Classification is currently valid.

isFloating

Indicates if the Statistical Classification is a floating classification. In a floating statistical classification, a validity period should be defined for all Classification Items which will allow the display of the item structure and content at different points of time. (Source: GSIM StatisticalClassification/Floating

changeFromBase

Describes the relationship between the variant and its base Statistical Classification, including regroupings, aggregations added and extensions. (Source: GSIM StatisticalClassification/Changes from base Statistical Classification)

purposeOfVariant

If the Statistical Classification is a variant, notes the specific purpose for which it was developed. (Source: GSIM StatisticalClassification/Purpose of variant)

copyright

Copyright of the statistical classification.

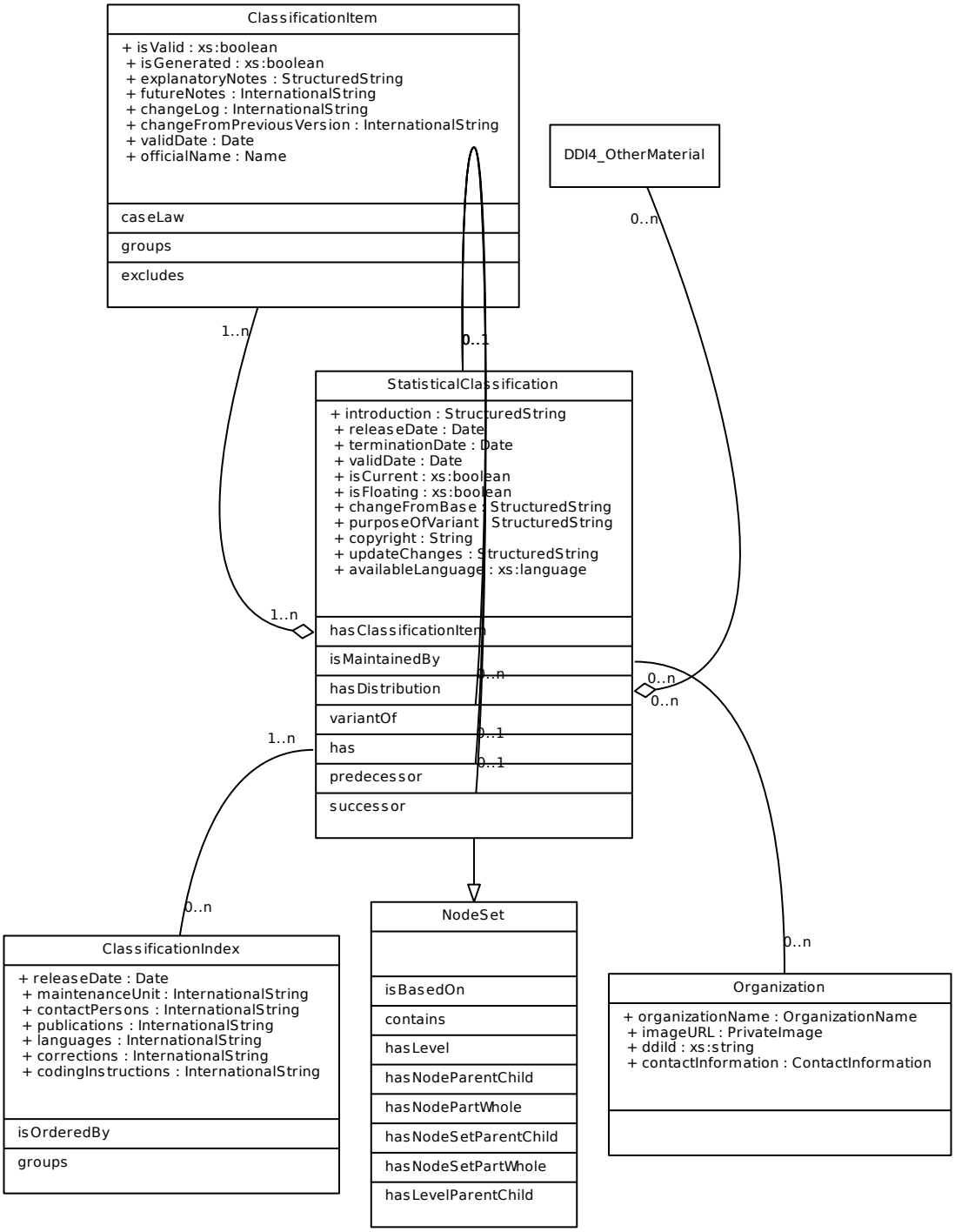
updateChanges

Summary description of changes which have occurred since the most recent classification version or classification update came into force.

availableLanguage

A list of languages in which the Statistical Classification is available. Repeat for each langauge.

Graph


ClassificationItem
+ isValid : xs:boolean
+ isGenerated : xs:boolean
+ explanatoryNotes : StructuredString
+ futureNotes : InternationalString
+ changeLog : InternationalString
+ changeFromPreviousVersion : InternationalString
+ validDate : Date
+ officialName : Name
caseLaw
groups
excludes
DDI4_OtherMaterial
0..n
1..n
0..1
StatisticalClassification
+ introduction : StructuredString
+ releaseDate : Date
+ terminationDate : Date
+ validDate : Date
+ isCurrent : xs:boolean
+ isFloating : xs:boolean
+ changeFromBase : StructuredString
+ purposeOfVariant : StructuredString
+ copyright : String
+ updateChanges : StructuredString
+ availableLanguage : xs:language
1..n
hasClassificationItem
isMaintainedBy
0..n
hasDistribution
0..n
0..n
variantOf
1..n
has
0..1
predecessor
0..1
successor
0..n
ClassificationIndex
+ releaseDate : Date
+ maintenanceUnit : InternationalString
+ contactPersons : InternationalString
+ publications : InternationalString
+ languages : InternationalString
+ corrections : InternationalString
+ codingInstructions : InternationalString
isOrderedBy
groups
NodeSet
isBasedOn
contains
hasLevel
hasNodeParentChild
hasNodePartWhole
hasNodeSetParentChild
hasNodeSetPartWhole
hasLevelParentChild
0..n
Organization
+ organizationName : OrganizationName
+ imageURL : PrivateImage
+ ddiId : xs:string
+ contactInformation : ContactInformation

5.8.30 ValueDomain

The permitted range of values for a characteristic of a variable. [GSIM 1.1]

Extends

AnnotatedIdentifiable

Properties

Name	Type	Cardinality
unitOfMeasurement	xs:string	0..1
label	Label	0..n
definition	StructuredString	0..1
description	StructuredString	0..1

unitOfMeasurement

The unit in which the data values are measured (kg, pound, euro).

label

A display label for the object. May be expressed in multiple languages. Repeat for labels with different content, for example, labels with differing length limitations.

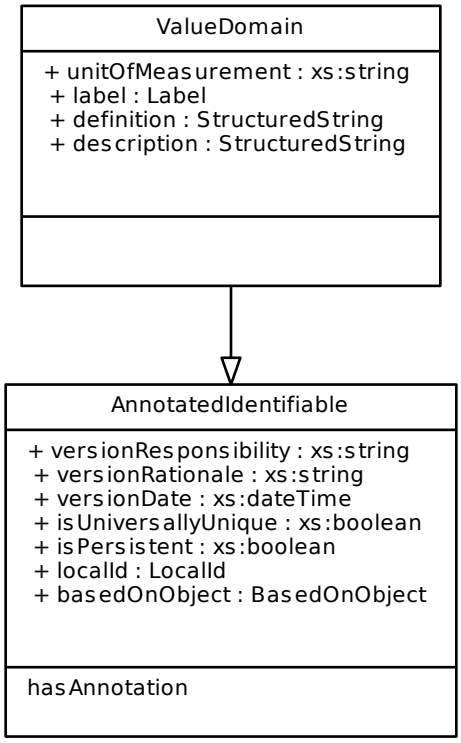
definition

A definition of the object. May be expressed in multiple languages and supports the use of structured content.

description

A description of the content and purpose of the object. May be expressed in multiple languages and supports the use of structured content.

Graph



5.8.31 Vocabulary

A vocabulary is an established list of standardized terminology for use in indexing and retrieval of information.

Extends

ConceptSystem

Properties

Name	Type	Cardinality
abbreviation	InternationalString	0..n
location	URI	0..1
comments	StructuredString	0..n

abbreviation

Abbreviation of vocabulary title.

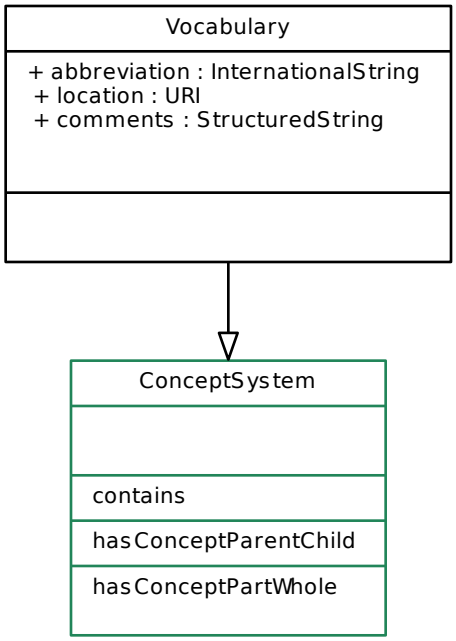
location

Location of external resource providing information about the vocabulary.

comments

Information for the user regarding the reasons for use of the vocabulary and appropriate usage constraints.

Graph



5.9 DDIUtility

DDI Utility Package Contents

5.10 DDIDocument

DDI Document Package Contents

5.11 Comparison

A maintainable module containing maps between objects of the same or similar type. Maps allow for pair-wise mapping of two objects by describing their similarities and differences in order to make assertions regarding their comparability. Currently maps allow for the comparison of concepts, variables, questions, categories, universes, and representations that have managed content (code, category, numeric, text, datetime and scale). These mapping(s) inform users on the comparability of two objects and facilitate automation. Note that all maps are pairwise, identifying two schemes and the correlation between two items in those schemes. Due to the complexity of some objects, multiple mappings may be required to cover details of the comparison of component parts, e.g. a QuestionMap may also have a related RepresentationMap. By using a set of pairwise comparisons, it is possible to describe complex correspondences - pairwise comparisons are easier to process. In addition to providing a standard name, label, and description, Comparison consists of a simple stack of comparison maps. Comparison maps are currently limited to those objects that can be referenced and are sufficiently structured to support a clear comparison. Contents

5.12 Collections

Generic collection structure to support managed and unmanaged collections containing both unique and non-unique members. It also supports the definition of correspondences, unordered and ordered, between collections and members. Such a generic structure can be used to model different types of groupings, from simple unordered sets to all sorts of hierarchies, nesting and ordered sets/bags. In addition, they can be extended with richer semantics (e.g. generic, partitive, and instance, among others). Contents

5.12.1 Collection

Collection container (set or bag). It could have an optional order relation (total or partial) associated to it to model linear order, hierarchies and nesting. A Collection is also a subtype of Member to allow for nested collections.

Extends

Member

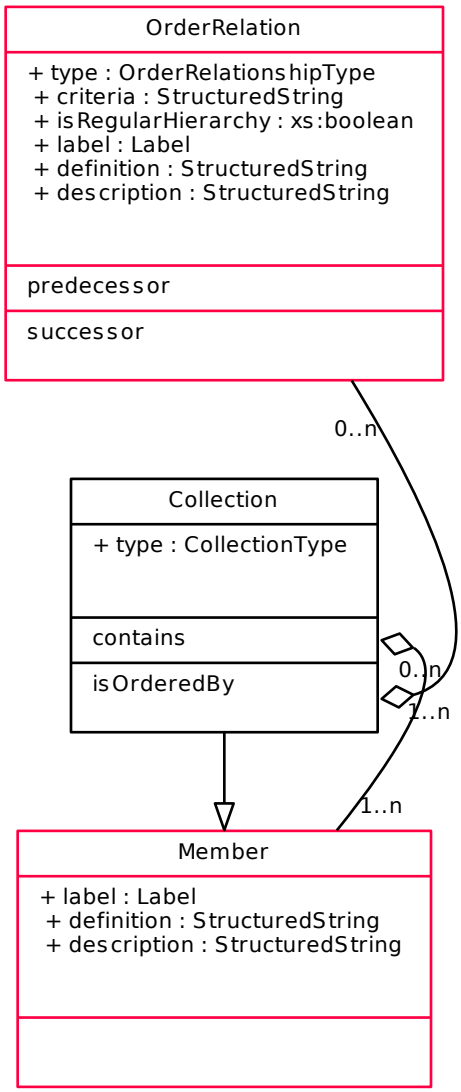
Properties

Name	Type	Cardinality
type	CollectionType	0..1

type

Whether the collection is a bag or a set.

Graph



5.12.2 CollectionCorrespondence

Generic (untyped) relationship between collections.

Extends

AnnotatedIdentifiable

Properties

Name	Type	Cardinality
label	Label	0..n
definition	StructuredString	0..1
description	StructuredString	0..1

label

A display label for the CollectionCorrespondence. May be expressed in multiple languages. Repeat for labels with different content, for example, labels with differing length limitations.

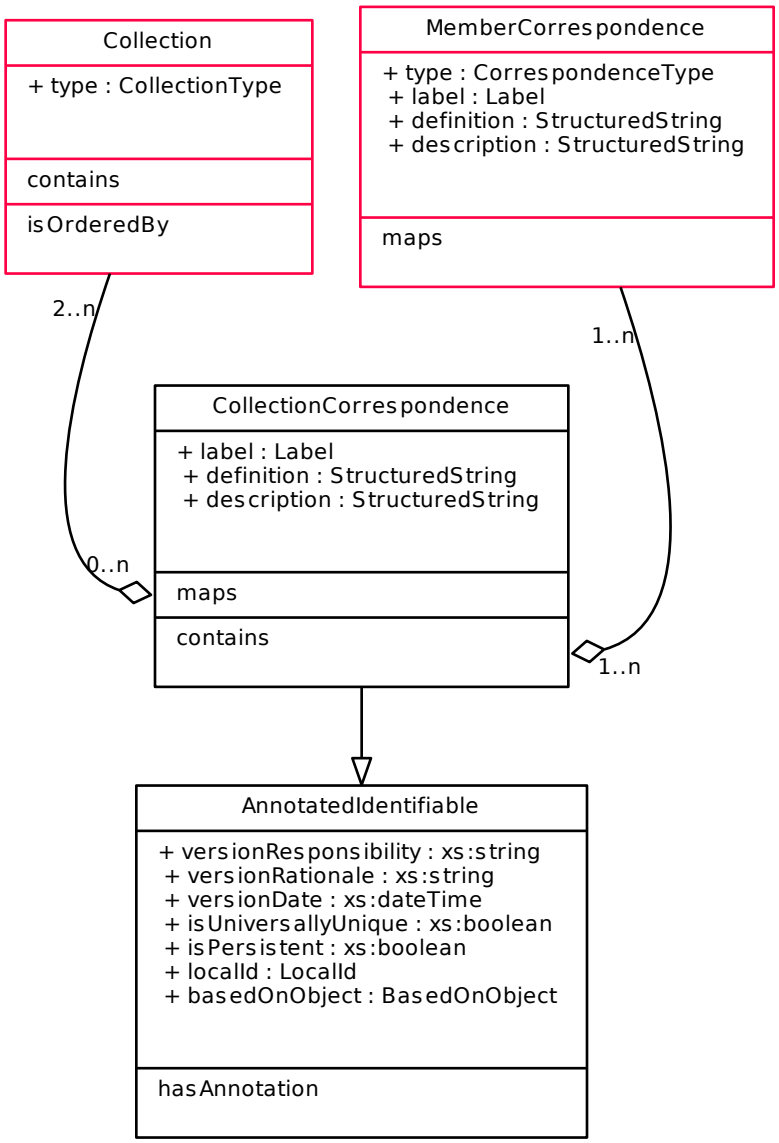
definition

A definition of the CollectionCorrespondence. May be expressed in multiple languages and supports the use of structured content.

description

A description of the purpose or use of the CollectionCorrespondence. May be expressed in multiple languages and supports the use of structured content.

Graph



5.12.3 Member

Generic class representing members of a collection.

Extends

AnnotatedIdentifiable

Properties

Name	Type	Cardinality
label	Label	0..n
definition	StructuredString	0..1
description	StructuredString	0..1

label

A display label for the Member. May be expressed in multiple languages. Repeat for labels with different content, for example, labels with differing length limitations.

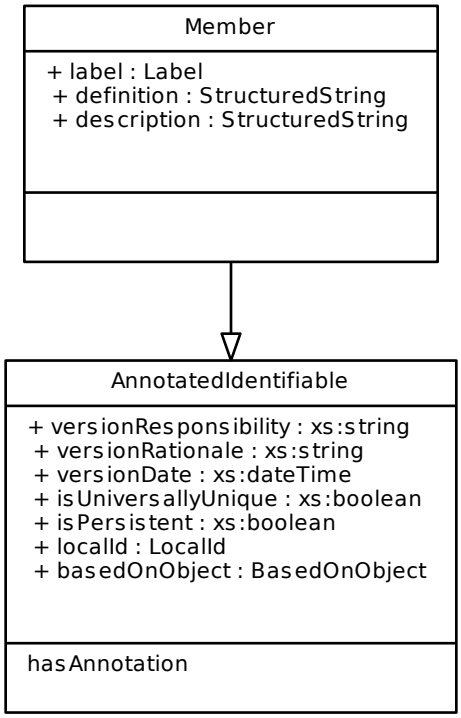
definition

A definition of the Member. May be expressed in multiple languages and supports the use of structured content.

description

A description of the purpose or use of the Member. May be expressed in multiple languages and supports the use of structured content.

Graph



5.12.4 MemberCorrespondence

Generic (untyped) relationship between members of collections.

Extends

AnnotatedIdentifiable

Properties

Name	Type	Cardinality
type	CorrespondenceType	0..1
label	Label	0..n
definition	StructuredString	0..1
description	StructuredString	0..1

type

Type of correspondence in terms of commonalities and differences between two members.

label

A display label for the MemberCorrespondence. May be expressed in multiple languages. Repeat for labels with different content, for example, labels with differing length limitations.

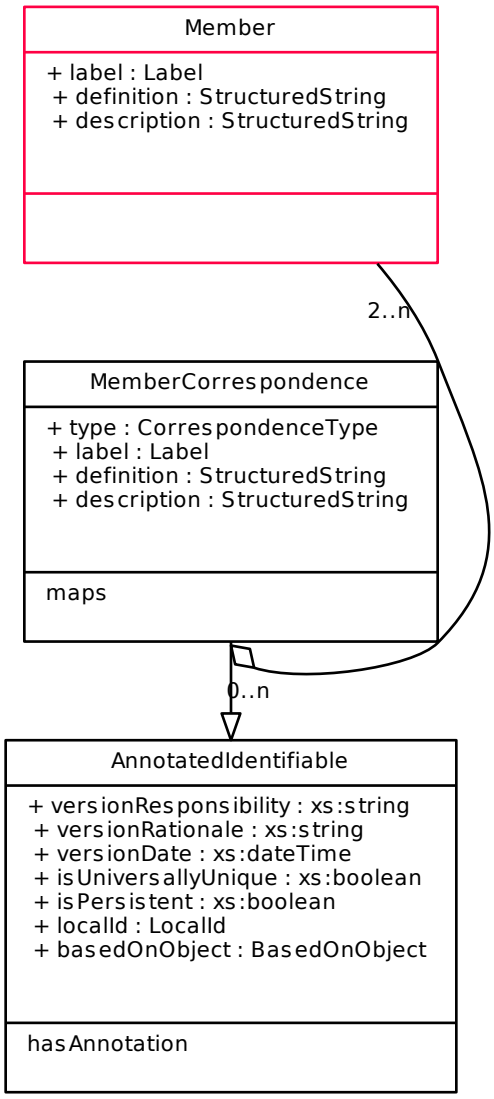
definition

A definition of the MemberCorrespondence. May be expressed in multiple languages and supports the use of structured content.

description

A description of the purpose or use of the MemberCorrespondence. May be expressed in multiple languages and supports the use of structured content.

Graph



5.12.5 OrderRelation

Binary relation over members in a collection (set or bag) that is always reflexive, antisymmetric, and transitive. It can also be either total or partial. It must contain like items.

Extends

AnnotatedIdentifiable

Properties

Name	Type	Cardinality
type	OrderRelationshipType	0..1
criteria	StructuredString	0..1
isRegularHierarchy	xs:boolean	0..1
label	Label	0..n
definition	StructuredString	0..1
description	StructuredString	0..1

type

Whether the order relation is total or partial.

criteria

Intensional definition of the order criteria (e.g. alphabetical, numerical, increasing, decreasing, etc.)

isRegularHierarchy

Indicates whether the tree defined by the order relation is regular or not. i.e., all leaves are at the same level..

label

A display label for the OrderRelation. May be expressed in multiple languages. Repeat for labels with different content, for example, labels with differing length limitations.

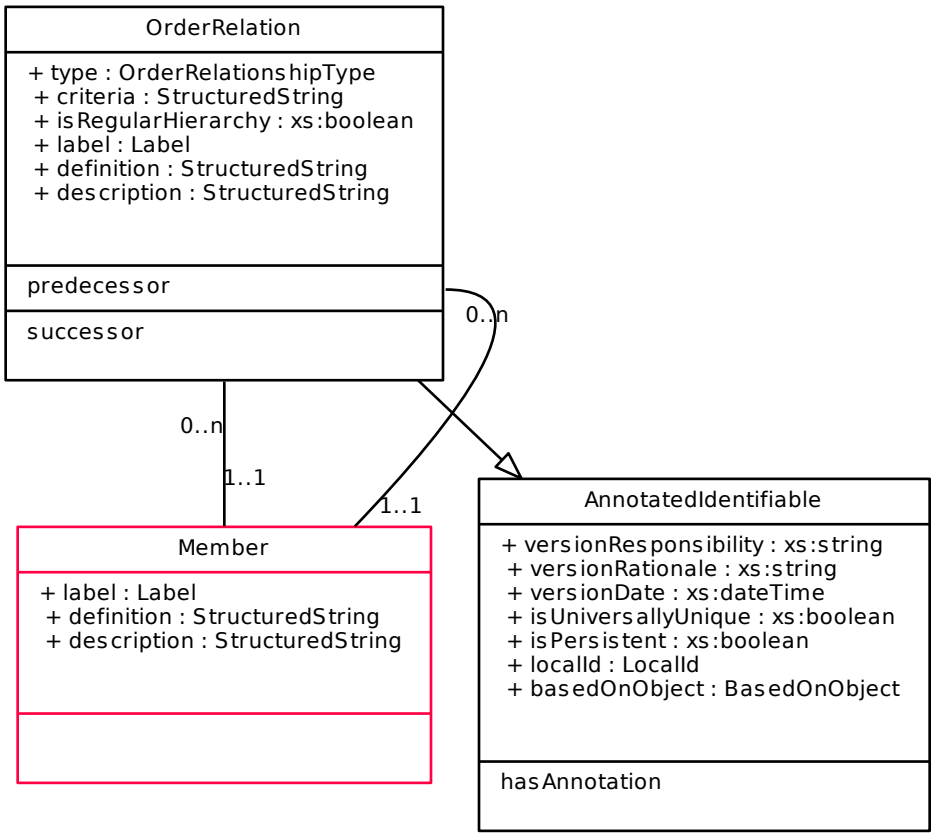
definition

A definition of the OrderRelation. May be expressed in multiple languages and supports the use of structured content.

description

A description of the purpose or use of the OrderRelation. May be expressed in multiple languages and supports the use of structured content.

Graph



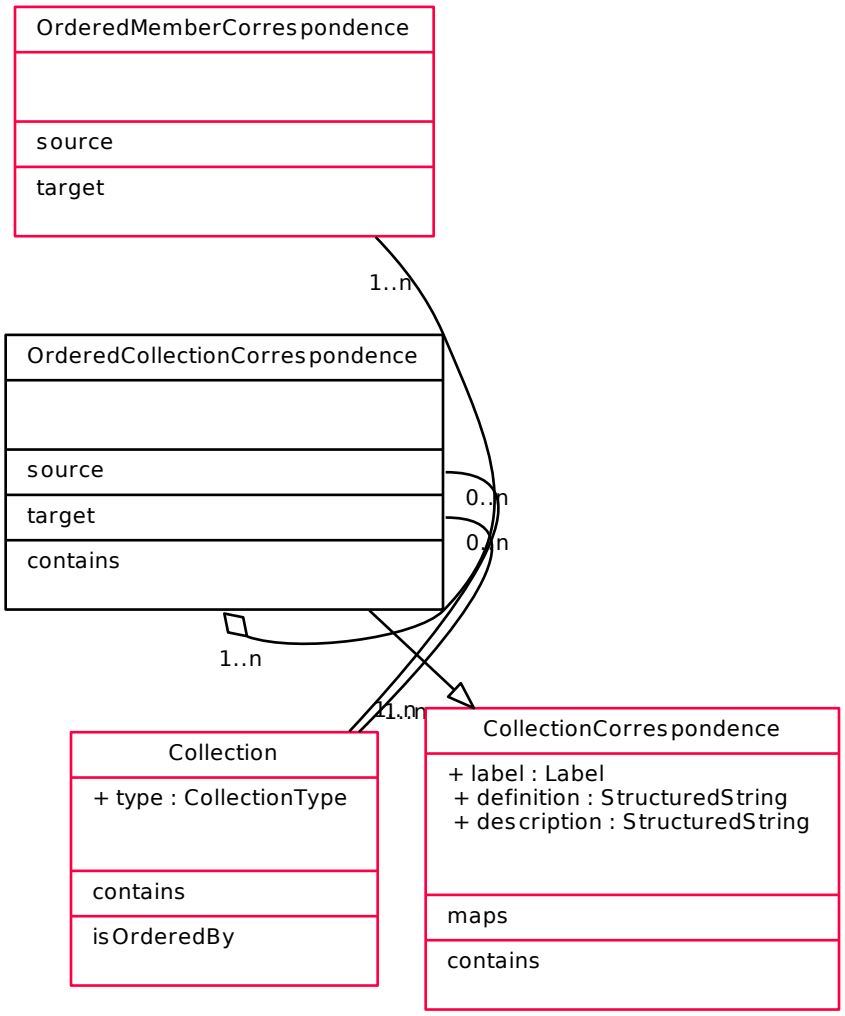
5.12.6 OrderedCollectionCorrespondence

Generic (untyped) ordered relationship between collections.

Extends

CollectionCorrespondence

Graph



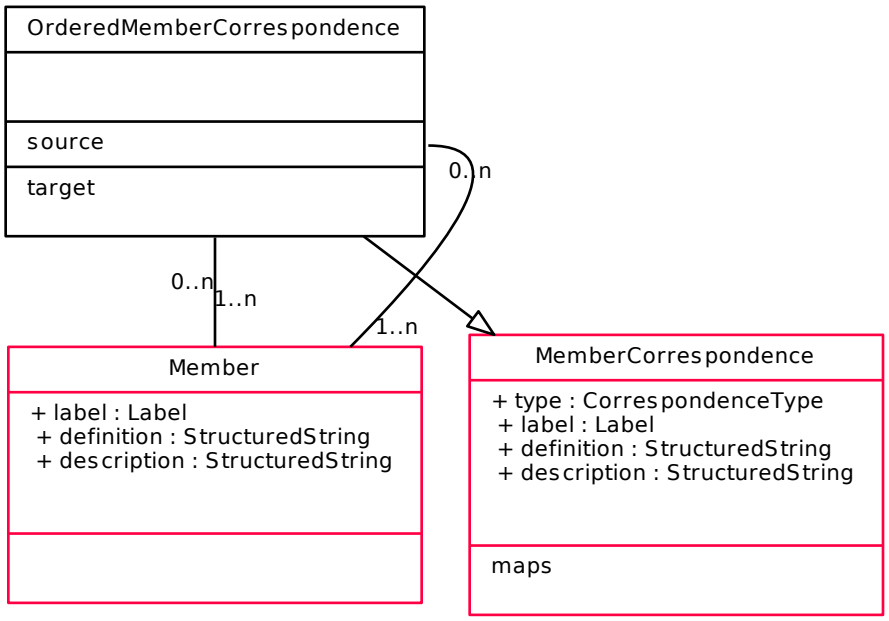
5.12.7 OrderedMemberCorrespondence

Ordered relationship between members of collections.

Extends

MemberCorrespondence

Graph



5.13 BaseObjects

The package is a holding place for objects that are widely used but are not treated as properties; primarily identifiable types. A decision on the status and location of these objects needs to be made. Contents

5.14 ComplexDataTypes

Extensions of base type Primitives Contents

5.14.1 AccessLocation

A set of access information for a Machine including external and internal URL, mime type, and physical location

Properties

Name	Type	Cardinality
externalURLReference	URL	0..n
internalURLReference	xs:anyURI	0..1
mimeType	CodeValueType	0..1
physicalLocation	InternationalString	0..n

externalURLReference

An external URL

internalURLReference

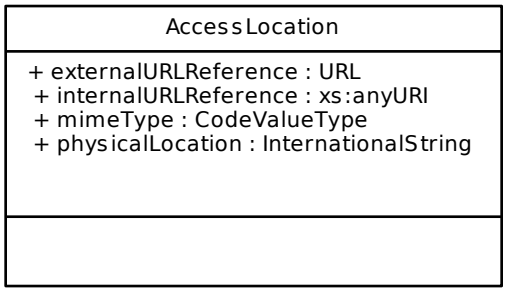
The internal URL.

mimeType

physicalLocation

The physical location of the machine

Graph



5.14.2 Address

Location address identifying each part of the address as separate elements, identifying the type of address, the level of privacy associated with the release of the address, and a flag to identify the preferred address for contact.

Properties

Name	Type	Cardinality
typeOfAddress	CodeValueType	0..1
line	xs:string	0..n
cityPlaceLocal	xs:string	0..1
stateProvince	xs:string	0..1
postalCode	xs:string	0..1
countryCode	CountryCode	0..1
timeZone	CodeValueType	0..1
effectivePeriod	xs:date	0..1
privacy	CodeValueType	0..1
isPreferred	xs:boolean	0..1
geographicPoint	Point	0..1
regionalCoverage	CodeValueType	0..1
typeOfLocation	CodeValueType	0..1
locationName	Name	0..1

typeOfAddress

Indicates address type (i.e. home, office, mailing, etc.)

line

Number and street including office or suite number. May use multiple lines.

cityPlaceLocal

City, place, or local area used as part of an address.

stateProvince

A major subnational division such as a state or province used to identify a major region within an address.

postalCode

Postal or ZIP Code

countryCode

Country of the location

timeZone

Time zone of the location expressed as code.

effectivePeriod

Clarifies when the identification information is accurate.

privacy

Specify the level privacy for the address as public, restricted, or private. Supports the use of an external controlled vocabulary

isPreferred

Set to “true” if this is the preferred location for contacting the organization or individual.

geographicPoint

Geographic coordinates corresponding to the address.

regionalCoverage

The region covered by the agent at this address

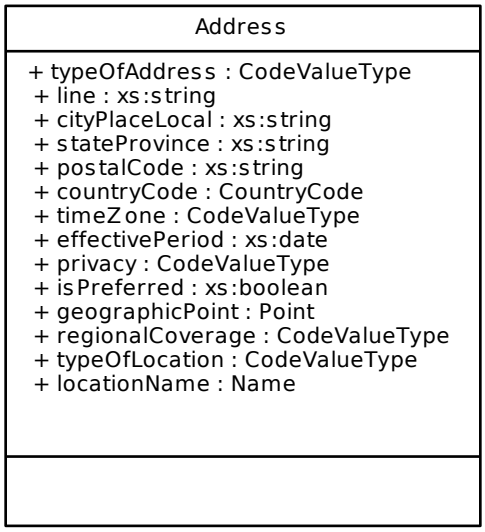
typeOfLocation

The type or purpose of the location (i.e. regional office, distribution center, home)

locationName

Name of the location if applicable.

Graph



5.14.3 AgentAssociation

A basic structure for declaring the name of an Agent inline, reference to an Agent, and role specification. This object is used primarily within Annotation.

Properties

Name	Type	Cardinality
agent	BibliographicName	0..1
role	PairedCodeValueType	0..n

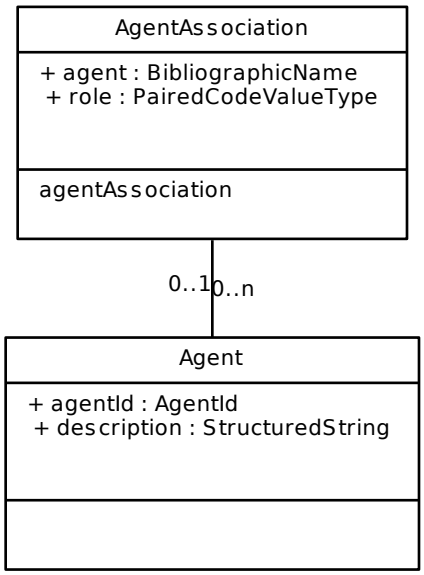
agent

Full name of the contributor. Language equivalents should be expressed within the International String structure.

role

The role of the of the Agent within the context of the parent property name with information on the extent to which the role applies. Allows for use of external controlled vocabularies. Reference should be made to the vocabulary within the structure of the role.

Graph



5.14.4 AgentId

Persistent identifier for a researcher using a system like ORCID

Properties

Name	Type	Cardinality
agentIdValue	xs:string	1..1
agentIdType	xs:string	1..1

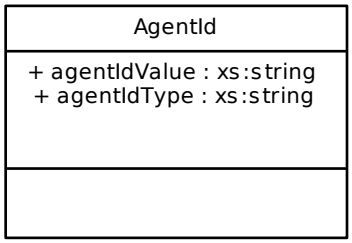
agentIdValue

The identifier for the agent.

agentIdType

The identifier system in use.

Graph



5.14.5 AnnotationDate

A generic date type for use in Annotation which provides that standard date structure plus a property to define the date type (Publication date, Accepted date, Copyrighted date, Submitted date, etc.). Equivilent of http://purl.org/dc/elements/1.1/date where the type of date may identify the Dublin Core refinement term.

Extends

Date

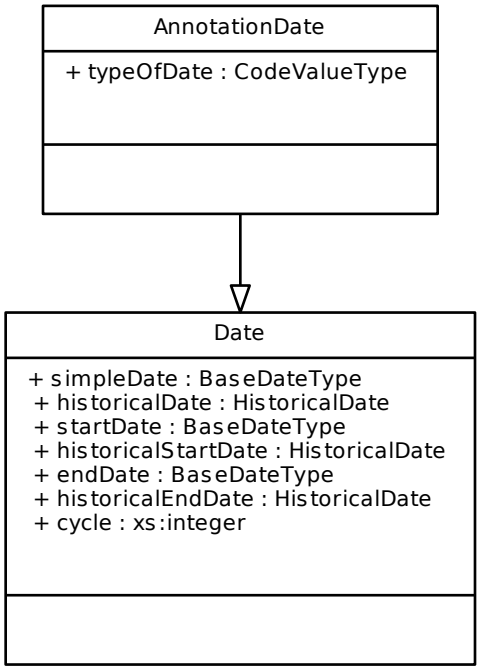
Properties

Name	Type	Cardinality
typeOfDate	CodeValueType	0..n

typeOfDate

Use to specify the type of date. This may reflect the refinements of dc:date such as dateAccepted, dateCopyrighted, dateSubmitted, etc.

Graph



5.14.6 AudioSegment

Describes the type and length of the audio segment.

Properties

Name	Type	Cardinality
typeOfAudioClip	CodeValueType	0..1
audioClipBegin	xs:string	0..1
audioClipEnd	xs:string	0..1

typeOfAudioClip

The type of audio clip provided. Supports the use of a controlled vocabulary.

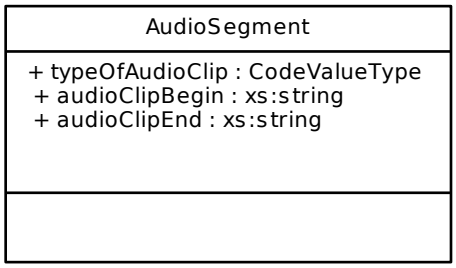
audioClipBegin

The point to begin the audio clip. If no point is provided the assumption is that the start point is the beginning of the clip provided.

audioClipEnd

The point to end the audio clip. If no point is provided the assumption is that the end point is the end of the clip provided.

Graph



5.14.7 BasedOnObject

Use when creating an object that is based on an existing object or objects that are managed by a different agency or when the new object is NOT simply a version change but you wish to maintain a reference to the object that served as a basis for the new object. BasedOnObject may contain references to any number of objects which serve as a basis for this object, a BasedOnRationaleDescription of how the content of the referenced object was incorporated or altered, and a BasedOnRationaleCode to allow for specific typing of the BasedOnReference according to an external controlled vocabulary.

Properties

Name	Type	Cardinality
basedOnRationaleDescription	InternationalString	0..1
basedOnRationaleCode	CodeValueType	0..1

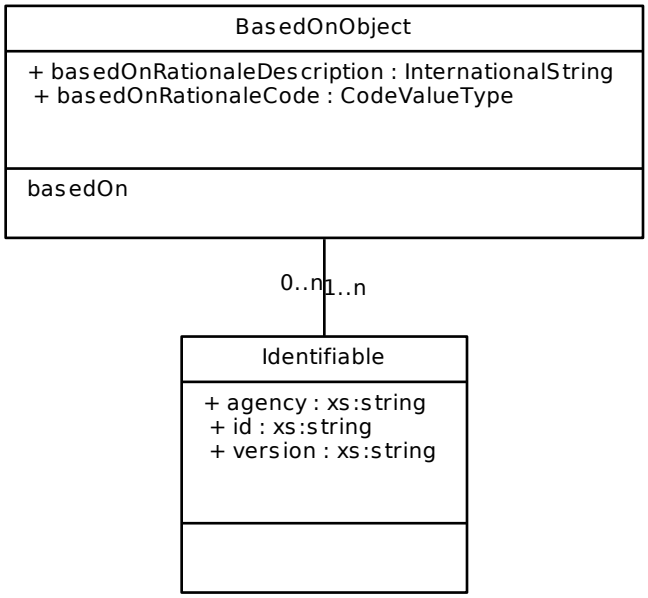
basedOnRationaleDescription

Textual description of the rationale/purpose for the based on reference to inform users as to the extent and implication of the version change. May be expressed in multiple languages.

basedOnRationaleCode

RationaleCode is primarily for internal processing flags within an organization or system. Supports the use of an external controlled vocabulary.

Graph



5.14.8 BibliographicName

Personal names should be listed surname or family name first, followed by forename or given name. When in doubt, give the name as it appears, and do not invert. In the case of organizations where there is clearly a hierarchy present, list the parts of the hierarchy from largest to smallest, separated by full stops and a space. If it is not clear whether there is a hierarchy present, or unclear which is the larger or smaller portion of the body, give the name as it appears in the item. The name may be provided in one or more languages.

Extends

InternationalString

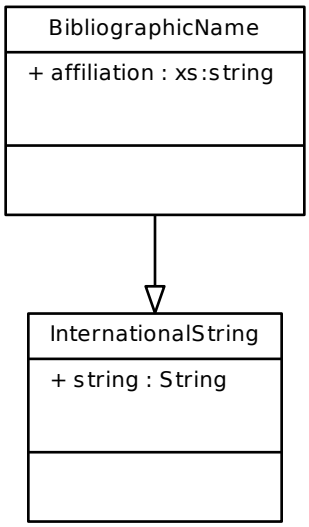
Properties

Name	Type	Cardinality
affiliation	xs:string	0..1

affiliation

The affiliation of this person to an organization. This is generally an organization or sub-organization name and should be related to the specific role within which the individual is being listed.

Graph



5.14.9 CharacterOffset

Specification of the character offset for the beginning and end of the segment.

Properties

Name	Type	Cardinality
startCharOffset	xs:integer	0..1
endCharOffset	xs:integer	0..1

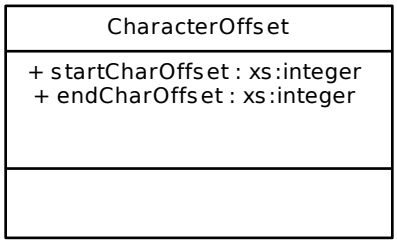
startCharOffset

Number of characters from beginning of the document, indicating the inclusive start of the text range.

endCharOffset

Number of characters from the beginning of the document, indicating the inclusive end of the text segment.

Graph



5.14.10 CodeValueType

Allows for string content which may be taken from an externally maintained controlled vocabulary (code value). If the content is from a controlled vocabulary provide the code value, as well as a reference to the code list from which the value is taken. Provide as many of the identifying attributes as needed to adequately identify the controlled vocabulary. Note that DDI has published a number of controlled vocabularies applicable to several locations using the CodeValue structure. Use of shared controlled vocabularies helps support interoperability and machine actionability.

Properties

Name	Type	Cardinality
codeValue	xs:string	1..1
codeListID	xs:string	0..1
codeListName	xs:string	0..1
codeListAgencyName	xs:string	0..1
codeListVersionID	xs:string	0..1
otherValue	xs:string	0..1
codeListURN	xs:string	0..1
codeListSchemeURN	xs:string	0..1

codeValue

The actual value.

codeListID

The ID of the code list (controlled vocabulary) that the content was taken from.

codeListName

The name of the code list.

codeListAgencyName

The name of the agency maintaining the code list.

codeListVersionID

The version number of the code list (default is 1.0).

otherValue

If the value of the string is "Other" or the equivalent from the codelist, this attribute can provide a more specific value not found in the codelist.

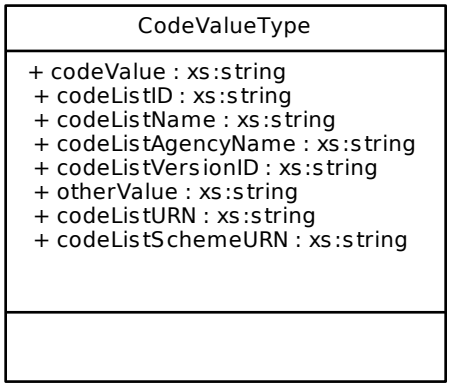
codeListURN

The URN of the codelist.

codeListSchemeURN

If maintained within a scheme, the URN of the scheme containing the codelist.

Graph



5.14.11 CommandCode

Contains information on the command used for processing data. Contains a description of the command which should clarify for the user the purpose and process of the command, an in-line provision of the command itself, a reference to an external version of the command such as a coding script, and the option for attaching an extension to DDI to permit

insertion of a command code in a foreign namespace. The definition of the InParameter, OutParameter, and Binding declared within CommandCode are available for use by all formats of the command.

Properties

Name	Type	Cardinality
description	StructuredString	0..1
binding	Binding	0..n

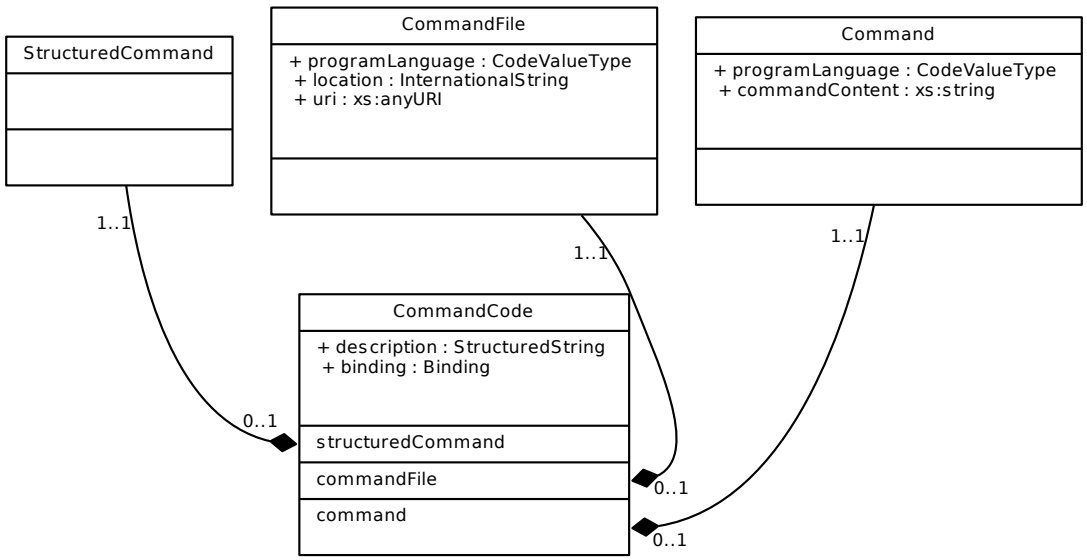
description

A description of the purpose and use of the command code provided. Supports multiple languages.

binding

Defines the link between the OutParameter of an external object to an InParameter of this CommandCode.

Graph



5.14.12 ConditionalText

Text which has a changeable value depending on a stated condition, response to earlier questions, or as input from a set of metrics (pre-supplied data).

Extends

TextContent

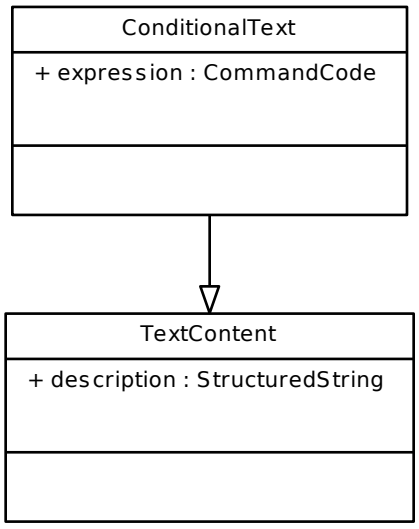
Properties

Name	Type	Cardinality
expression	CommandCode	0..1

expression

The condition on which the associated text varies expressed by a command code. For example, a command that inserts an age by calculating the difference between today's date and a previously defined date of birth.

Graph



5.14.13 ContactInformation

Contact information for the individual or organization including location specification, address, URL, phone numbers, and other means of communication access. Address, location, telephone, and other means of communication can be repeated to express multiple means of a single type or change over time. Each major piece of contact information (with the exception of URL) contains the element EffectiveDates in order to date stamp the period for which the information is valid.

Properties

Name	Type	Cardinality
website	URL	0..n
email	Email	0..n
electronicMessaging	ElectronicMessageSystem	0..n
address	Address	0..n
telephone	Telephone	0..n

website

The URL of the Agent's website

email

Email contact information

electronicMessaging

Electronic messaging other than email

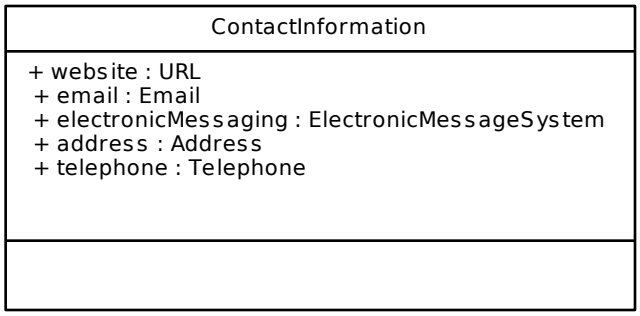
address

The address for contact.

telephone

Telephone for contact

Graph



5.14.14 Content

Supports the optional use of XHTML formatting tags within the string structure. XHTML tag content is controlled by the schema, see http://www.w3.org/1999/xhtml/ for a detailed list of available tags. Language of the string is defined by the attribute xmlang. The content can be identified as translated (isTranslated), subject to translation (isTranslatable), the result of translation from one or more languages (translationSourceLanguages), and carry an indication whether or not it should be treated as plain text (isPlain).

Properties

Name	Type	Cardinality
content	xhtml:BlkNoForm.mix	1..n
xmlLang	xs:language	0..1
isTranslated	xs:boolean	0..1
isTranslatable	xs:boolean	0..1
translationSourceLanguage	xs:language	0..n
translationDate	xs:date	0..1
isPlainText	xs:boolean	0..1

content

The following xhtml tags are available for use in Content: address, blockquote, pre, h1, h2, h3, h4, h5, h6, hr, div, p, a, abbr, acronym, cite, code, dfn, em, kbd, q, samp, strong, var, b, big, i, small, sub, sup, tt, br, span, dl, dt, dd, ol, ul, li, table, caption, thead, tfoot, tbody, colgroup, col, tr, th, and td. They should be used with the xhtml namespace prefix, i.e., xhtmdiv. See DDI Technical Manual Part I for additional details.

xmlLang

Indicates the language of content.

isTranslated

Indicates whether content is a translation (true) or an original (false).

isTranslatable

Indicates whether content is translatable (true) or not (false).

translationSourceLanguage

List the language or language codes in a space delimited array. The language original may or may not be provided in this bundle of language specific strings.

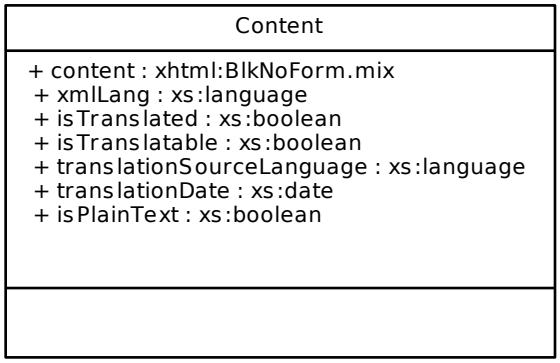
translationDate

The date the content was translated. Provision of translation date allows user to verify if translation was available during data collection or other time linked activity.

isPlainText

Indicates that the content is to be treated as plain text (no formatting). You may use DDIProfile to fix the value of this attribute to 'true' in cases where you wish to indicate that your system treats all content should be treated as plain text.

Graph



5.14.15 ContentDateOffset

Identifies the difference between the date applied to the data as a whole and this specific item such as previous year's income or residence 5 years ago. A value of true for the attribute isNegativeOffset indicates that the offset is the specified number of declared units prior to the date of the data as a whole and false indicates information regarding a future state.

Extends

CodeValueType

Properties

Name	Type	Cardinality
numberOfUnits	xs:decimal	0..1
isNegativeOffset	xs:boolean	0..1

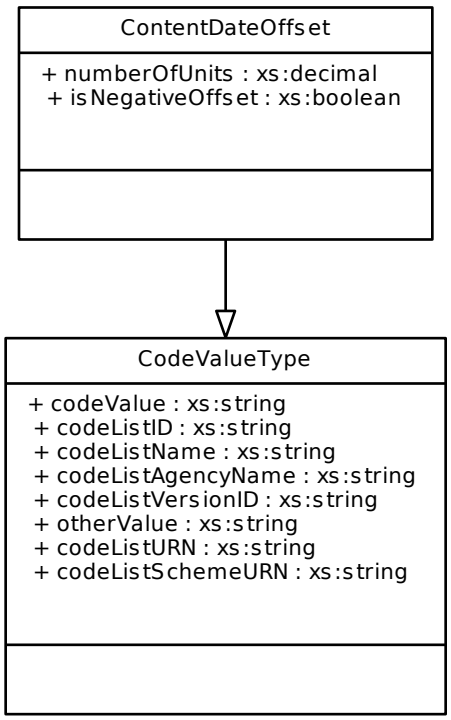
numberOfUnits

The number of units to off-set the date for this item expressed as a decimal.

isNegativeOffset

If set to "true" the date is offset the number of units specified PRIOR to the default date of the data. A setting of "false" indicates a date the specified number of units in the FUTURE from the default date of the data.

Graph



5.14.16 CorrespondenceType

Describes the commonalities and differences between two items using a textual description of both commonalities and differences plus an optional coding of the type of commonality.

Properties

Name	Type	Cardinality
commonality	StructuredString	0..1
difference	StructuredString	0..1
commonalityTypeCode	CodeValueType	0..n

commonality

A description of the common features of the two items using a StructuredString to support multiple language versions of the same content as well as optional formatting of the content.

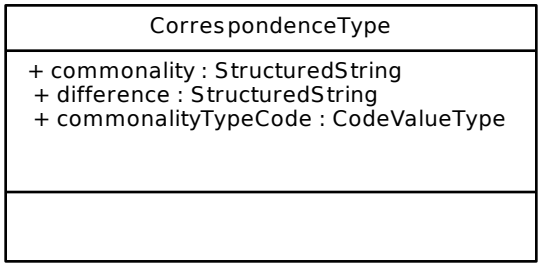
difference

A description of the differences between the two items using a StructuredString to support multiple language versions of the same content as well as optional formatting of the content.

commonalityTypeCode

Commonality expressed as a term or code. Supports the use of an external controlled vocabulary. If repeated, clarify each external controlled vocabulary used.

Graph



5.14.17 CountryCode

Provides means of expressing a code/term for the country plus an optional valid date.

Properties

Name	Type	Cardinality
effectiveDate	xs:dateTime	0..1
country	InternationalCodeValueType	1..1

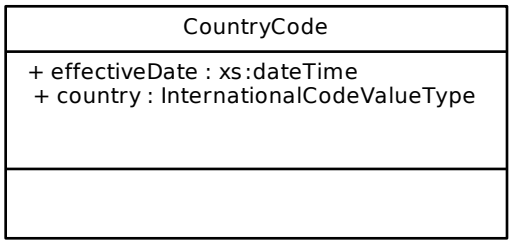
effectiveDate

If it is important to specify the date that this code is effective in order to accurately capture a name or similar change, specify that here.

country

The code or term used to designate the country. If a term, indicate the language.

Graph



5.14.18 Date

Provides the structure of a Date element, which allows a choice between single, simple dates (of BaseDateType) or date ranges. If the Date element contains a range, Cycle may be used to indicate occurrence of this range within a series of ranges as an integer identifying the cycle, i.e. the 4th wave of a data collection cycle would have. BaseDateType allows for different date time combinations to provide a simple and convenient mechanism to specify different date and time values with a machine actionable format specified by regular expressions.

Properties

Name	Type	Cardinality
simpleDate	BaseDateType	0..1
historicalDate	HistoricalDate	0..1
startDate	BaseDateType	0..1
historicalStartDate	HistoricalDate	0..1
endDate	BaseDateType	0..1
historicalEndDate	HistoricalDate	0..1
cycle	xs:integer	0..1

simpleDate

A single point in time. If a duration is expressed as a SimpleDate it is defining a period of time without a designated Start or End date.

historicalDate

A simple date expressed in a historical date format, including a specification of the date format and calendar used.

startDate

Start of a date range. If there is a start date with no end date provided, this implies that the end date is unknown but that the date being recorded is not just a simple date but a range of unknown duration.

historicalStartDate

A start date expressed in a historical date format, including a specification of the date format and calendar used.

endDate

End of a date range which may or may not have a known start date.

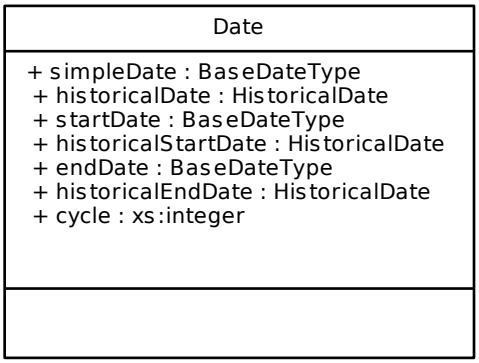
historicalEndDate

An end date expressed in a historical date format, including a specification of the date format and calendar used.

cycle

Use to indicate occurrence of this range within a series of ranges as an integer identifying the cycle, i.e. the 4th wave of a data collection cycle would have

Graph



5.14.19 DynamicText

Structure supporting the use of dynamic text, where portions of the textual content change depending on external information (pre-loaded data, response to an earlier query, environmental situations, etc.).

Properties

Name	Type	Cardinality
content	TextContent	1..n
isStructureRequired	xs:boolean	0..1
audienceLanguage	xs:language	0..1

content

This is the head of a substitution group and is never used directly as an element name. Instead it is replaced with either LiteralText or ConditionalText.

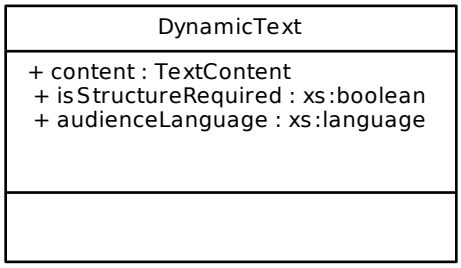
isStructureRequired

If textual structure (e.g. size, color, font, etc.) is required to understand the meaning of the content change value to "true".

audienceLanguage

Specifies the language of the intended audience. This is particularly important for clarifying the primary language of a mixed language textual string, for example when language testing and using a foreign word withing the question text.

Graph



5.14.20 ElectronicMessageSystem

Any non-email means of relaying a message electronically. This would include text messaging, Skype, Twitter, ICQ, or other emerging means of electronic message conveyance.

Properties

Name	Type	Cardinality
contactAddress	xs:string	0..1
typeOfService	CodeValueType	0..1
effectivePeriod	Date	0..1
privacy	CodeValueType	0..1
isPreferred	xs:boolean	0..1

contactAddress

Account identification for contacting

typeOfService

Indicates the type of service used. Supports the use of a controlled vocabulary.

effectivePeriod

Time period during which the account is valid.

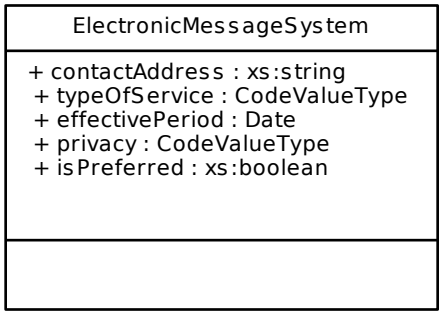
privacy

Specify the level privacy for the address as public, restricted, or private. Supports the use of an external controlled vocabulary.

isPreferred

Set to "true" if this is the preferred address.

Graph



5.14.21 Email

An e-mail address which conforms to the internet format (RFC 822) including its type and time period for which it is valid.

Properties

Name	Type	Cardinality
internetEmail	xs:string	0..1
typeOfEmail	CodeValueType	0..1
effectivePeriod	Date	0..1
privacy	CodeValueType	0..1
isPreferred	xs:boolean	0..1

internetEmail

The email address expressed as a string (should follow the Internet format specification - RFC 5322) e.g. user@server.ext, more complex and flexible examples are also supported by the format.

typeOfEmail

Code indicating the type of e-mail address. Supports the use of an external controlled vocabulary. (e.g. home, office)

effectivePeriod

Time period for which the e-mail address is valid.

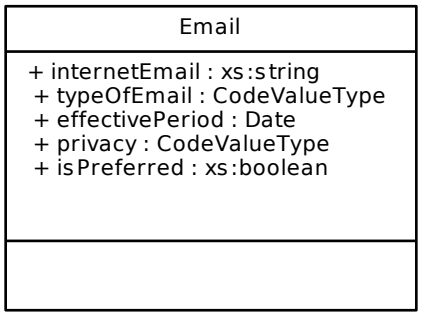
privacy

Indicates the level of privacy

isPreferred

Set to true if this is the preferred email

Graph



5.14.22 Form

A link to a form used by the metadata containing the form number, a statement regarding the contents of the form, a statement as to the mandatory nature of the form and a privacy level designation.

Properties

Name	Type	Cardinality
formNumber	xs:string	0..1
uri	xs:anyURI	0..1
statement	InternationalString	0..1
isRequired	xs:boolean	0..1

formNumber

The number or other means of identifying the form.

uri

The URN or URL of the form.

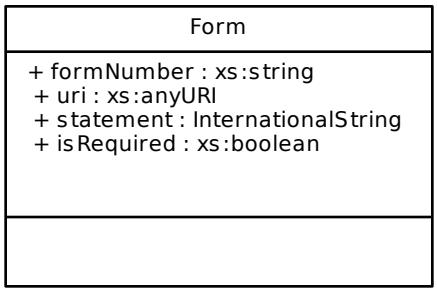
statement

A statement regarding the use, coverage, and purpose of the form.

isRequired

Set to “true” if the form is required. Set to “false” if the form is optional.

Graph



5.14.23 HistoricalDate

Used to preserve an historical date, formatted in a non-ISO fashion.

Properties

Name	Type	Cardinality
nonISODate	xs:string	1..1
historicalDateFormat	CodeValueType	0..1
calendar	CodeValueType	0..1

nonISODate

This is the date expressed in a non-ISO compliant structure. Primarily used to retain legacy content or to express non-Gregorian calender dates.

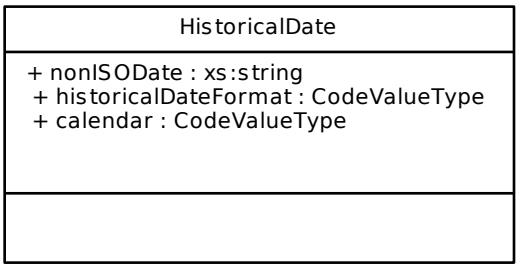
historicalDateFormat

Indicate the structure of the date provided in NonISODate. For example if the NonISODate contained 4/1/2000 the Historical Date Format would be mm/dd/yyyy. The use of a controlled vocabulary is strongly recommended to support interoperability.

calendar

Specifies the type of calendar used (e.g., Gregorian, Julian, Jewish).

Graph



5.14.24 Image

A reference to an image, with a description of its properties and type.

Properties

Name	Type	Cardinality
imageLocation	xs:anyURI	0..1
typeOfImage	CodeValueType	0..1
dpi	xs:integer	0..1
languageOfImage	xs:language	0..1

imageLocation

A reference to the location of the image using a URI.

typeOfImage

Brief description of the image type. Supports the use of an external controlled vocabulary.

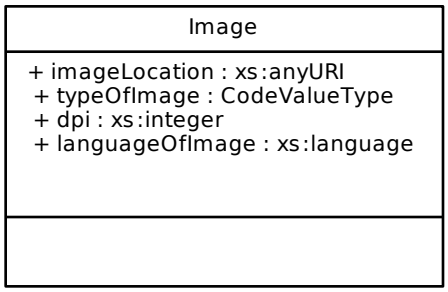
dpi

Provides the resolution of the image in dots per inch to assist in selecting the appropriate image for various uses.

languageOfImage

Language of image.

Graph



5.14.25 ImageArea

Defines the shape and area of an image used as part of a location representation. The shape is defined as a Rectangle, Circle, or Polygon and Coordinates provides the information required to define it.

Properties

Name	Type	Cardinality
coordinates	xs:string	0..1
shape	ShapeCoded	1..1

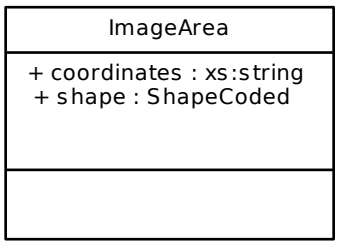
coordinates

A comma-delimited list of x,y coordinates, listed as a set of adjacent points for rectangles and polygons, and as a center-point and a radius for circles (x,y,r).

shape

A fixed set of valid responses includes Rectangle, Circle, and Polygon.

Graph



5.14.26 IndividualName

The name of an individual broken out into its component parts of prefix, first/given name, middle name, last/family/surname, and suffix. The preferred compilation of the name parts may also be provided. The legal or formal name of the individual should have the isFormal attribute set to true. The preferred name should be noted with the isPreferred attribute. The attribute sex provides information to assist in the appropriate use of pronouns.

Properties

Name	Type	Cardinality
prefix	xs:string	0..1
firstGiven	xs:string	0..1
middle	xs:string	0..n
lastFamily	xs:string	0..1
suffix	xs:string	0..1
fullName	InternationalString	0..1
effectivePeriod	Date	0..1
abbreviation	InternationalString	0..1
typeOfIndividualName	CodeValueType	0..1
sex	SexSpecificationType	0..1
isPreferred	xs:boolean	0..1
context	xs:string	0..1
isFormal	xs:boolean	0..1

prefix

Title that precedes the name of the individual, such as Ms., or Dr.

firstGiven

First (given) name of the individual

middle

Middle name or initial of the individual

lastFamily

Last (family) name /surname of the individual

suffix

Title that follows the name of the individual, such as Esq.

fullName

This provides a means of providing a full name as a single object for display or print such as identification badges etc. For example a person with the name of William Grace for official use may prefer a display name of Bill Grace on a name tag or other informal publication.

effectivePeriod

Clarifies when the name information is accurate.

abbreviation

An abbreviation or acronym for the name. This may be expressed in multiple languages. It is assumed that if only a single language is provided that it may be used in any of the other languages within which the name itself is expressed.

typeOfIndividualName

The type of individual name provided. the use of a controlled vocabulary is strongly recommended. At minimum his should include, e.g. PreviousFormalName, Nickname (or CommonName), Other.

sex

Sex allows for the specification of male, female or neutral. The purpose of providing this information is to assist others in the appropriate use of pronouns when addressing the individual. Note that many countries/languages may offer a neutral pronoun form.

isPreferred

If more than one name for the object is provided, use the isPreferred attribute to indicate which is the preferred name content. All other names should be set to isPreferred=”false”.

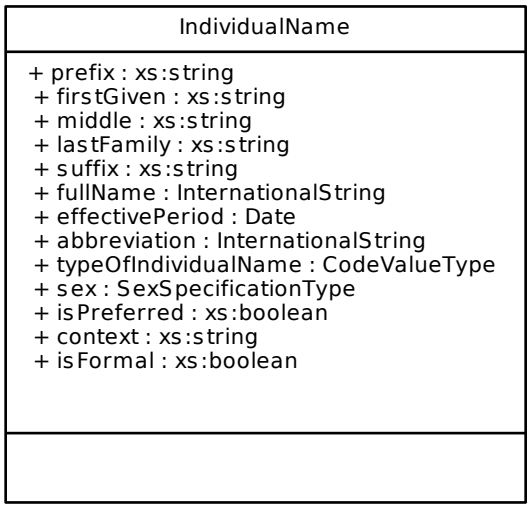
context

A name may be specific to a particular context, i.e. common usage, business, social, etc.. Identify the context related to the specified name.

isFormal

The legal or formal name of the individual should have the isFormal attribute set to true. To avoid confusion only one individual name should have the isFormal attribute set to true. Use the TypeOfIndividualName to further differentiate the type and applied usage when multiple names are provided.

Graph



5.14.27 InternationalCodeValueType

Allows for string content which may be taken from an externally maintained controlled vocabulary (code value). If the content is from a controlled vocabulary provide the code value, as well as a reference to the code list from which the value is taken. This differs from a CodeValue only by the provision of a language-location specification. DDI uses the International CodeValue in cases where controlled vocabularies are assumed to be highly language specific, such as nationally maintained subject headings, thesauri, or keywords derived from the content of documents. The ability to identify language is especially important when supporting searches by external language-specific search engines. Provide as many of the identifying attributes as needed to adequately identify the controlled vocabulary.

Extends

String

Properties

Name	Type	Cardinality
codeListID	xs:string	0..1
codeListName	xs:string	0..1
codeListAgencyName	xs:string	0..1
codeListVersionID	xs:string	0..1
otherValue	xs:string	0..1
codeListURN	xs:string	0..1
codeListSchemeURN	xs:string	0..1

codeListID

The ID of the code list (controlled vocabulary) that the content was taken from.

codeListName

The name of the code list.

codeListAgencyName

The name of the agency maintaining the code list.

codeListVersionID

The version number of the code list (default is 1.0).

otherValue

If the value of the string is "Other" or the equivalent from the codelist, this attribute can provide a more specific value not found in the codelist.

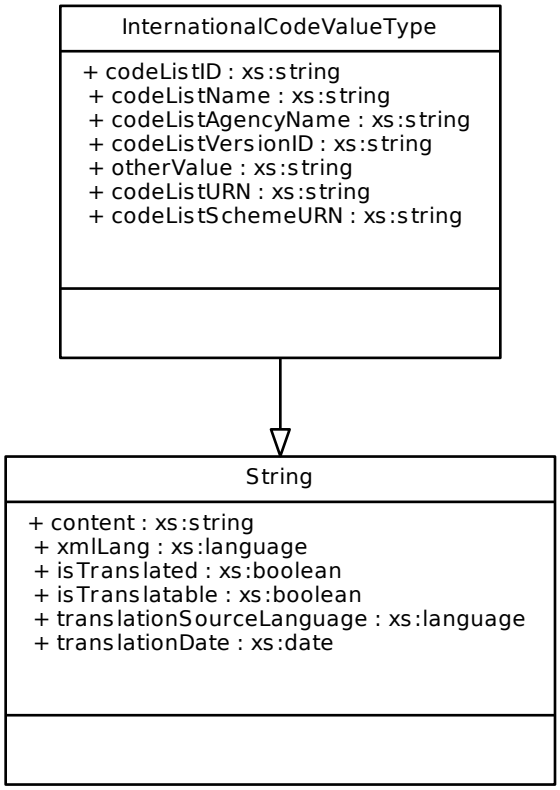
codeListURN

The URN of the codelist.

codeListSchemeURN

If maintained within a scheme, the URN of the scheme containing the codelist.

Graph



5.14.28 InternationalIdentifier

An identifier whose scope of uniqueness is broader than the local archive. Common forms of an international identifier are ISBN, ISSN, DOI or similar designator. Provides both the value of the identifier and the agency who manages it.

Properties

Name	Type	Cardinality
identifierContent	xs:string	0..1
managingAgency	CodeValueType	0..1
isURI	xs:boolean	0..1

identifierContent

An identifier as it should be listed for identification purposes.

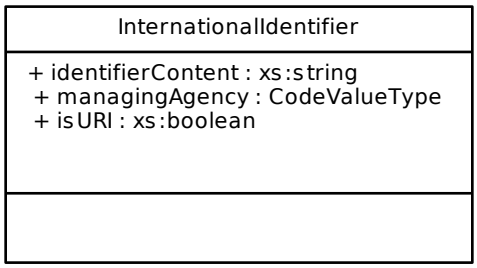
managingAgency

The identification of the Agency which assigns and manages the identifier, i.e., ISBN, ISSN, DOI, etc.

isURI

Set to "true" if Identifier is a URI

Graph



5.14.29 InternationalString

Packaging structure for multiple language versions of the same string content. Where an element of this type is repeatable, the expectation is that each repetition contains different content, each of which can be expressed in multiple languages. The language designation goes on the individual String.

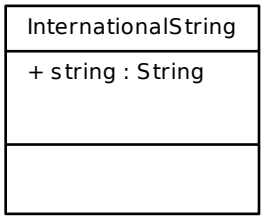
Properties

Name	Type	Cardinality
string	String	0..n

string

A non-formatted string of text with an attribute that designates the language of the text. Repeat this object to express the same content in another language.

Graph



5.14.30 Label

A structured display label. Label provides display content of a fully human readable display for the identification of the object.

Extends

StructuredString

Properties

Name	Type	Cardinality
locationVariant	xs:string	0..1
validForStartDate	Date	0..1
validForEndDate	Date	0..1
maxLength	xs:integer	0..1

locationVariant

Indicate the locality specification for content that is specific to a geographic area. May be a country code, sub-country code, or area name.

validForStartDate

Allows for the specification of a starting date for the period that this label is valid. The date must be formatted according to ISO 8601.

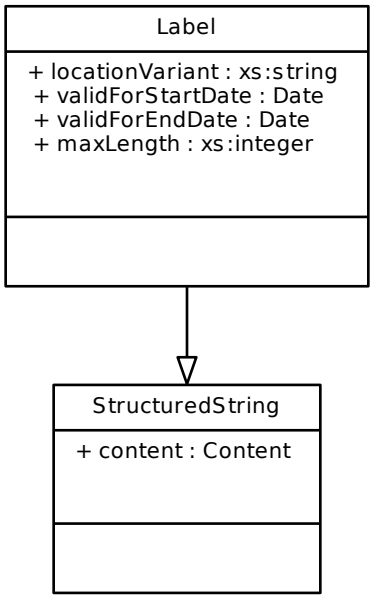
validForEndDate

Allows for the specification of a ending date for the period that this label is valid. The date must be formatted according to ISO 8601.

maxLength

A positive integer indicating the maximum number of characters in the label.

Graph



5.14.31 LineParameter

Specification of the line and offset for the beginning and end of the segment.

Properties

Name	Type	Cardinality
startLine	xs:integer	0..1
startOffset	xs:integer	0..1
endLine	xs:integer	0..1
endOffset	xs:integer	0..1

startLine

Number of lines from beginning of the document.

startOffset

Number of characters from start of the line specified in StartLine.

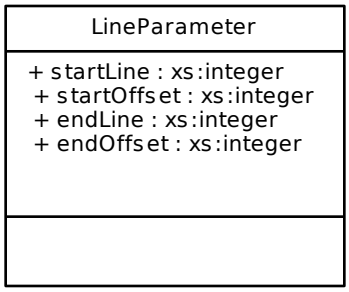
endLine

Number of lines from beginning of the document.

endOffset

Number of characters from the start of the line specified in EndLine.

Graph



5.14.32 LiteralText

Literal (static) text to be used in the instrument using the StructuredString structure plus an attribute allowing for the specification of white space to be preserved.

Extends

TextContent

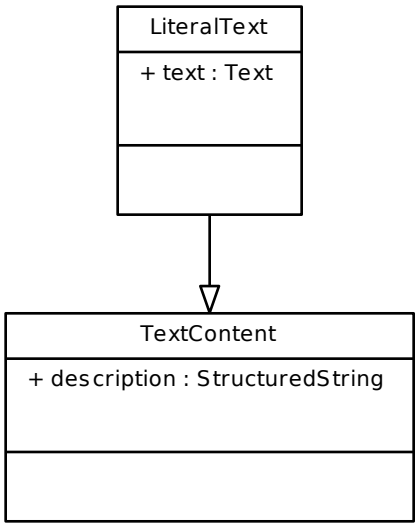
Properties

Name	Type	Cardinality
text	Text	0..1

text

The value of the static text string. Supports the optional use of XHTML formatting tags within the string structure. If the content of a literal text contains more than one language, i.e. "What is your understanding of the German word 'Gesundheit'?", the foreign language element should be placed in a separate LiteralText component with the appropriate xmlang value and, in this case, isTranslatable set to "false". If the existence of white space is critical to the understanding of the content (such as inclusion of a leading or trailing white space), set the attribute of Text xmspace to "preserve".

Graph



5.14.33 LocalId

This is an identifier in a given local context that uniquely references an object, as opposed to the full ddi identifier which has an agency plus the id.

Properties

Name	Type	Cardinality
localIdValue	xs:string	1..1
localIdType	xs:string	1..1
localIdVersion	xs:string	0..1

localIdValue

Value of the local ID.

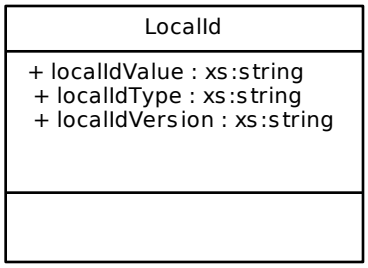
localIdType

Type of identifier, specifying the context of the identifier.

localIdVersion

Version of the Local ID.

Graph



5.14.34 LocationName

Name of the location using the DDI Name structure and the ability to add an effective date.

Extends

Name

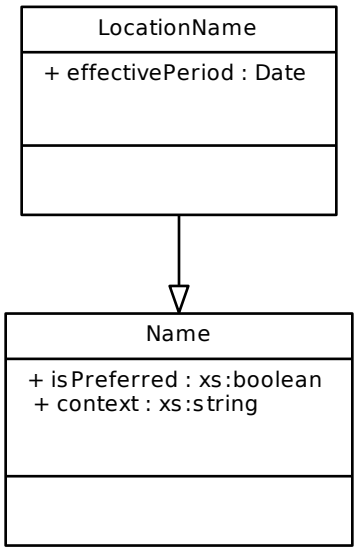
Properties

Name	Type	Cardinality
effectivePeriod	Date	0..1

effectivePeriod

The time period for which this name is accurate and in use.

Graph



5.14.35 Name

A reusable type assigned to an element with the naming convention XxxName e.g. OrganizationName at selected locations where the element may be assumed to be administered by a registry or is otherwise shared. This is a human understandable name (word, phrase, or mnemonic) that reflects the ISO/IEC 11179-5 naming principles. An item administered by a registry should have at least one name. Names within an administered registry should follow the naming conventions of the registry. If more than one name is provided the context of each name should be noted and one name selected as the preferred name. See ISO/IEC 11179-5 Information Technology - Metadata Registries (MDR) Part 5: naming and identification principles. ISO/IEC1179-5:2005(E).

Extends

InternationalString

Properties

Name	Type	Cardinality
isPreferred	xs:boolean	0..1
context	xs:string	0..1

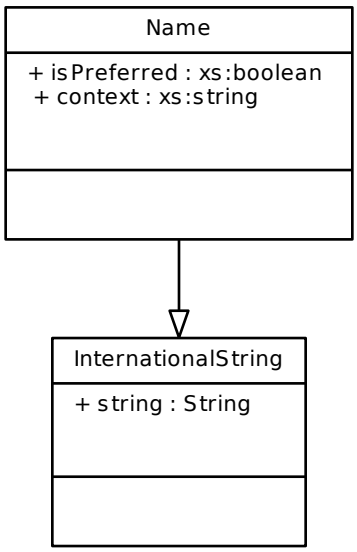
isPreferred

If more than one name for the object is provided, use the isPreferred attribute to indicate which is the preferred name content. All other names should be set to isPreferred="false".

context

A name may be specific to a particular context, i.e., a type of software, or a section of a registry. Identify the context related to the specified name.

Graph



5.14.36 OrganizationName

Names by which the organization is known. Use the attribute isFormal="true" to designate the legal or formal name of the Organization. The preferred name should be noted with the isPreferred attribute. Names may be typed with TypeOfOrganizationName to indicate their appropriate usage.

Extends

Name

Properties

Name	Type	Cardinality
abbreviation	InternationalString	0..1
typeOfOrganizationName	CodeValueType	0..1
effectivePeriod	Date	0..1
isFormal	xs:boolean	0..1

abbreviation

An abbreviation or acronym for the name. This may be expressed in multiple languages. It is assumed that if only a single language is provided that it may be used in any of the other languages within which the name itself is expressed.

typeOfOrganizationName

The type of organization name provided. the use of a controlled vocabulary is strongly recommended. At minimum this should include, e.g. PreviousFormalName, Nickname (or CommonName), Other.

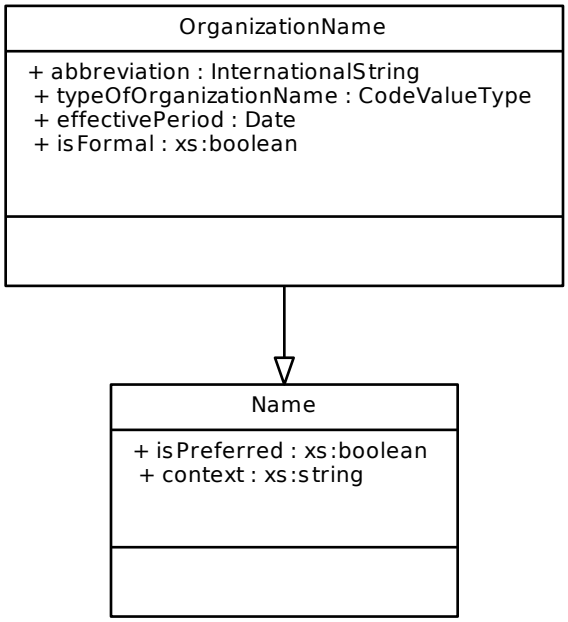
effectivePeriod

The time period for which this name is accurate and in use.

isFormal

The legal or formal name of the organization should have the isFormal attribute set to true. To avoid confusion only one organization name should have the isFormal attribute set to true. Use the TypeOfOrganizationName to further differentiate the type and applied usage when multiple names are provided.

Graph



5.14.37 PairedCodeValueType

A tightly bound pair of items from a controlled vocabulary. The extent property describes the extent to which the parent term applies for the specific case.

Extends

CodeValueType

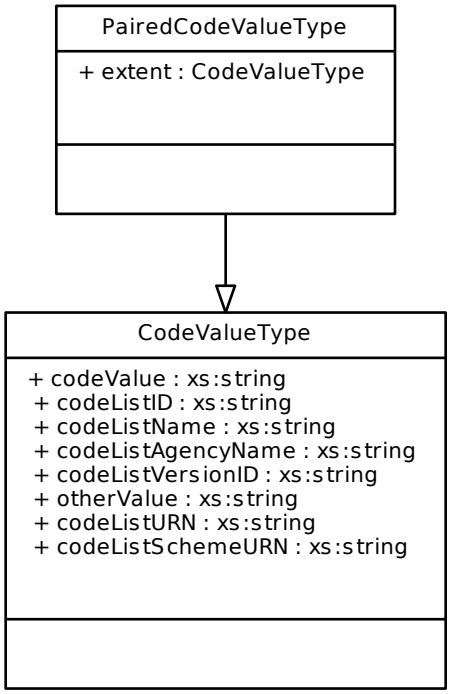
Properties

Name	Type	Cardinality
extent	CodeValueType	0..1

extent

Describes the extent to which the parent term applies for the specific case using an external controlled vocabulary.

Graph



5.14.38 Point

A geographic point consisting of an X and Y coordinate. Each coordinate value is expressed separately providing its value and format.

Properties

Name	Type	Cardinality
xCoordinate	SpatialCoordinate	0..1
yCoordinate	SpatialCoordinate	0..1

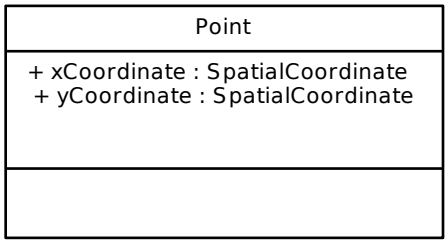
xCoordinate

An X coordinate (latitudinal equivalent) value and format expressed using the Spatial Coordinate structure.

yCoordinate

A Y coordinate (longitudinal equivalent) value and format expressed using the Spatial Coordinate structure.

Graph



5.14.39 Polygon

A closed plane figure bounded by three or more line segments, representing a geographic area. Contains either the URI of the file containing the polygon, a specific link code for the shape within the file, and a file format, or a minimum of 4 points to describe the polygon in-line. Note that the first and last point must be identical in order to close the polygon. A triangle has 4 points. A geographic time designating the time period that the shape is valid should be included. If the date range is unknown use a SingleDate indicating a date that the shape was known to be valid.

Properties

Name	Type	Cardinality
externalURI	xs:anyURI	0..1
polygonLinkCode	xs:string	0..1
shapeFileFormat	CodeValueType	0..1
point	Point	4..n

externalURI

Note that ExternalURI points to the boundary file location.

polygonLinkCode

The PolygonLinkCode is the identifier of the specific polygon within the file. For example in an NHGIS file the LinkCodeForPolygon for Tract 101.01 in Hennepin County in Minnesota is 2700530010101.

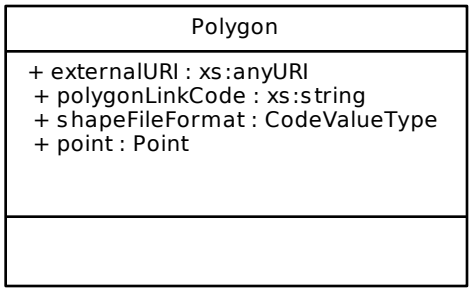
shapeFileFormat

The format of the shape file existing at the location indicated by the sibling ExternalURI element.

point

A geographic point defined by a latitude and longitude. A minimum of 4 points is required as the first and last point should be identical in order to close the polygon. Note that a triangle has three sides and requires 3 unique points plus a fourth point replicating the first point in order to close the polygon.

Graph



5.14.40 PrivateImage

References an image using the standard Image description. In addition to the standard attributes provides an effective date (period), the type of image, and a privacy ranking.

Extends

Image

Properties

Name	Type	Cardinality
effectivePeriod	Date	0..1
privacy	CodeValueType	0..1

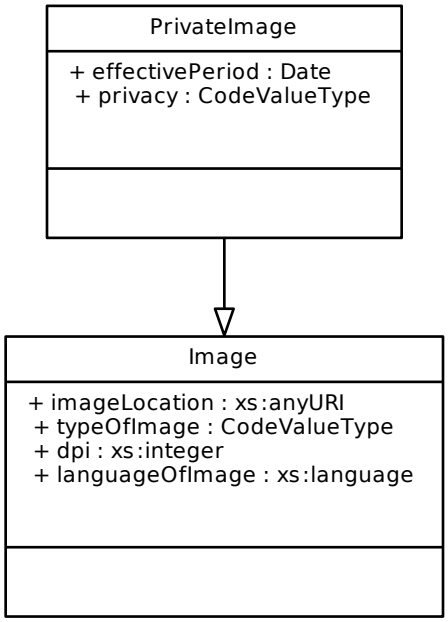
effectivePeriod

The period for which this image is effective/valid.

privacy

Specify the level privacy for the image as public, restricted, or private. Supports the use of an external controlled vocabulary.

Graph



5.14.41 ProprietaryInfo

Contains information proprietary to the software package which produced the data file. This is expressed as a set of key(name)-value pairs.

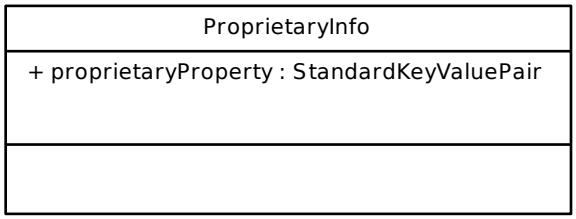
Properties

Name	Type	Cardinality
proprietaryProperty	StandardKeyValuePair	0..n

proprietaryProperty

A structure that supports the use of a standard key value pair. Note that this information is often not interoperable and is provided to support the use of the metadata within specific systems.

Graph



5.14.42 Range

Indicates the range of items expressed as a string, such as an alphabetic range.

Properties

Name	Type	Cardinality
rangeUnit	xs:string	0..1
minimumValue	RangeValue	0..1
maximumValue	RangeValue	0..1

rangeUnit

Specifies the units in the range.

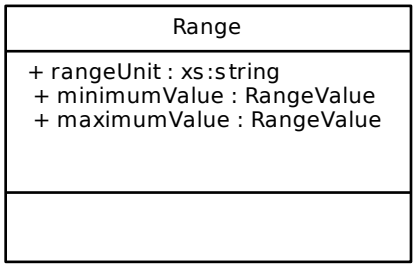
minimumValue

Minimum value in the range.

maximumValue

Maximum value in the range.

Graph



5.14.43 RangeValue

Describes a bounding value of a string.

Extends

Value

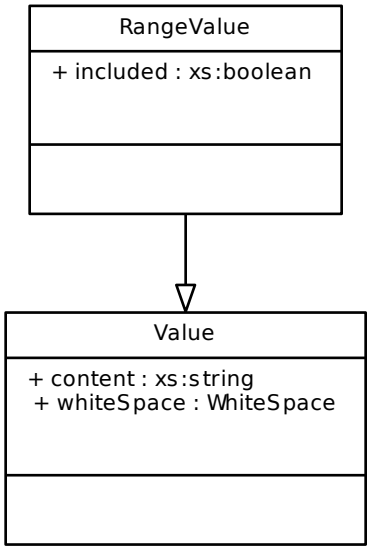
Properties

Name	Type	Cardinality
included	xs:boolean	0..1

included

Set to “true” if the value is included in the range.

Graph



5.14.44 ReferenceDate

The date covered by the annotated object. In addition to specifying a type of date (e.g. collection period, census year, etc.) the date or time span may be associated with a particular subject or keyword. This allows for the expression of a referent date associated with specific subjects or keywords. For example, a set of date items on income and labor force status may have a referent date for the year prior to the collection date.

Extends

AnnotationDate

Properties

Name	Type	Cardinality
subject	InternationalCodeValueType	0..n
keyword	InternationalCodeValueType	0..n

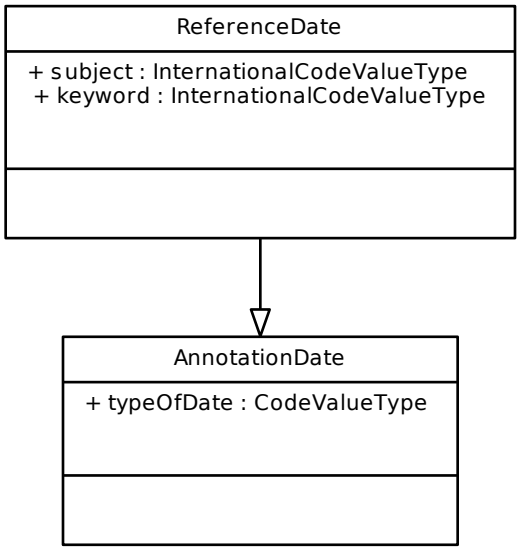
subject

If the date is for a subset of data only such as a referent date for residence 5 years ago, use Subject to specify the coverage of the data this date applies to. May be repeated to reflect multiple subjects.

keyword

If the date is for a subset of data only such as a referent date for residence 5 years ago, use keyword to specify the coverage of the data this date applies to. May be repeated to reflect multiple keywords.

Graph



5.14.45 Relationship

Relationship specification between this item and the item to which it is related. Provides a reference to any identifiable object and a description of the relationship.

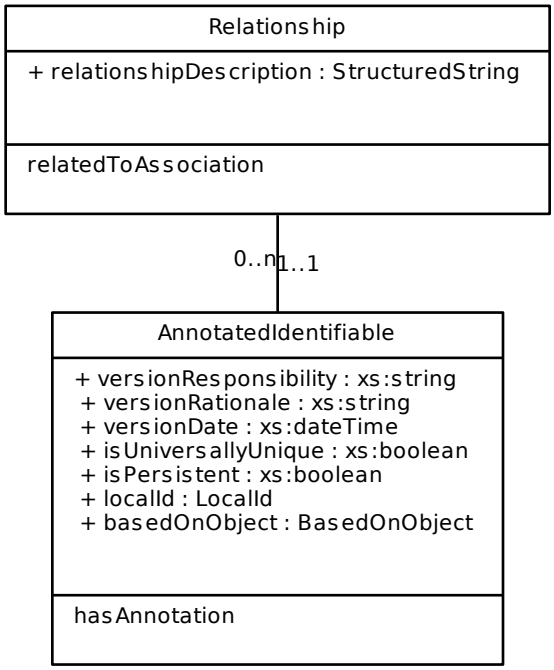
Properties

Name	Type	Cardinality
relationshipDescription	StructuredString	0..1

relationshipDescription

A description of the nature of the relationship between the parent element of the relationship item and the DDI object to which it is related.

Graph



5.14.46 ResourceIdentifier

Provides a means of identifying a related resource and provides the typeOfRelationship. Makes use of a controlled vocabulary for typing the relationship. Standard usage may include: describesDate, isDescribedBy, isFormatOf, isPartOf, isReferencedBy, isReplacedBy, isRequiredBy, isVersionOf, references, replaces, requires, etc.

Extends

InternationalIdentifier

Properties

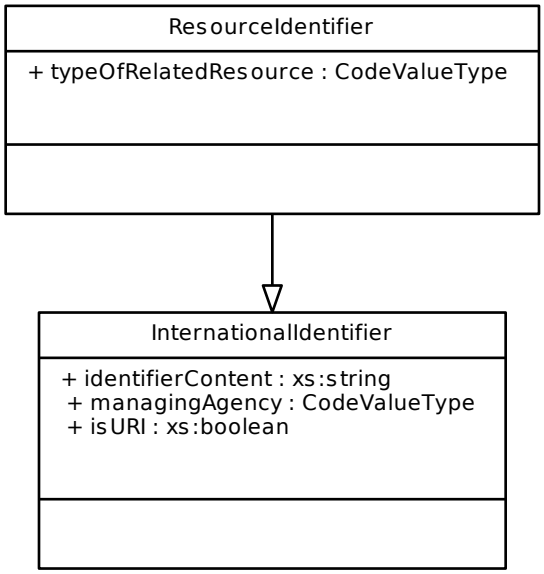
Name	Type	Cardinality
typeOfRelatedResource	CodeValueType	0..n

typeOfRelatedResource

The type of relationship between the annotated object and the related resource. Standard usage may include: describesDate, isDescribedBy, isFormatOf, isPartOf, isReferencedBy, isReplacedBy, isRequiredBy, isVersionOf, references,

replaces, requires, etc.

Graph



5.14.47 Segment

A structure used to express explicit segments or regions within different types of external materials (Textual, Audio, Video, XML, and Image). Provides the appropriate start, stop, or region definitions for each type.

Properties

Name	Type	Cardinality
audioSegment	AudioSegment	0..n
videoSegment	VideoSegment	0..n
xml	xs:string	0..n
textualSegment	TextualSegment	0..n
imageArea	ImageArea	0..n

audioSegment

Describes the type and length of the audio segment.

videoSegment

Describes the type and length of the video segment.

xml

An X-Pointer expression identifying a node in the XML document.

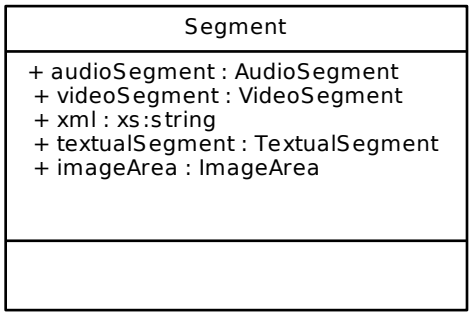
textualSegment

Defines the segment of textual content used by the parent object. Can identify a set of lines and or characters used to define the segment

imageArea

Defines the shape and area of an image used as part of a location representation. The shape is defined as a Rectangle, Circle, or Polygon and Coordinates provides the information required to define it.

Graph



5.14.48 Software

Describes a specific software package, which may be commercially available or custom-made.

Properties

Name	Type	Cardinality
softwareName	Name	0..n
softwarePackage	CodeValueType	0..1
softwareVersion	xs:string	0..1
description	StructuredString	0..1
date	Date	0..1
function	CodeValueType	0..n
xmlLang	xs:language	0..1

softwareName

The name of the software package, including its producer.

softwarePackage

A coded value from a controlled vocabulary, describing the software package.

softwareVersion

The version of the software package. Defaults to '1.0'.

description

A description of the content and purpose of the software. May be expressed in multiple languages and supports the use of structured content.

date

Supported date of the software package with, at minimum, a release date if known.

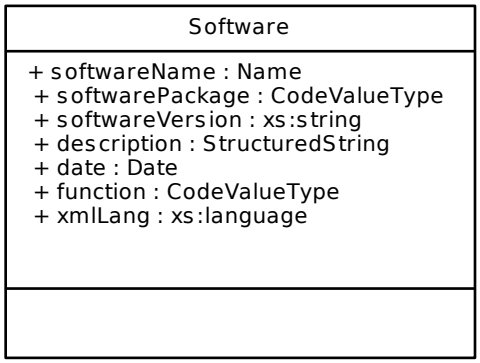
function

Identifies the functions handled by this software. Repeat for multiple functions. It may be advisable to note only those functions used in the specific usage of the software.

xmlLang

Language (human language) of the software package.

Graph



5.14.49 SpatialCoordinate

Lists the value and format type for the coordinate value. Note that this is a single value (X coordinate or Y coordinate) rather than a coordinate pair.

Properties

Name	Type	Cardinality
coordinateValue	xs:string	0..1
coordinateType	PointFormat	1..1

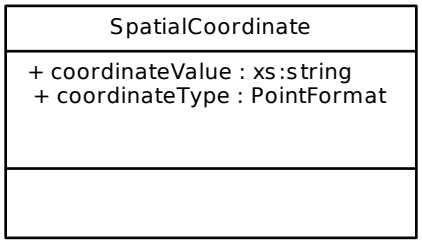
coordinateValue

The value of the coordinate expressed as a string.

coordinateType

Identifies the type of point coordinate system using a controlled vocabulary. Point formats include decimal degree, degrees minutes seconds, decimal minutes, meters, and feet.

Graph



5.14.50 SpecificSequence

Describes the ordering of items when not otherwise indicated. There are a set number of values for ItemSequenceType, but also a provision for describing an alternate ordering using a command language.

Properties

Name	Type	Cardinality
itemSequence	ItemSequence	1..1
alternateSequence	CommandCode	0..1

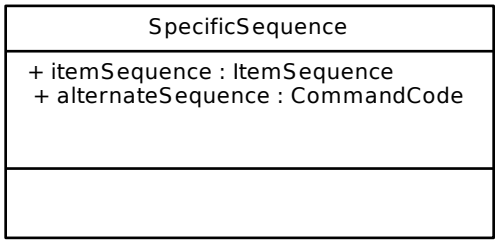
itemSequence

Identifies the type of sequence to use. Values include InOrderOfAppearance, Random, Rotate, and Other.

alternateSequence

Information on the command used to generate an alternative means of determining sequence changes. If used, the ItemSequenceType should be "Other".

Graph



5.14.51 StandardKeyValuePair

A basic data representation for computing systems and applications expressed as a tuple (attribute key, value). Attribute keys may or may not be unique.

Properties

Name	Type	Cardinality
attributeKey	CodeValueType	0..1
attributeValue	CodeValueType	0..1

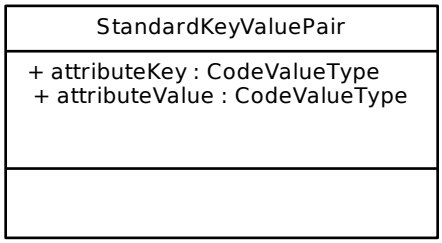
attributeKey

This key (sometimes referred to as a name) expressed as a string. Supports the use of an external controlled vocabulary which is the recommended approach.

attributeValue

The value assigned to the named Key expressed as a string. Supports the use of an external controlled vocabulary.

Graph



5.14.52 Statistic

The value of the statistics and whether it is weighted and/or includes missing values.

Properties

Name	Type	Cardinality
isWeighted	xs:boolean	0..1
computationBase	ComputationBaseList	0..1
content	xs:string	1..1

isWeighted

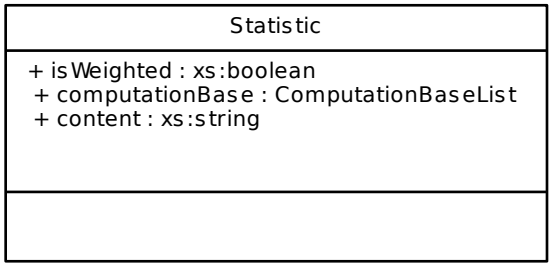
Set to “true” if the statistic is weighted using the weight designated in VariableStatistics.

computationBase

Defines the cases included in determining the statistic. The options are total=all cases, valid and missing (invalid); validOnly=Only valid values, missing (invalid) are not included in the calculation; missingOnly=Only missing (invalid) cases included in the calculation.

content

Graph



5.14.53 String

Allows for non-formatted strings that may be translations from other languages, or that may be translatable into other languages. Only one string per language/location type is allowed. String contains the following attributes, xmlang to designate the language, isTranslated with a default value of false to designate if an object is a translation of another language, isTranslatable with a default value of true to designate if the content can be translated, translationSourceLanguage to indicate the source languages used in creating this translation, and translationDate.

Properties

Name	Type	Cardinality
content	xs:string	1..1
xmlLang	xs:language	0..1
isTranslated	xs:boolean	0..1
isTranslatable	xs:boolean	0..1
translationSourceLanguage	xs:language	0..n
translationDate	xs:date	0..1

content

value of this string

xmlLang

Indicates the language of content. Note that xmlang allows for a simple 2 or 3 character language code or a language code extended by a country code , for example en-au for English as used in Australia.

isTranslated

Indicates whether content is a translation (true) or an original (false).

isTranslatable

Indicates whether content is translatable (true) or not (false). An example of something that is not translatable would be a MNEMONIC of an object or a number.

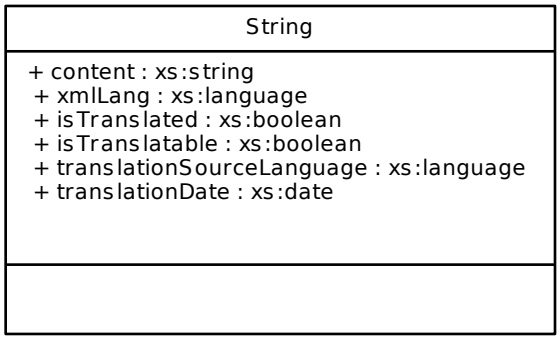
translationSourceLanguage

List the language code of the source. Repeat of multiple language sources are used.

translationDate

The date the content was translated. Provision of translation date allows user to verify if translation was available during data collection or other time linked activity.

Graph



5.14.54 StructuredString

Packaging structure for multiple language versions of the same string content, for objects that allow for internal formatting using XHTML tags. Where an element of this type is repeatable, the expectation is that each repetition contains different content, each of which can be expressed in multiple languages.

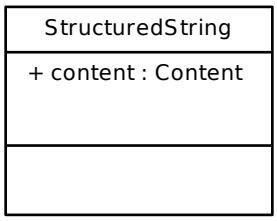
Properties

Name	Type	Cardinality
content	Content	1..n

content

Supports the optional use of XHTML formatting tags within the string structure. In addition to the language designation and information regarding translation, the attribute isPlain can be set to true to indicate that the content should be treated as plain unstructured text, including any XHTML formatting tags. Repeat the content element to provide multiple language versions of the same content.

Graph



5.14.55 Telephone

Details of a telephone number including the number, type of number, a privacy setting and an indication of whether this is the preferred contact number.

Properties

Name	Type	Cardinality
telephoneNumber	xs:string	0..1
typeOfTelephone	CodeValueType	0..1
effectivePeriod	Date	0..1
privacy	CodeValueType	0..1
isPreferred	xs:boolean	0..1

telephoneNumber

The telephone number including country code if appropriate.

typeOfTelephone

Indicates type of telephone number provided (home, fax, office, cell, etc.). Supports the use of a controlled vocabulary.

effectivePeriod

Time period during which the telephone number is valid.

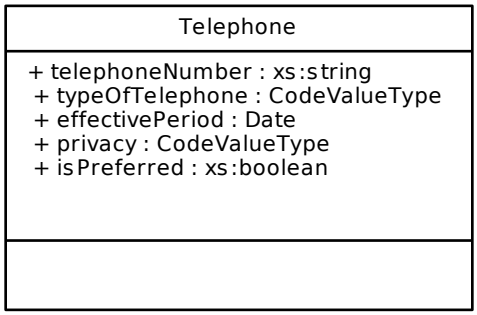
privacy

Specify the level privacy for the telephone number as public, restricted, or private. Supports the use of an external controlled vocabulary.

isPreferred

Set to "true" if this is the preferred telephone number for contact.

Graph



5.14.56 Text

The static portion of the text expressed as a StructuredString with the ability to preserve whitespace if critical to the understanding of the content.

Extends

Content

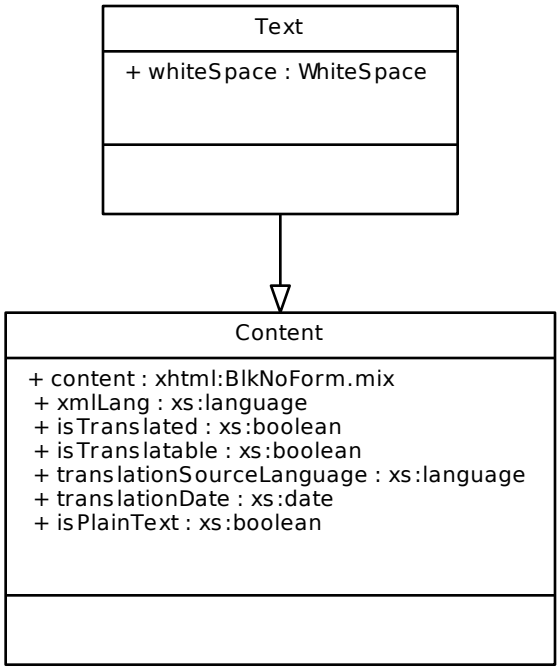
Properties

Name	Type	Cardinality
whiteSpace	WhiteSpace	0..1

whiteSpace

The default setting states that leading and trailing white space will be removed and multiple adjacent white spaces will be treated as a single white space. If the existance of any of these white spaces is critical to the understanding of the content, change the value of this attribute to "preserve".

Graph



5.14.57 TextContent

Abstract type existing as the head of a substitution group. May be replaced by any valid member of the substitution group TextContent. Provides the common element Description to all members using TextContent as an extension base.

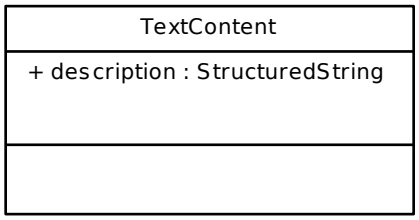
Properties

Name	Type	Cardinality
description	StructuredString	0..1

description

A description of the content and purpose of the text segment. May be expressed in multiple languages and supports the use of structured content.

Graph



5.14.58 TextualSegment

Defines the segment of textual content used by the parent object. Can identify a set of lines and or characters used to define the segment.

Properties

Name	Type	Cardinality
lineParamenter	LineParameter	1..1
characterParameter	CharacterOffset	1..1

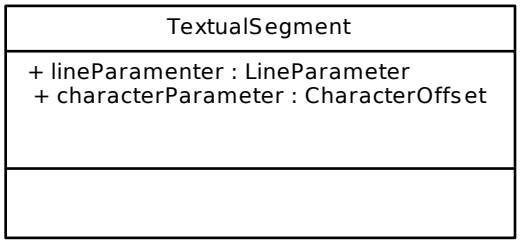
lineParamenter

Specification of the line and offset for the beginning and end of the segment.

characterParameter

Specification of the character offset for the beginning and end of the segment.

Graph



5.14.59 URI

A URN or URL for a file with a flag to indicate if it is a public copy.

Properties

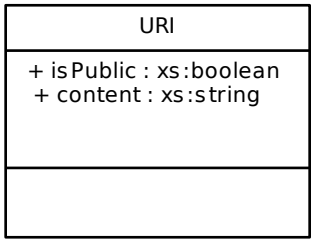
Name	Type	Cardinality
isPublic	xs:boolean	0..1
content	xs:string	1..1

isPublic

Set to “true” (default value) if this file is publicly available. This does not imply that there are not restrictions to access. Set to “false” if this is not publicly available, such as a backup copy, an internal processing data file, etc.

content

Graph



5.14.60 URL

A web site URL

Properties

Name	Type	Cardinality
isPreferred	xs:boolean	0..1
content	xs:anyURI	1..1
typeOfWebsite	CodeValueType	0..1
effectivePeriod	Date	0..1
privacy	CodeValueType	0..1

isPreferred

Set to “true” if this is the preferred URL.

content

The content of the URL

typeOfWebsite

The type of URL for example personal, project, organization, division, etc.

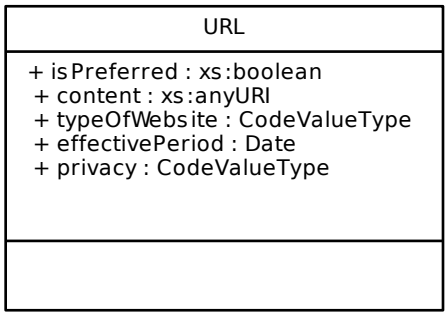
effectivePeriod

The period for which this URL is valid.

privacy

Indicates the privacy level of this URL

Graph



5.14.61 Value

The Value expressed as an xs:string with the ability to preserve whitespace if critical to the understanding of the content.

Properties

Name	Type	Cardinality
content	xs:string	1..1
whiteSpace	WhiteSpace	0..1

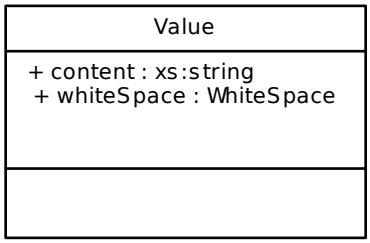
content

The actual content of this value as a string

whiteSpace

The default setting states that leading and trailing white space will be removed and multiple adjacent white spaces will be treated as a single white space. If the existence of any of these white spaces is critical to the understanding of the content, change the value of this attribute to “preserve”.

Graph



5.14.62 VideoSegment

Describes the type and length of the video segment.

Properties

Name	Type	Cardinality
typeOfVideoClip	CodeValueType	0..1
videoClipBegin	xs:string	0..1
videoClipEnd	xs:string	0..1

typeOfVideoClip

The type of video clip provided. Supports the use of a controlled vocabulary.

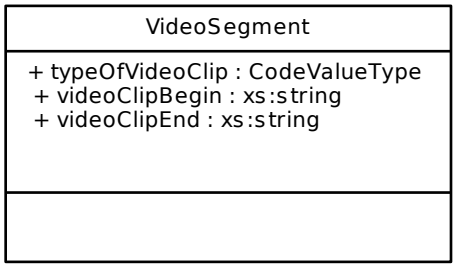
videoClipBegin

The point to begin the video clip. If no point is provided the assumption is that the start point is the beginning of the clip provided.

videoClipEnd

The point to end the video clip. If no point is provided the assumption is that the end point is the end of the clip provided.

Graph



5.14.63 XMLPrefixMap

Maps a specified prefix to a namespace. For each XML namespace used in the profile's XPath expressions, the XML namespaces must have their prefix specified using this element.

Properties

Name	Type	Cardinality
xmlPrefix	xs:string	0..1
xmlNamespace	xs:string	0..1

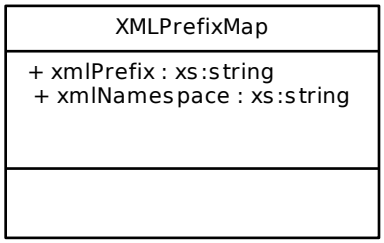
xmlPrefix

Specify the exact prefix used.

xmlNamespace

Specify the namespace which the prefix represents.

Graph



5.15 Conceptual

Conceptual covers all of the basic components of ISO/IEC 11179 as captured and represented in the GSIM Concepts Group http://www1.unece.org/stat/platform/display/GSIMclick/Concepts+Group The Conceptual package will review all parts of the GSIM Concepts Group content and determine if and where adjustments need to be made to interact well with the overall mandate of DDI regarding support for work outside of the GSIM sphere and required levels of abstraction.

Contents

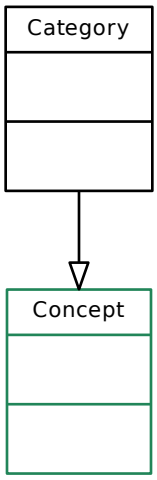
5.15.1 Category

A Concept whose role is to define and measure a characteristic.

Extends

Concept

Graph



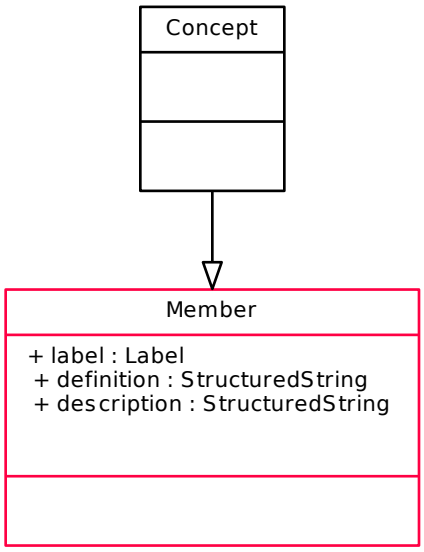
5.15.2 Concept

Unit of thought differentiated by characteristics [GSIM 1.1]

Extends

Member

Graph



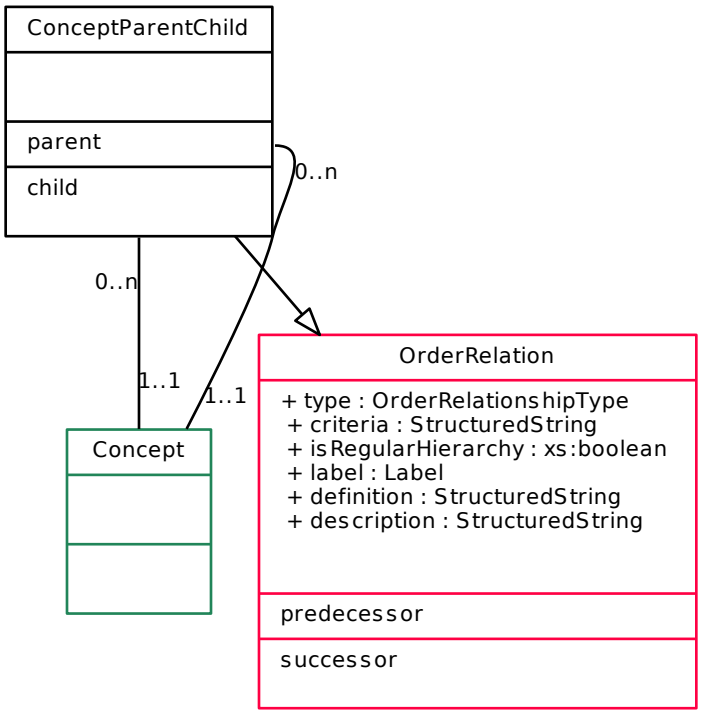
5.15.3 ConceptParentChild

Parent-child specialization of OrderRelation between Concepts within a ConceptSystem.

Extends

OrderRelation

Graph



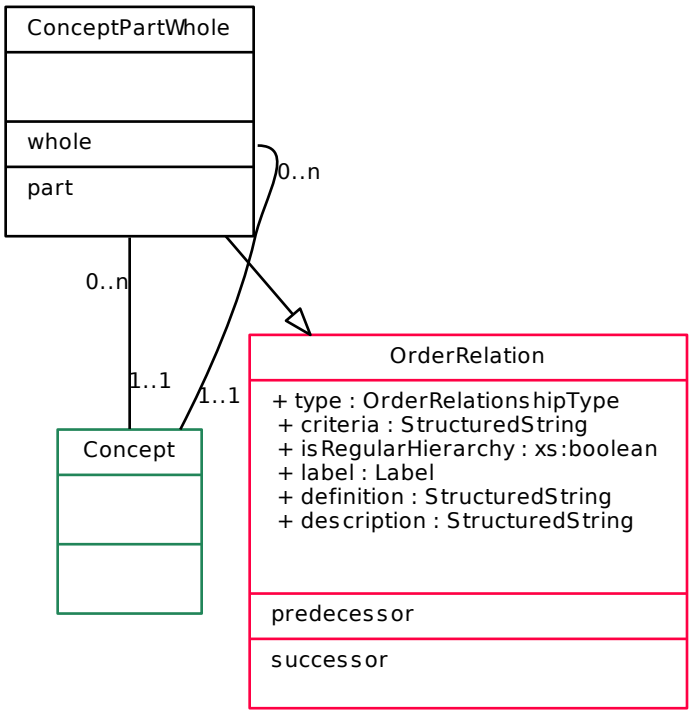
5.15.4 ConceptPartWhole

Part-whole specialization of OrderRelation between Concepts within a ConceptSystem.

Extends

OrderRelation

Graph



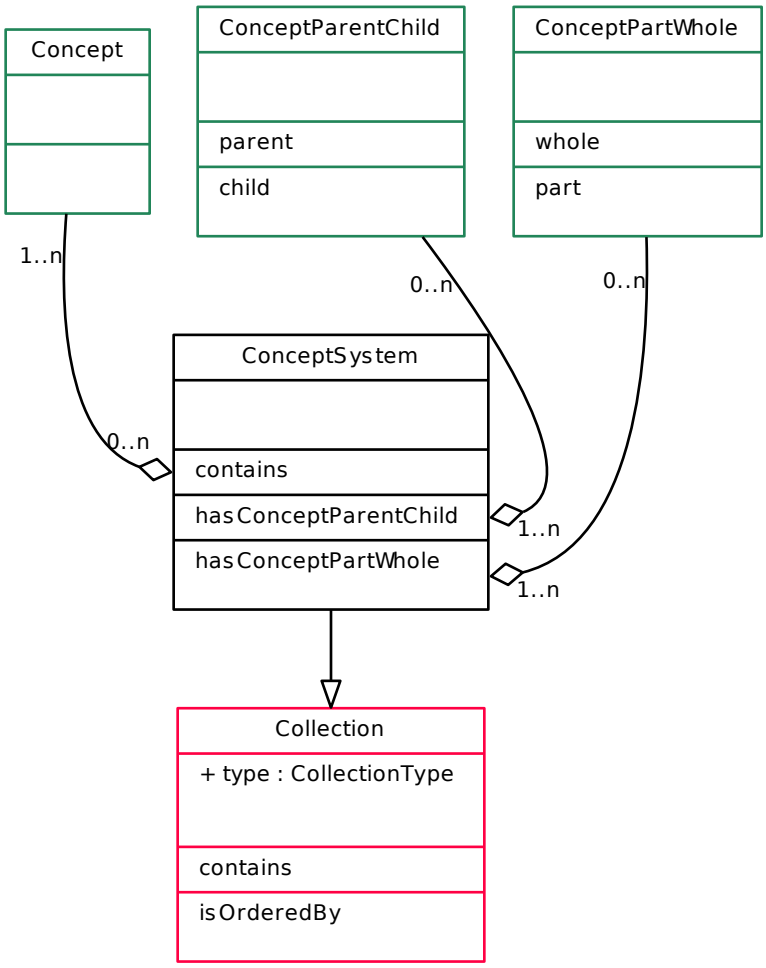
5.15.5 ConceptSystem

A set of Concepts structured by the relations among them. [GSIM 1.1]

Extends

Collection

Graph

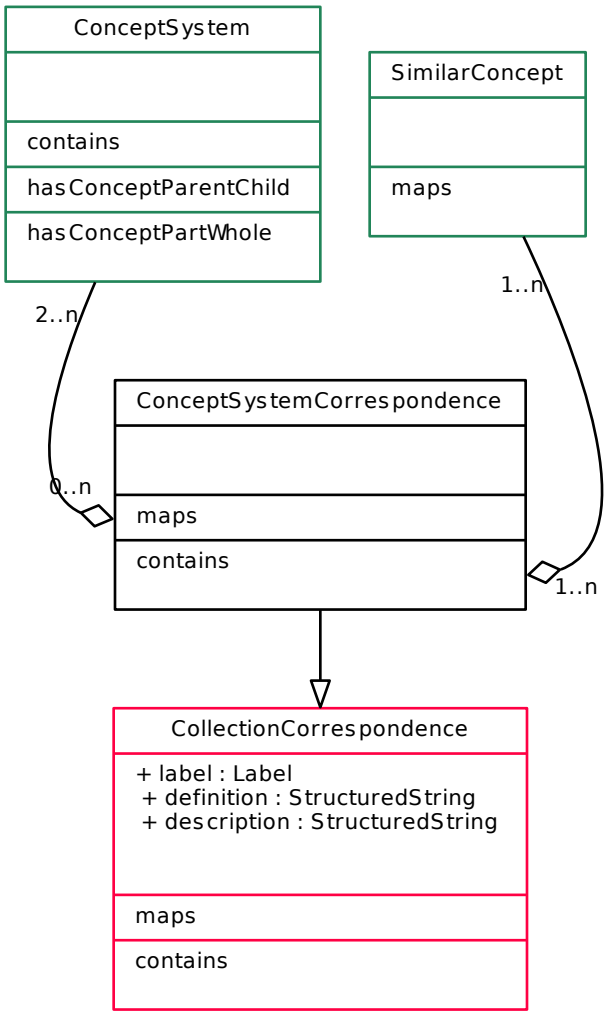


5.15.6 ConceptSystemCorrespondence

Extends

CollectionCorrespondence

Graph



5.15.7 ConceptualDomain

Set of valid Concepts. The Concepts can be described by either enumeration or by an expression.

Extends

AnnotatedIdentifiable

Properties

Name	Type	Cardinality
label	Label	0..n
definition	StructuredString	0..1
description	StructuredString	0..1

label

A display label for the Conceptual Domain. May be expressed in multiple languages. Repeat for labels with different content, for example, labels with differing length limitations.

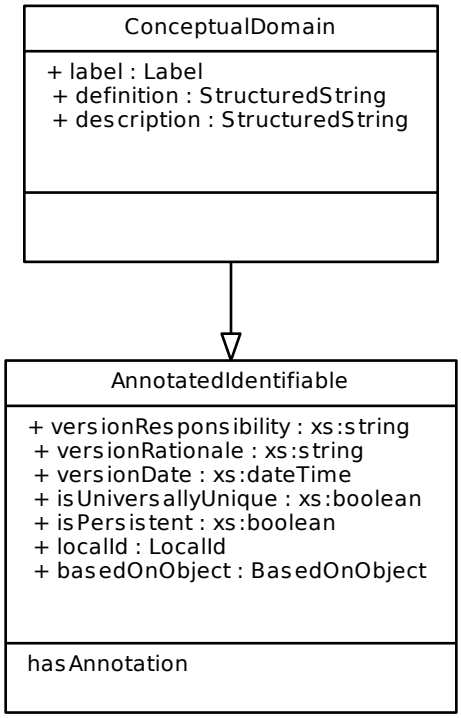
definition

A description of the content and purpose of the Conceptual Domain. May be expressed in multiple languages and supports the use of structured content.

description

A description of the purpose or use of a concept. May be expressed in multiple languages and supports the use of structured content.

Graph



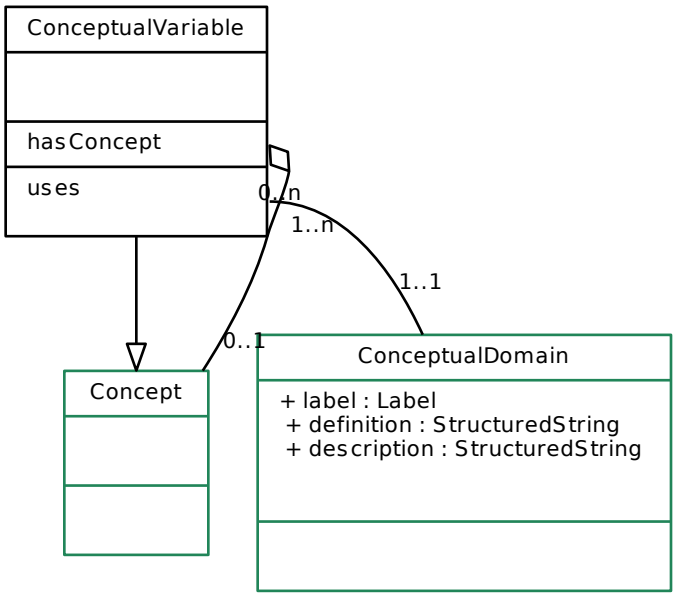
5.15.8 ConceptualVariable

The use of a Concept as a characteristic of a Universe intended to be measured [GSIM 1.1]

Extends

Concept

Graph



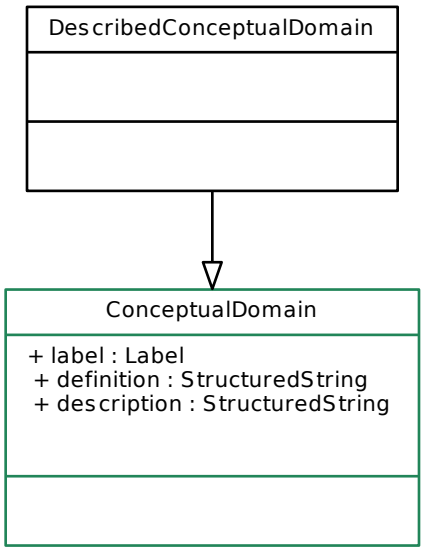
5.15.9 DescribedConceptualDomain

A Conceptual Domain defined by an expression.

Extends

ConceptualDomain

Graph



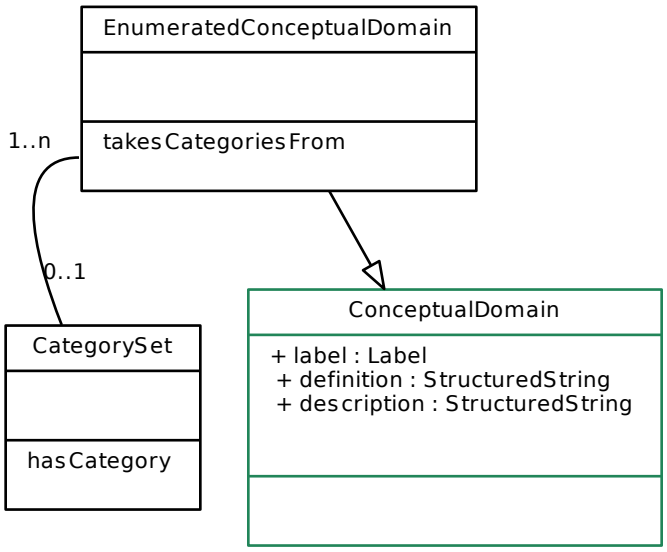
5.15.10 EnumeratedConceptualDomain

A Conceptual Domain expressed as a list of Categories.

Extends

ConceptualDomain

Graph



5.15.11 InstanceVariable

The use of a Represented Variable within a Data Set.

Extends

RepresentedVariable

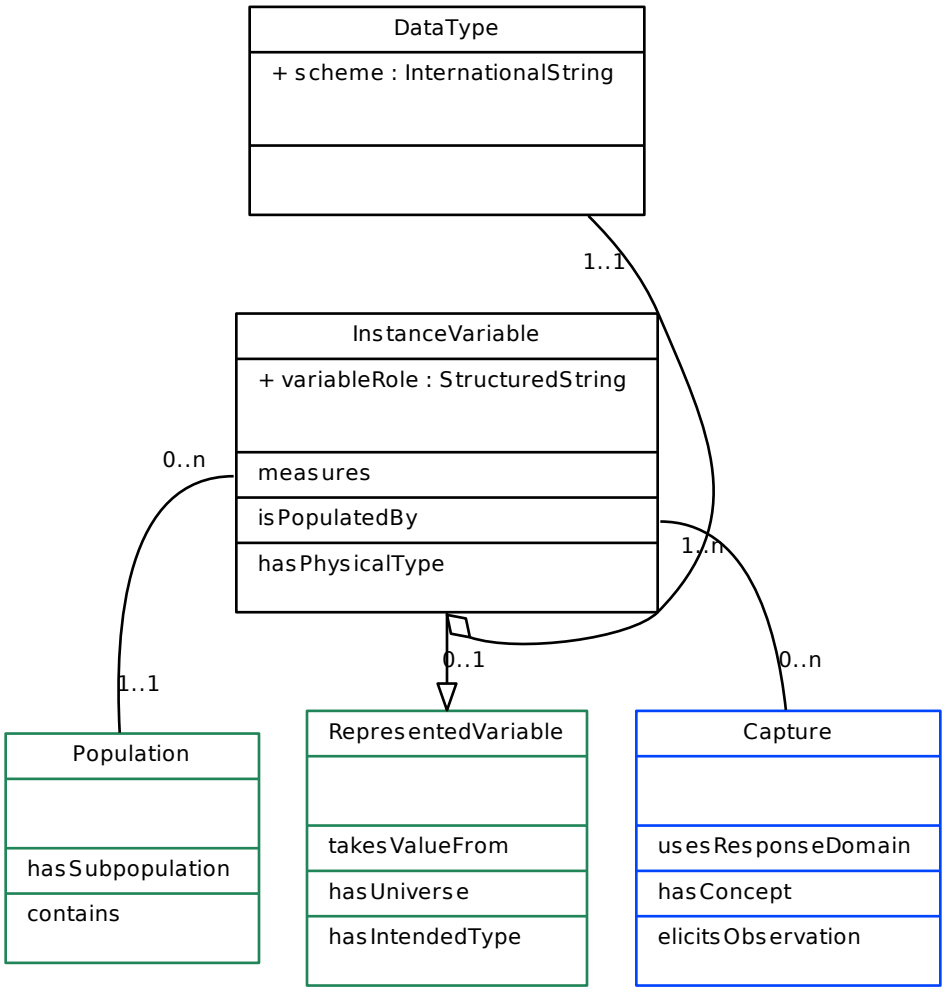
Properties

Name	Type	Cardinality
variableRole	StructuredString	0..1

variableRole

An Instance Variable can take different roles, e.g. Identifier, Measure and Attribute. Note that DataStructure takes care of the ordering of Identifiers.

Graph



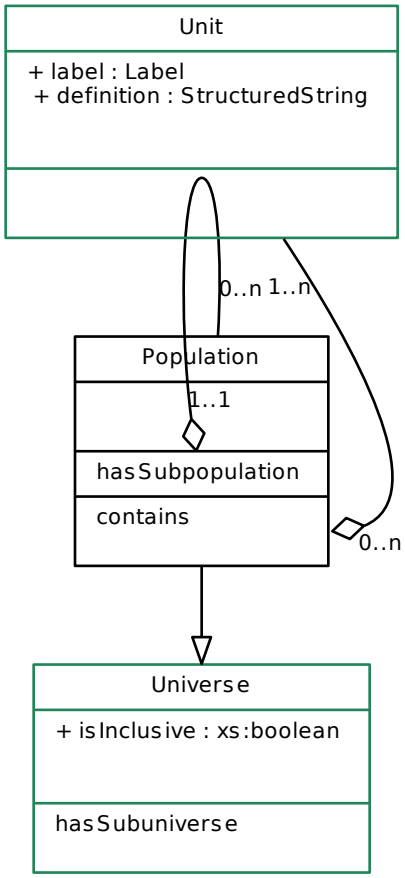
5.15.12 Population

Set of specific units (people, entities, objects, events), usually in a given time and geography.

Extends

Universe

Graph



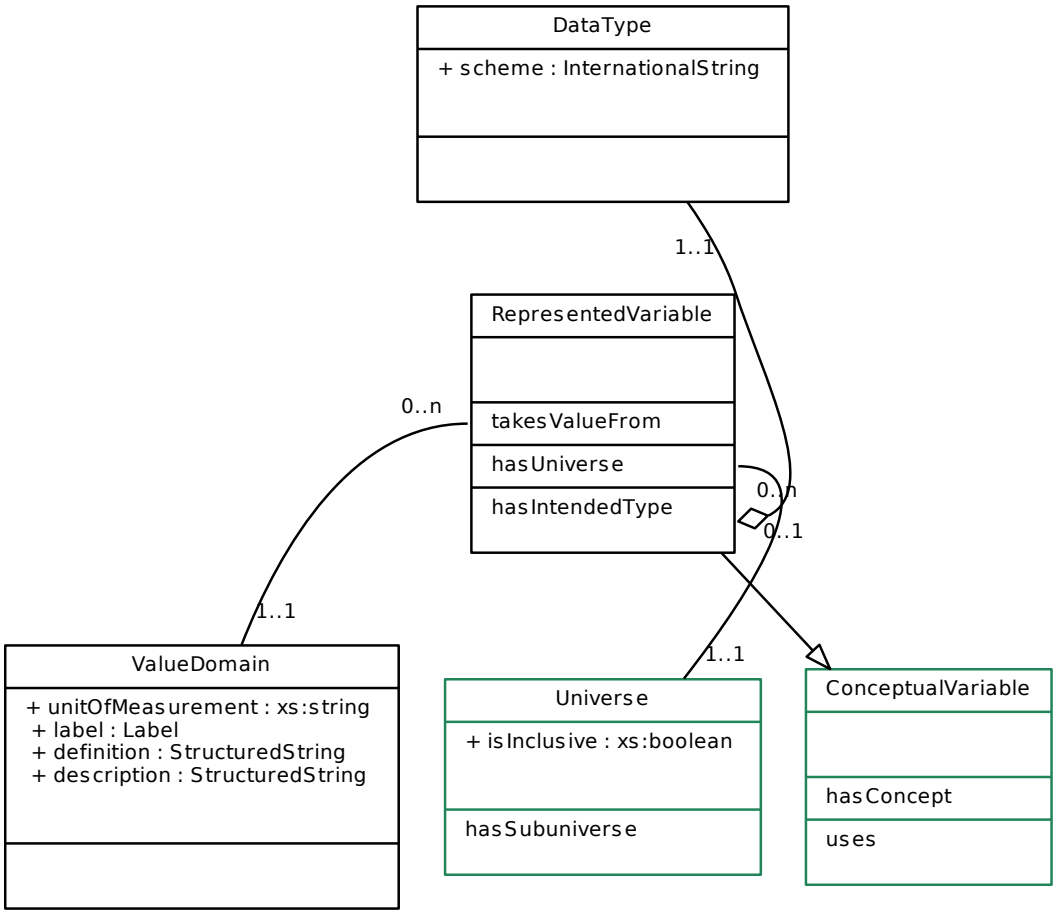
5.15.13 RepresentedVariable

A combination of a characteristic of a universe to be measured and how that measure will be represented.

Extends

ConceptualVariable

Graph



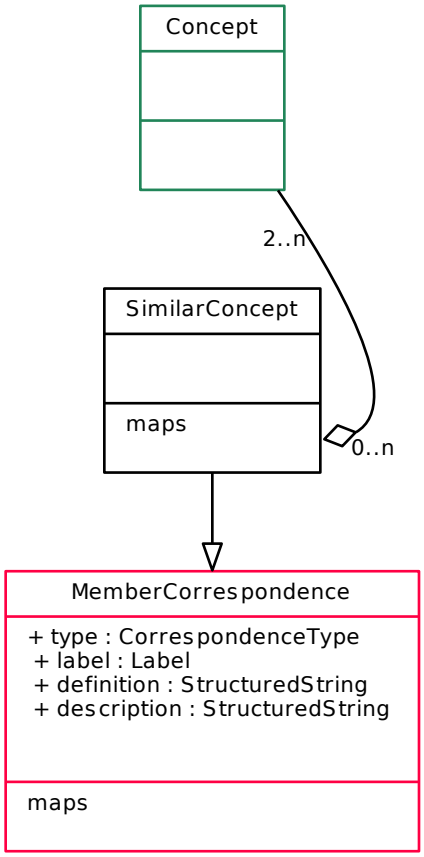
5.15.14 SimilarConcept

A reference to a concept with similar meaning and a description of their differences. The similar concept structure allows specification of similar concepts to address cases where confusion may affect the appropriate use of the concept.

Extends

MemberCorrespondence

Graph



5.15.15 Unit

The object of interest in a process step related to the collection or use of observational data.

Extends

AnnotatedIdentifiable

Properties

Name	Type	Cardinality
label	Label	0..n
definition	StructuredString	0..1

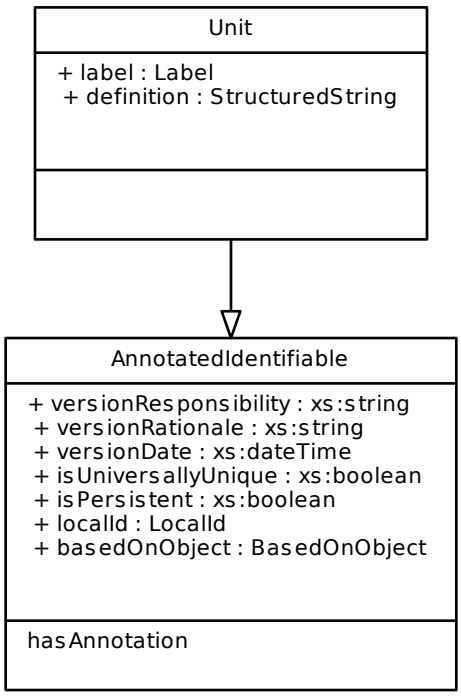
label

A display label for the Unit. May be expressed in multiple languages. Repeat for labels with different content, for example, labels with differing length limitations.

definition

A description of the content and purpose of the Unit. May be expressed in multiple languages and supports the use of structured content.

Graph



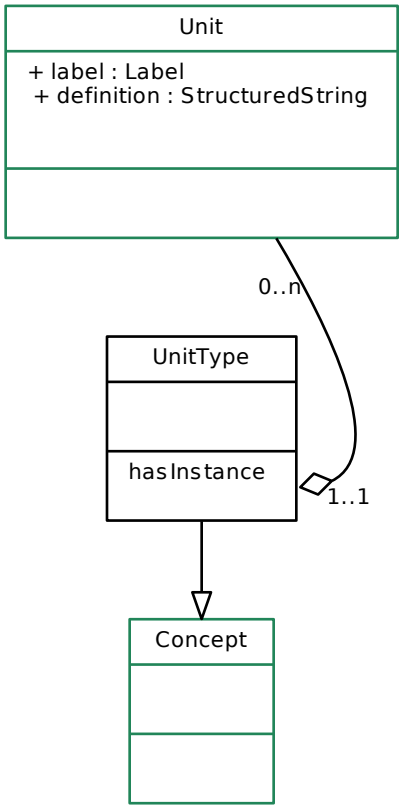
5.15.16 UnitType

A Unit Type is a class of objects of interest.

Extends

Concept

Graph



5.15.17 Universe

A defined class of people, entities, events, or objects, with no specification of time and geography, contextualizing a Unit Type

Extends

UnitType

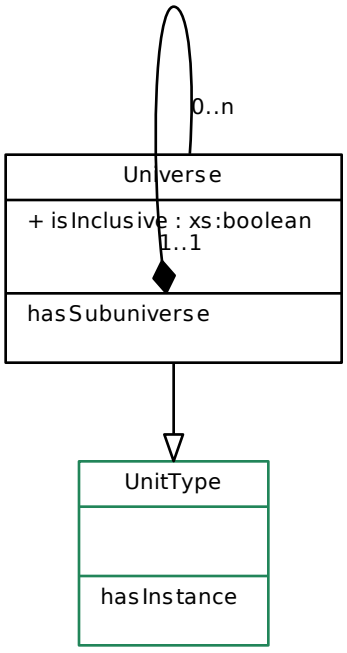
Properties

Name	Type	Cardinality
isInclusive	xs:boolean	0..1

isInclusive

The default value is “true”. The description statement of a universe is generally stated in inclusive terms such as “All persons with university degree”. Occasionally a universe is defined by what it excludes, i.e., “All persons except those with university degree”. In this case the value would be changed to “false”.

Graph



5.16 DataCapture

The DataCapture package contains objects used to describe the process of collecting, acquiring, or capturing data from various sources. Sources could include surveys, databases, registries, administrative data, bio-medical devices, environmental sensors, or any other source or instrument.

Note: this package was previously referred to as SimpleInstrument. Contents

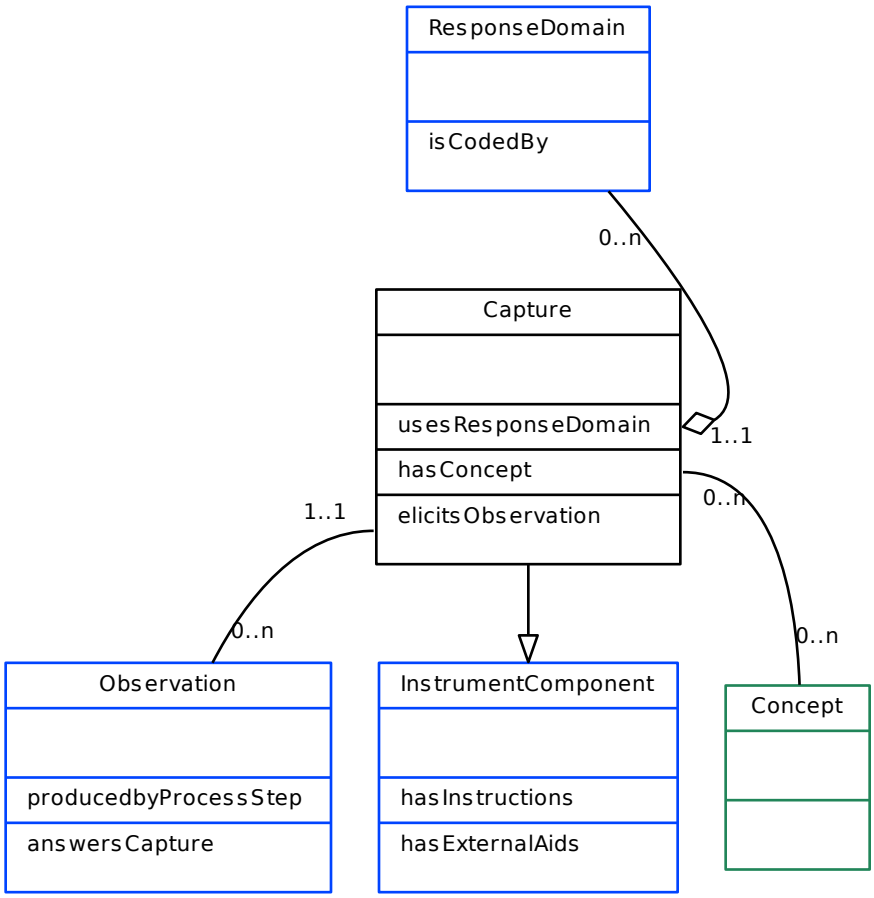
5.16.1 Capture

A Capture is an abstract object. Concrete objects that extend Capture describe the means of obtaining research data.

Extends

InstrumentComponent

Graph



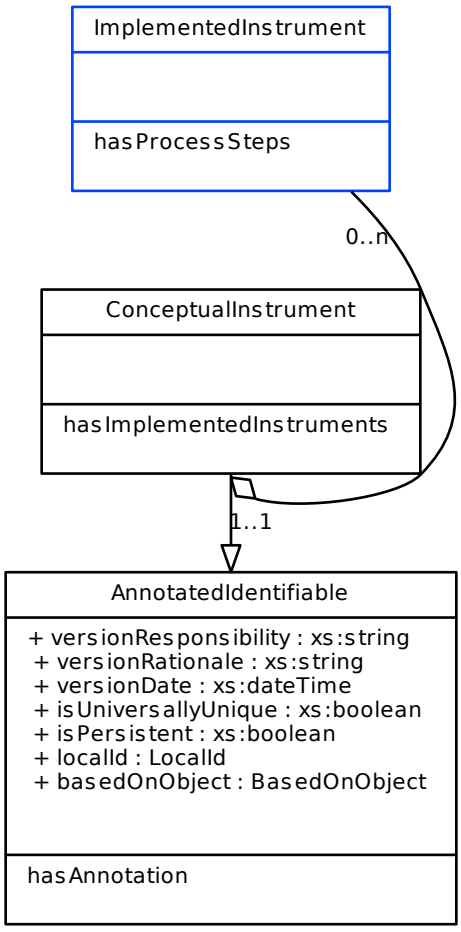
5.16.2 ConceptualInstrument

Design plan for creating a data capture tool. From GSIM: The complete questionnaire design with a relationship to the top level questionnaire component (control construct).

Extends

AnnotatedIdentifiable

Graph



5.16.3 ExternalAid

Any external stimulus material used in an instrument that aids or facilitates data capture, or that is presented to a respondent and about which measurements are made.

Extends

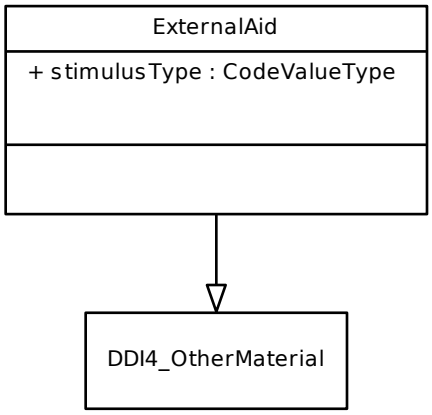
OtherMaterial

Properties

Name	Type	Cardinality
stimulusType	CodeValueType	0..1

stimulusType

Graph



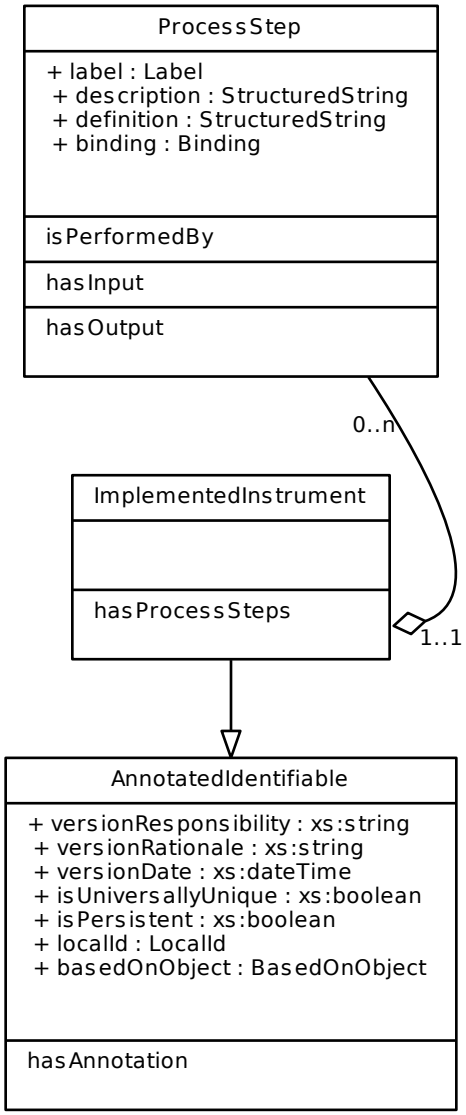
5.16.4 ImplementedInstrument

ImplementedInstruments are mode and/or unit specific.

Extends

AnnotatedIdentifiable

Graph



5.16.5 Instruction

Provides the content and description of data capture instructions. Contains the "how to" information for administering an instrument.

Extends

InstrumentComponent

Properties

Name	Type	Cardinality
associatedImage	StructuredString	0..1
instructionText	DynamicText	0..n

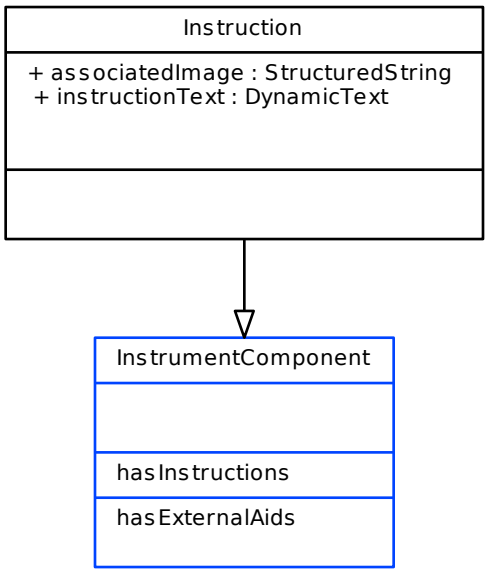
associatedImage

An image associated with the Instruction, located at the provided URN or URL.

instructionText

The content of the Instruction text provided using DynamicText. Note that when using Dynamic Text, the full InstructionText must be repeated for multi-language versions of the content. The InstructionText may also be repeated to provide a dynamic and plain text version of the instruction. This allows for accurate rendering of the instruction in a non-dynamic environment like print.

Graph



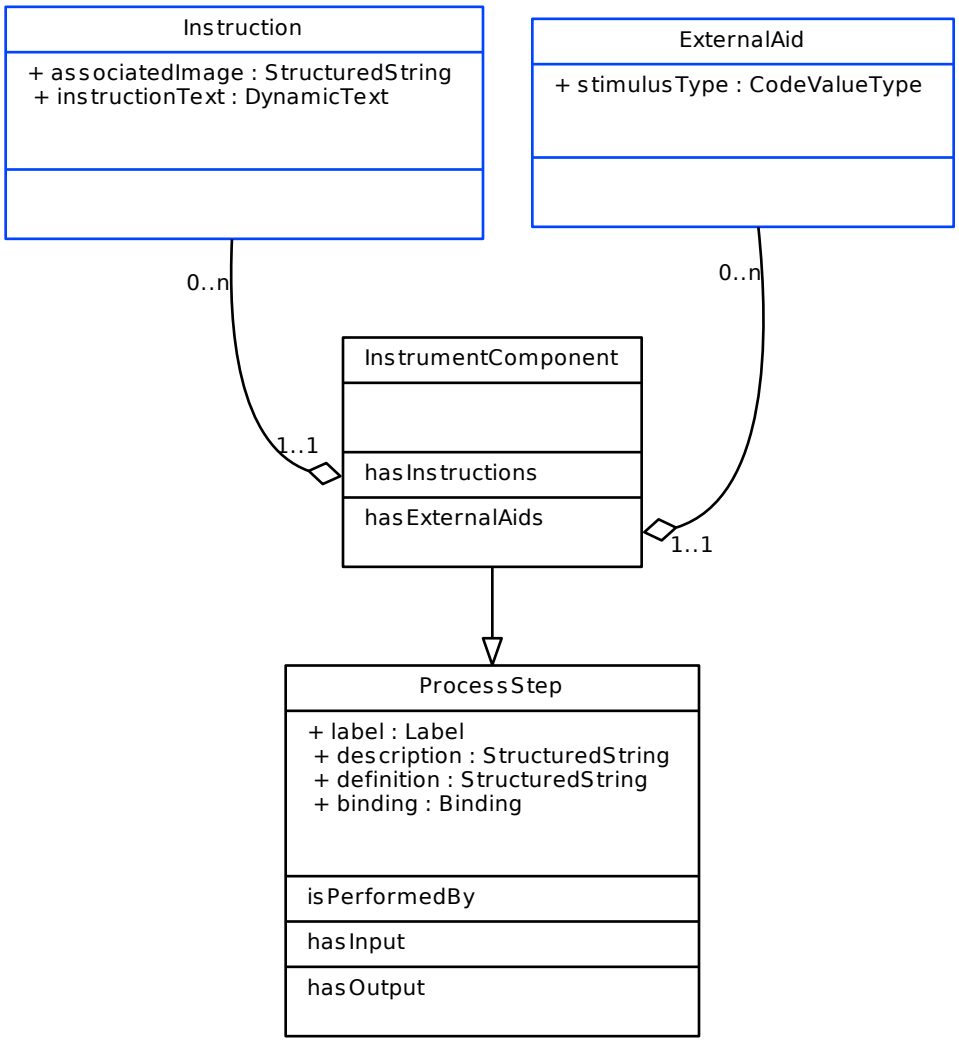
5.16.6 InstrumentComponent

InstrumentComponent is an abstract object which extends ProcessStep. The purpose of InstrumentComponent is to provide a common parent for Capture (e.g., Question, Measure), Statement, and Instructions.

Extends

ProcessStep

Graph



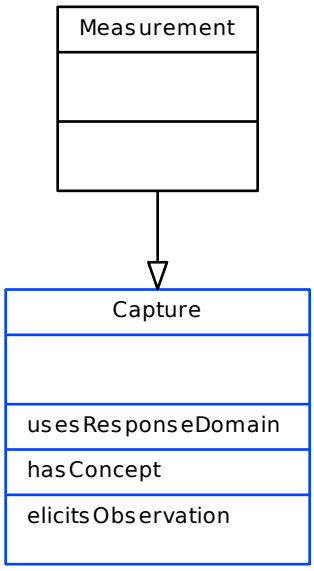
5.16.7 Measurement

Any data capture method other than Question.

Extends

Capture

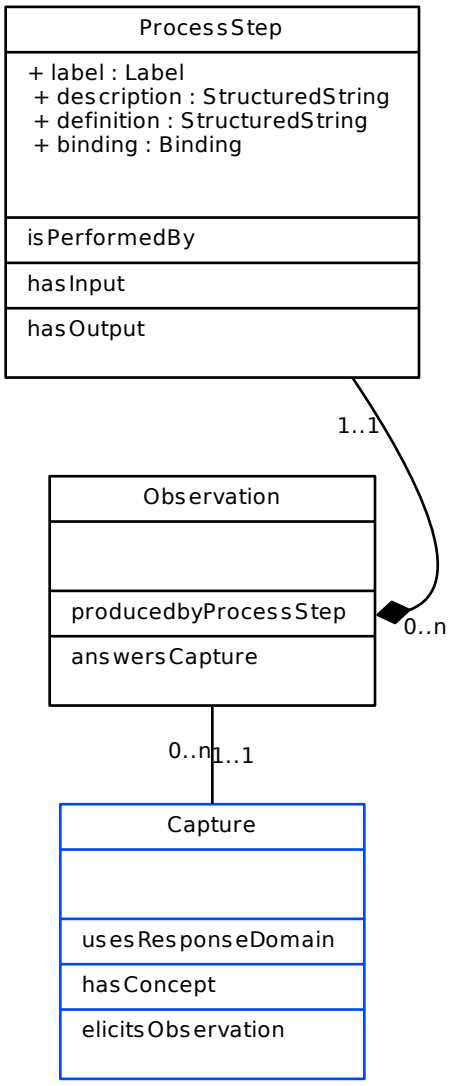
Graph



5.16.8 Observation

The result of applying a particular Capture in an Instrument to some experimental Unit

Graph



5.16.9 Question

A query of a human subject

Extends

Capture

Properties

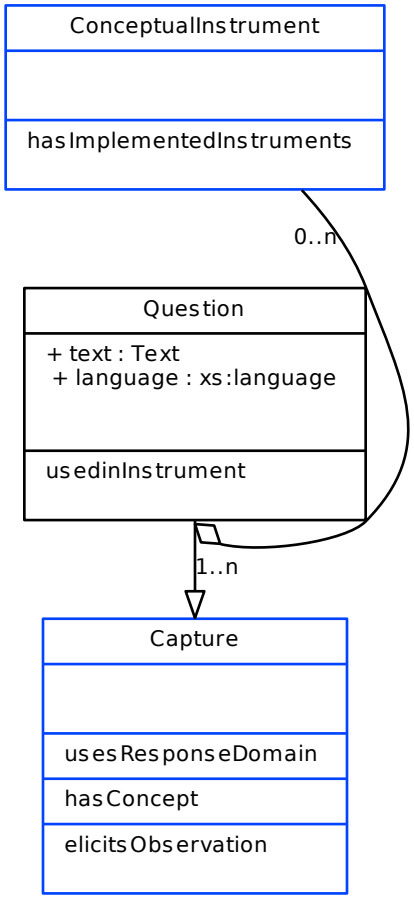
Name	Type	Cardinality
text	Text	1..1
language	xs:language	1..1

text

Literal question text

language

Graph



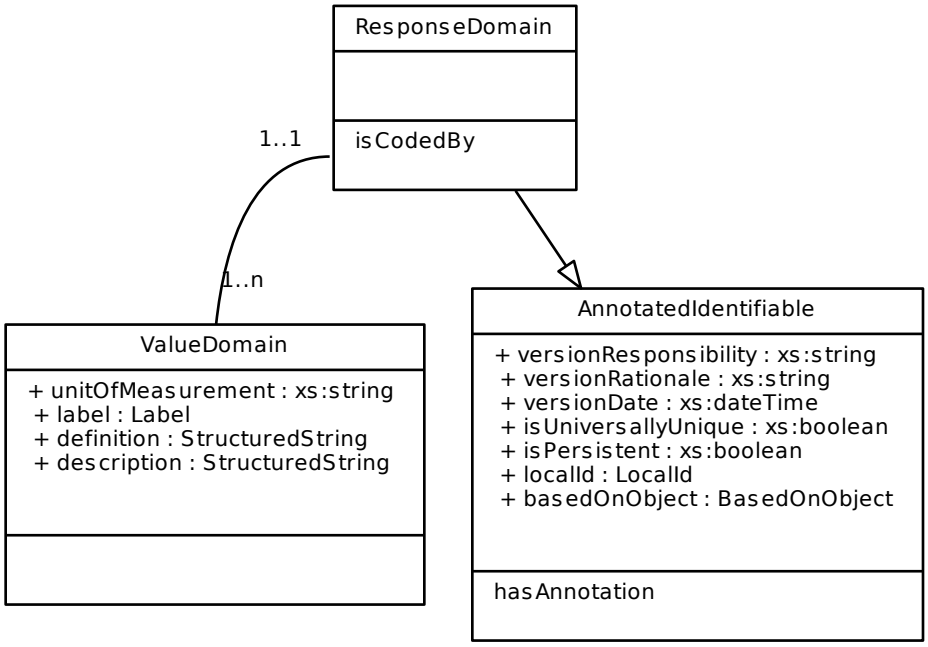
5.16.10 ResponseDomain

The possible list of values that are allowed by a Capture.

Extends

AnnotatedIdentifiable

Graph



5.16.11 Statement

A Statement is human readable text or referred material.

Extends

InstrumentComponent

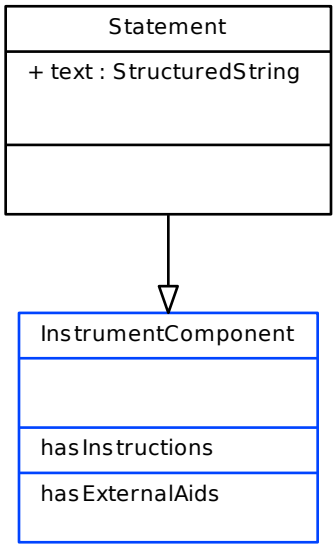
Properties

Name	Type	Cardinality
text	StructuredString	1..1

text

Structured human-readable text

Graph



5.17 Correspondences

Correspondences package Contents

5.18 CoreProcess

CoreProcess model for DDI 4. The CoreProcess contains a set of objects that serve as the basis for describing specific processes using DDI object. It is intended to be specialized to support specific applications. Some examples of its use can be seen in DataCapture, HistoricalProcess, and PrescriptiveProcess. Contents

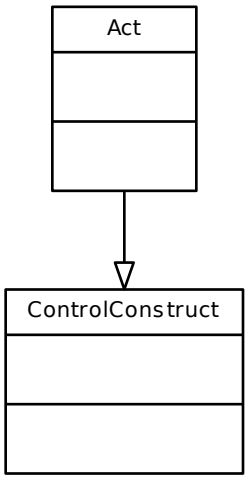
5.18.1 Act

An Act is a type of ControlConstruct. An Act has many subtypes including an Instruction, a Question, an Instrument and a StudyUnit. Both Acts and ControlConstructs are triggered when the conditions of a ControlConstruct are met.

Extends

ControlConstruct

Graph



5.18.2 Binding

A structure used to bind the content of a parameter declared as the source to a parameter declared as the target. For example, binding the output of a question to the input of a generation instruction. Question A has an OutParameter X. Generation Instruction has an InParameter Y used in the recode instruction. Binding defines the content of InParameter Y to be whatever is provided by OutParameter X for use in the calculation of the recode.

Properties

Name	Type	Cardinality
sourceParameter	xs:string	1..1
targetParameter	xs:string	1..1

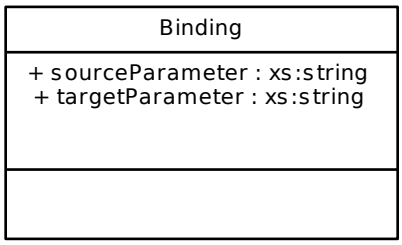
sourceParameter

A structure used to bind the content of a parameter declared as the source to a parameter declared as the target. For example, binding the output of a question to the input of a generation instruction. Question A has an OutParameter X. Generation Instruction has an InParameter Y used in the recode instruction. Binding defines the content of InParameter Y to be whatever is provided by OutParameter X for use in the calculation of the recode.[Referenced object not explicit]

targetParameter

A structure used to bind the content of a parameter declared as the source to a parameter declared as the target. For example, binding the output of a question to the input of a generation instruction. Question A has an OutParameter X. Generation Instruction has an InParameter Y used in the recode instruction. Binding defines the content of InParameter Y to be whatever is provided by OutParameter X for use in the calculation of the recode.[Referenced object not explicit]

Graph

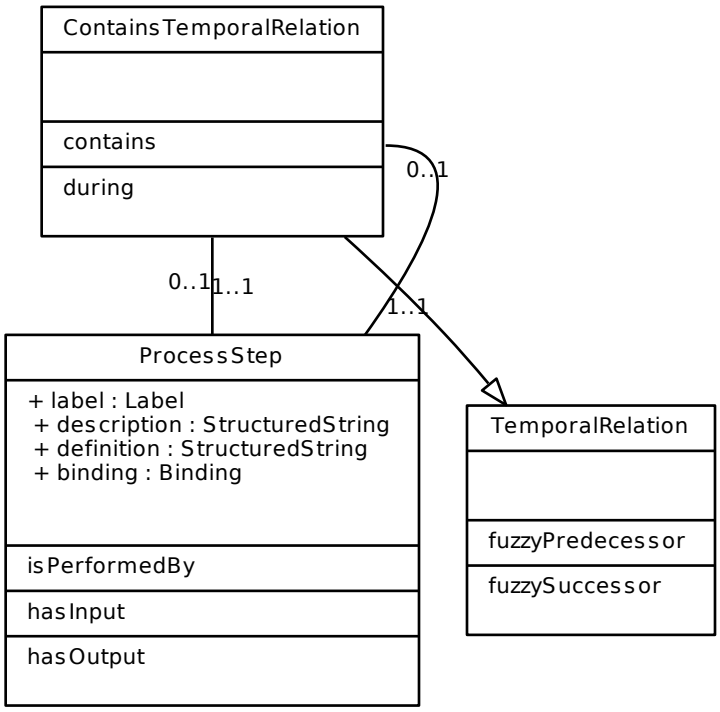


5.18.3 ContainsTemporalRelation

Extends

TemporalRelation

Graph



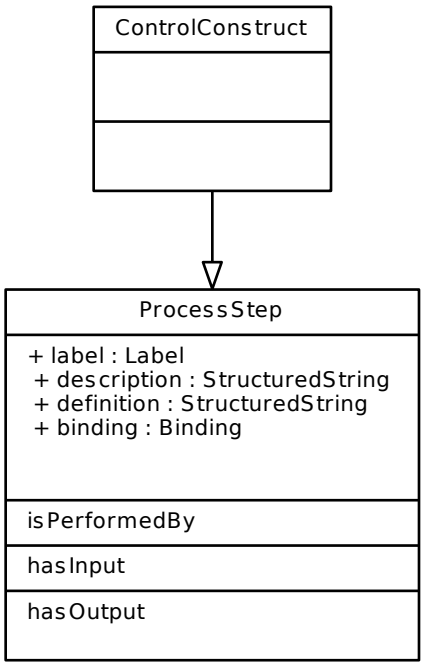
5.18.4 ControlConstruct

A ControlConstruct is used in the definition of the sequence of execution of process steps.

Extends

ProcessStep

Graph

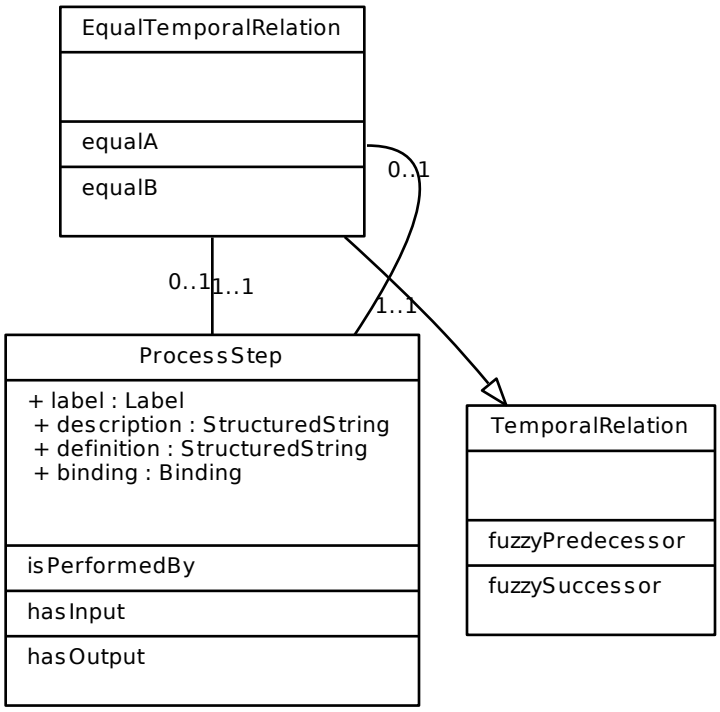


5.18.5 EqualTemporalRelation

Extends

TemporalRelation

Graph

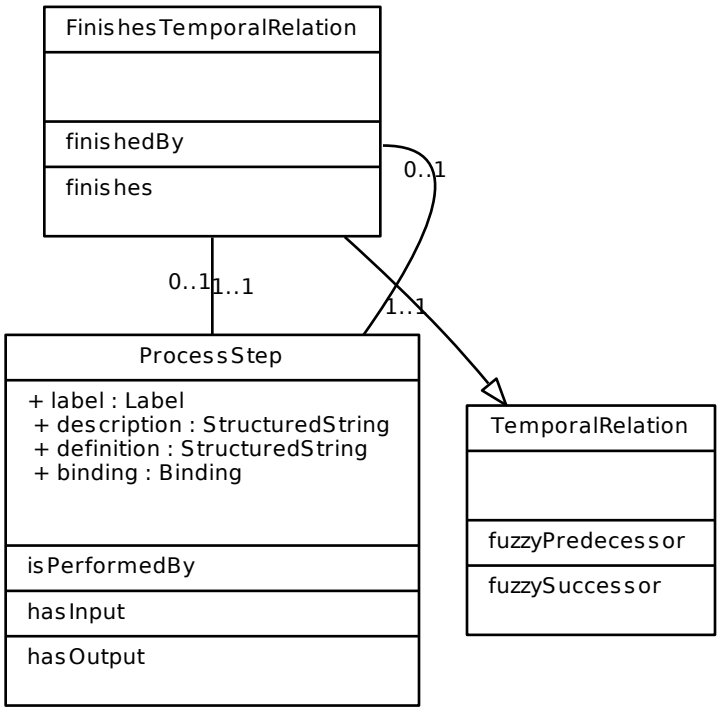


5.18.6 FinishesTemporalRelation

Extends

TemporalRelation

Graph



5.18.7 IfThenElse

IfThenElse describes an if-then-else decision type of control construct. IF the stated condition is met, the THEN clause is trigged, otherwise the ELSE clause is triggered.

Extends

ControlConstruct

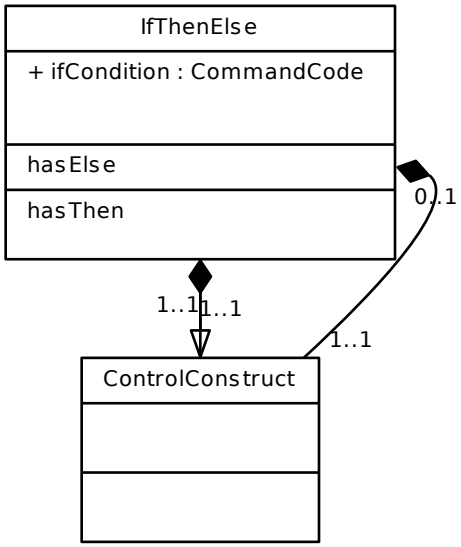
Properties

Name	Type	Cardinality
ifCondition	CommandCode	1..1

ifCondition

The condition which must be met to trigger the Then clause, expressed as a CommandCode. The condition is an expression in the programming language used in the instrument.

Graph



5.18.8 Input

Input to a process step, either a type of an object or an instance.

Extends

Parameter

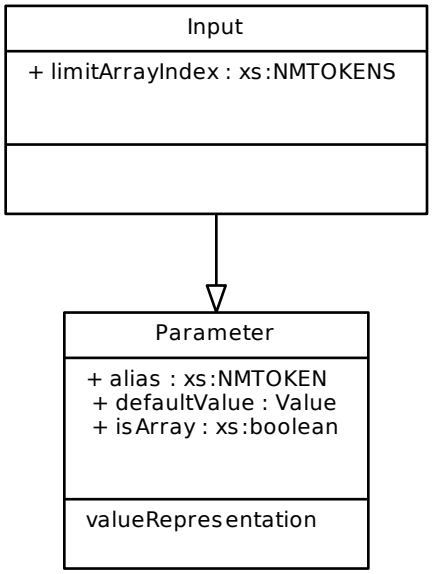
Properties

Name	Type	Cardinality
limitArrayIndex	xs:NMTOKENS	0..1

limitArrayIndex

When the Input represents an array of items, this attribute specifies the index identification of the items within the zero-based array which should be treated as input parameters. If not specified, the full array is treated as the input parameter.

Graph



5.18.9 Loop

Describes an action which loops until a limiting condition is met.

Extends

ControlConstruct

Properties

Name	Type	Cardinality
initialValue	xs:integer	0..1
loopWhile	CommandCode	0..1
stepValue	xs:integer	0..1

initialValue

The command used to set the initial value for the process. Could be a simple value.

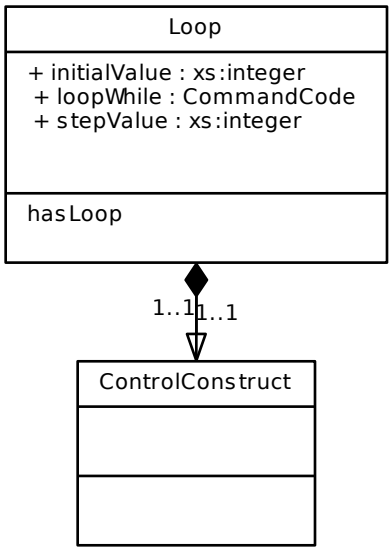
loopWhile

The command used to determine whether the "LoopWhile" condition is met.

stepValue

The command used to set the incremental or step value for the process. Could be a simple value.

Graph

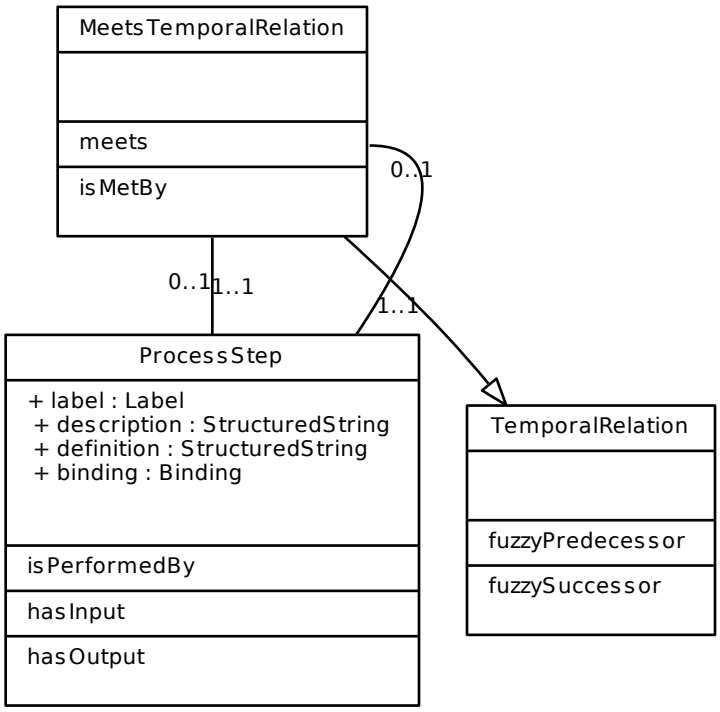


5.18.10 MeetsTemporalRelation

Extends

TemporalRelation

Graph



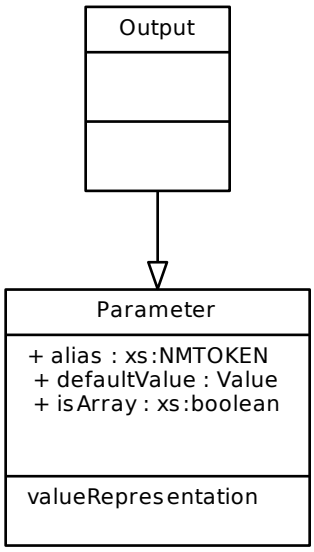
5.18.11 Output

Output to a process step, either a type of an object or an instance.

Extends

Parameter

Graph

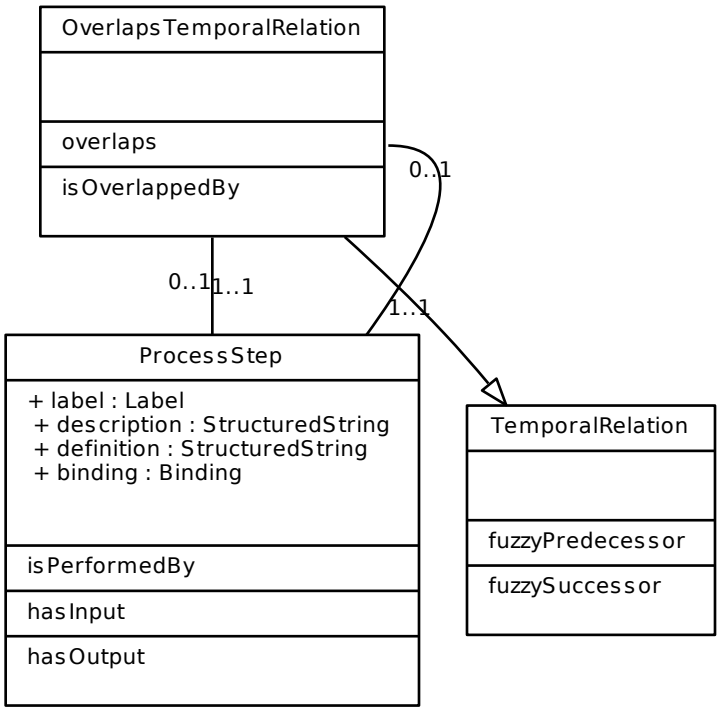


5.18.12 OverlapsTemporalRelation

Extends

TemporalRelation

Graph

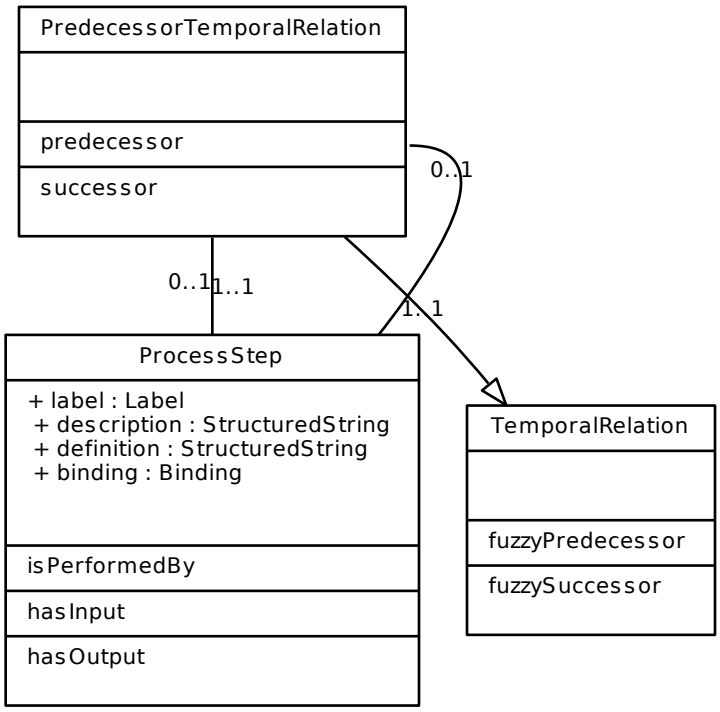


5.18.13 PredecessorTemporalRelation

Extends

TemporalRelation

Graph



5.18.14 ProcessStep

Work package performed by a service to transform inputs to outputs considering rules as defined in the control construct.

Extends

AnnotatedIdentifiable

Properties

Name	Type	Cardinality
label	Label	0..1
description	StructuredString	0..1
definition	StructuredString	0..1
binding	Binding	0..1

label

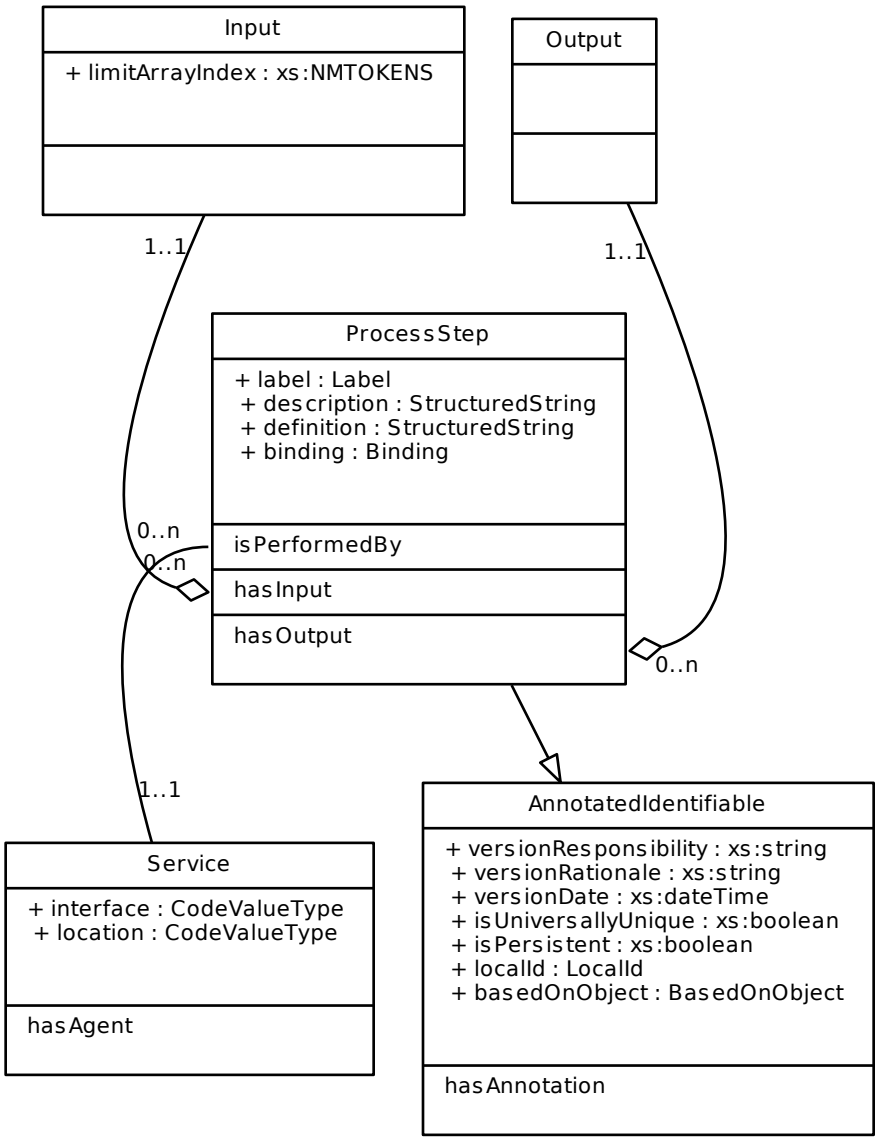
description

definition

binding

A structure used to bind the content of a parameter declared as the source to a parameter declared as the target. The binding may be defined as part of the ProcessStep in which case it is called “in line”. The binding may also be defined by a ProcessStep user such as an Instance Variable which might get its value from the ProcessStep. In this case we say the ProcessStep or a set of ProcessSteps that form a processing pipeline is reusable and “declarative” only.

Graph



5.18.15 RepeatUntil

Specifies a ControlConstruct to be repeated until a specified condition is met. Before each iteration the condition is tested. When the condition is met, control passes back to the containing control construct.

Extends

ControlConstruct

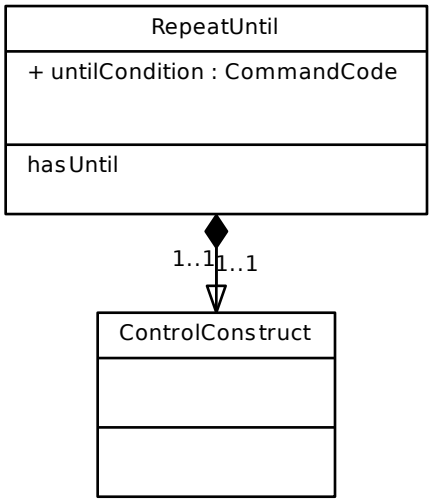
Properties

Name	Type	Cardinality
untilCondition	CommandCode	1..1

untilCondition

Information on the command used to determine whether the "Until" condition is met.

Graph



5.18.16 RepeatWhile

Specifies a ControlConstruct to be repeated while a specified condition is met. Before each iteration the condition is tested. When the condition is not met, control passes back to the containing control construct.

Extends

ControlConstruct

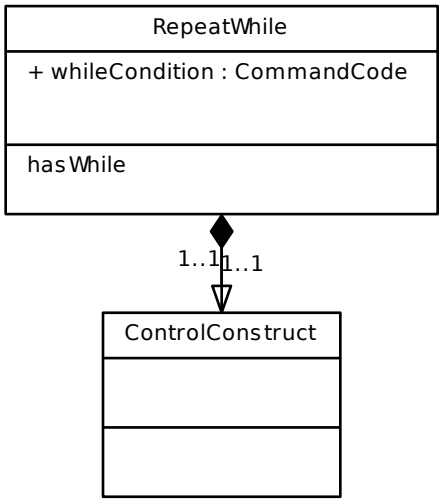
Properties

Name	Type	Cardinality
whileCondition	CommandCode	1..1

whileCondition

Information on the command used to determine whether the "While" condition is met.

Graph



5.18.17 Sequence

Provides a sequence order for operations expressed as control constructs. The sequence can be typed to support local processing or classification flags and alternate sequencing instructions (such as randomize for each respondent).

Extends

ControlConstruct

Properties

Name	Type	Cardinality
typeOfSequence	CodeValueType	0..n
constructSequence	SpecificSequence	0..1

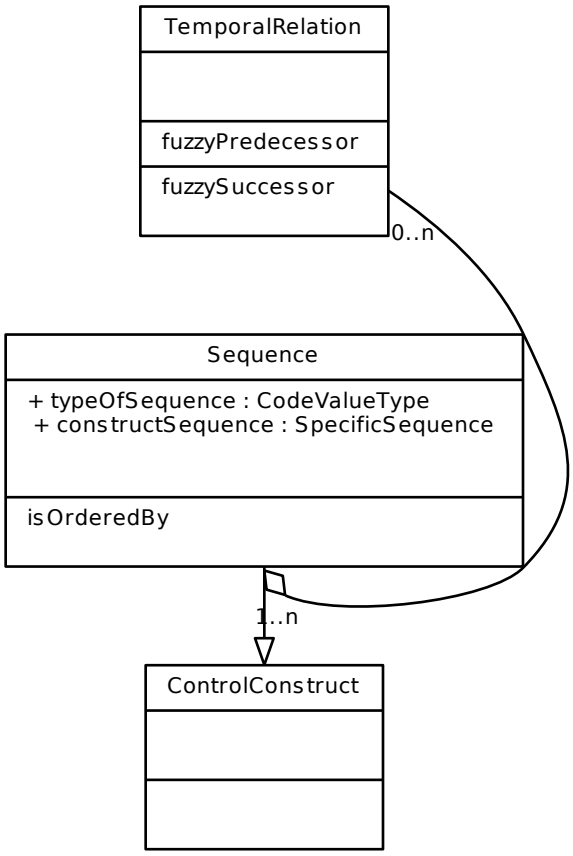
typeOfSequence

Provides the ability to "type" a sequence for classification or processing purposes. Supports the use of an external controlled vocabulary.

constructSequence

Describes alternate ordering for different cases using the SpecificSequence structure. If you set the sequence to anything other than order of appearance the only allowable children are QuestionConstruct or Sequence. Contents must be randomizable.

Graph



5.18.18 Service

A means of performing a Business Function (an ability that an organization possesses, typically expressed in general and high level terms and requiring a combination of organization, people, processes and technology to achieve).

(source: GSIM)

Extends

ProcessStep

Properties

Name	Type	Cardinality
interface	CodeValueType	0..1
location	CodeValueType	0..1

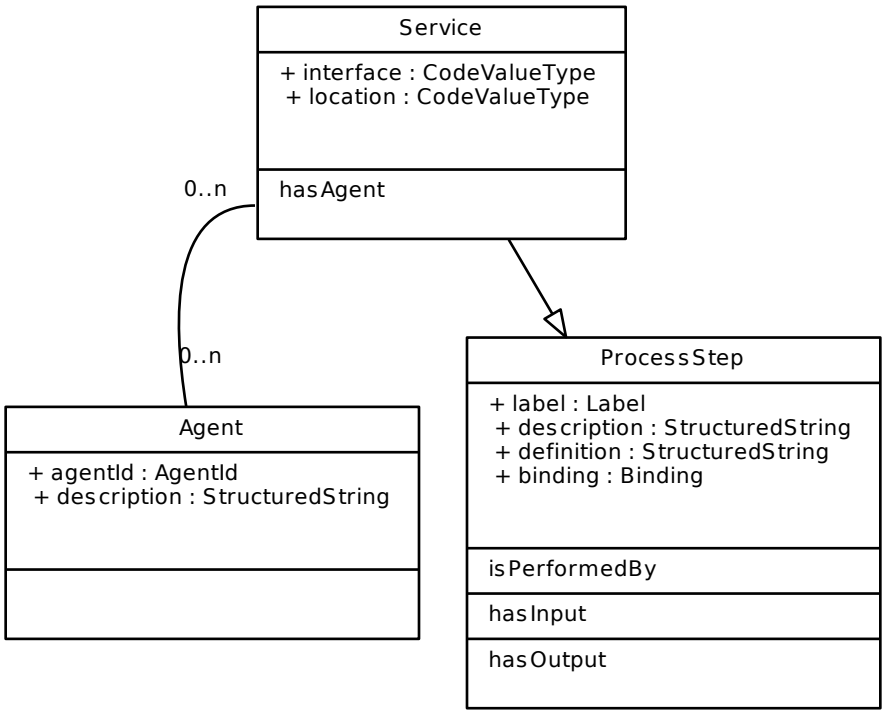
interface

Specifies how to communicate with the service.

location

Specifies where the service can be accessed.

Graph

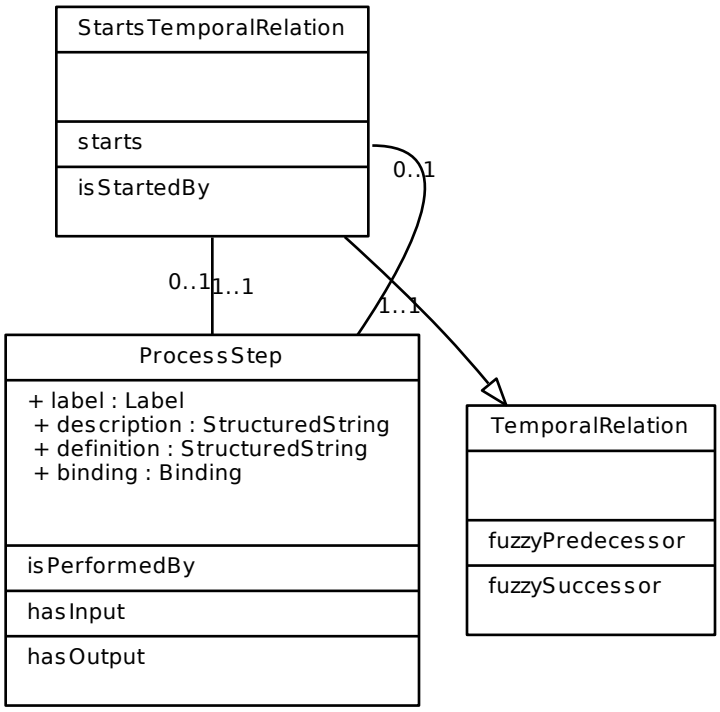


5.18.19 StartsTemporalRelation

Extends

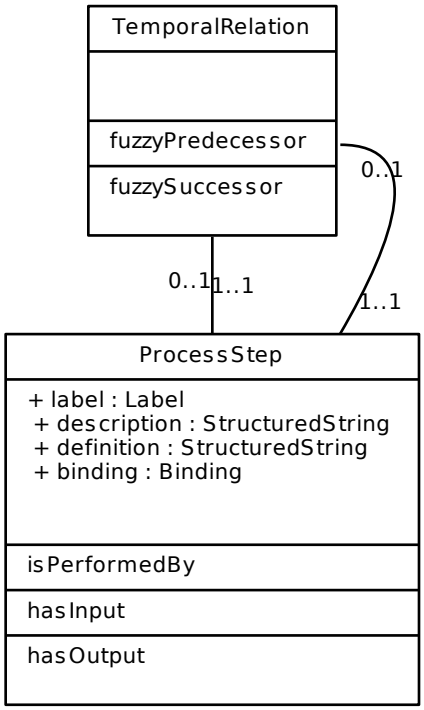
TemporalRelation

Graph


StartsTemporalRelation
starts
isStartedBy
0..1
0..1
1..1
1..1
ProcessStep
+ label : Label
+ description : StructuredString
+ definition : StructuredString
+ binding : Binding
isPerformedBy
hasInput
hasOutput
TemporalRelation
fuzzyPredecessor
fuzzySuccessor

5.18.20 TemporalRelation

Graph



5.19 Agents

Agents Package covers the description of Organizations, Individuals, and Machines (non-human) and their relationships over time. Agents are related to by processes, annotation content, and other locations where individuals, organizations, or machines are involved with the activities covered by DDI metadata. Contents

5.19.1 Agent

An actor that performs a role in relation to a process.

Extends

AnnotatedIdentifiable

Properties

Name	Type	Cardinality
agentId	AgentId	0..n
description	StructuredString	0..1

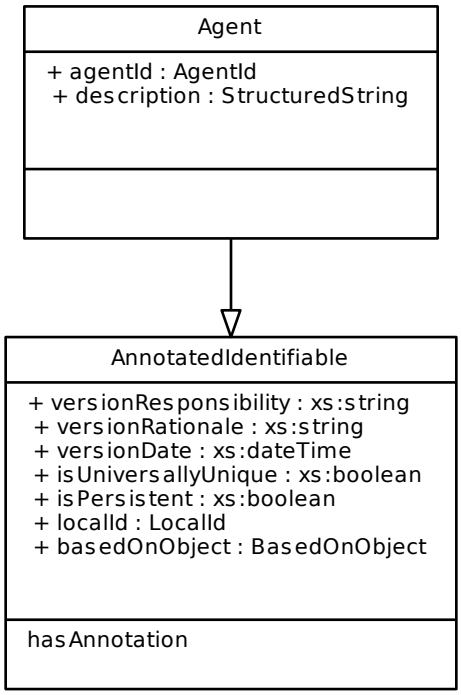
agentId

An identifier within a specified system for specifying an agent

description

Multilingual description allowing for internal formatting using XHTML tags.

Graph



5.19.2 AuthorizationSource

Identifies the authorizing agency and allows for the full text of the authorization (law, regulation, or other form of authorization).

Extends

AnnotatedIdentifiable

Properties

Name	Type	Cardinality
statementOfAuthorization	StructuredString	0..1
legalMandate	InternationalString	0..1
authorizationDate	xs:dateTime	0..1
description	StructuredString	0..1

statementOfAuthorization

Text of the authorization (law, mandate, approved business case).

legalMandate

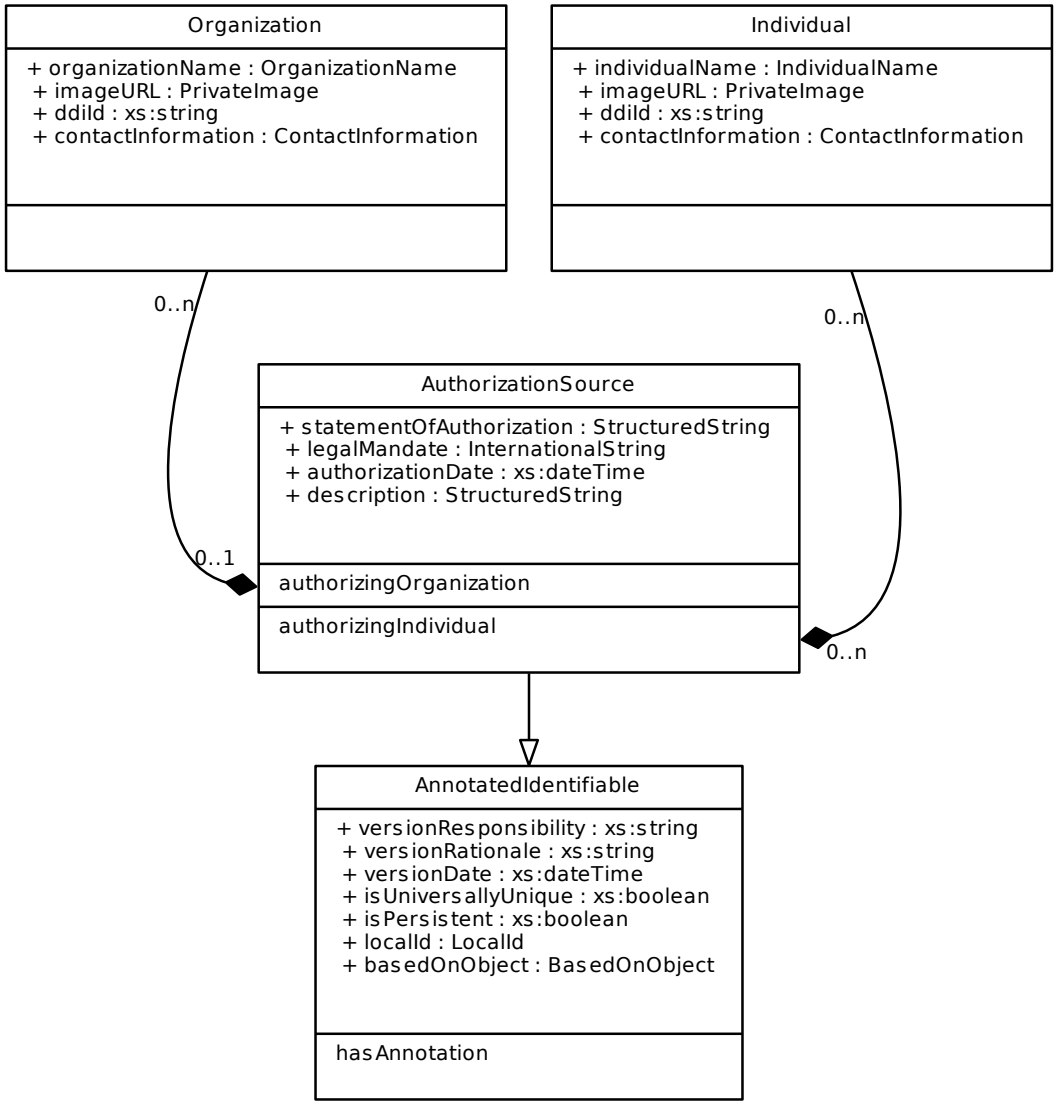
Provide a legal citation to a law authorizing the study/data collection. For example, a legal citation for a law authorizing a country's census.

authorizationDate

Identifies the date of Authorization.

description

Graph



5.19.3 Individual

A person who acts, or is designated to act towards a specific purpose.

Extends

Agent

Properties

Name	Type	Cardinality
individualName	IndividualName	1..n
imageURL	PrivateImage	0..n
ddiId	xs:string	0..n
contactInformation	ContactInformation	0..1

individualName

The name of an individual broken out into its component parts of prefix, first/given name, middle name, last/family/surname, and suffix.

imageURL

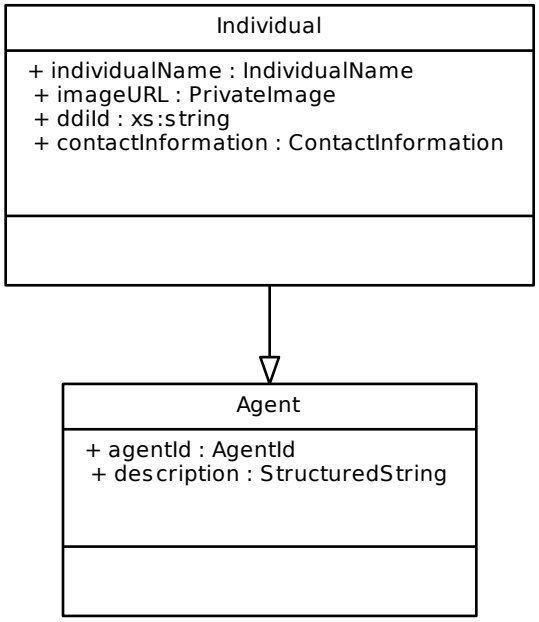
The URL of an image of the individual.

ddiId

The agency identifier of the individual according to the DDI Alliance agent registry.

contactInformation

Graph



5.19.4 Machine

Mechanism or computer program used to implement a process.

Extends

Agent

Properties

Name	Type	Cardinality
typeOfMachine	CodeValueType	0..1
machineName	Name	0..1
accessLocation	AccessLocation	0..1
function	CodeValueType	0..n
interface	CodeValueType	0..n
imageURL	PrivateImage	0..n
ownerOperatorContact	ContactInformation	0..1

typeOfMachine

The kind of machine used - software, web service, physical machine, from a controlled vocabulary

machineName

The name of the machine

accessLocation

The locations where the machine can be access

function

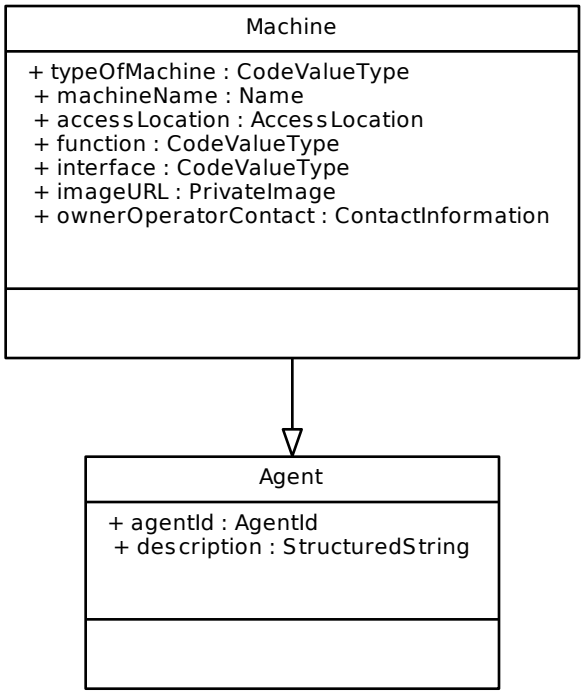
The function of the machine

interface

imageURL

ownerOperatorContact

Graph



5.19.5 Organization

A framework of authority designated to act toward some purpose.

Extends

Agent

Properties

Name	Type	Cardinality
organizationName	OrganizationName	1..n
imageURL	PrivateImage	0..n
ddiId	xs:string	0..n
contactInformation	ContactInformation	0..1

organizationName

Names by which the organization is known.

imageURL

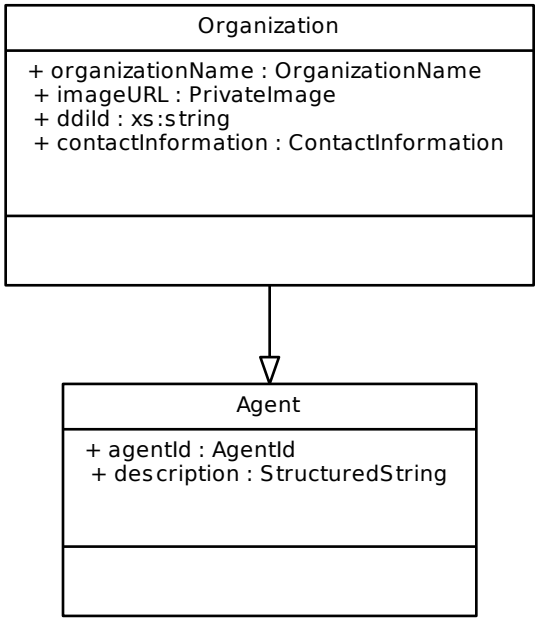
The URL of an image of the organization.

ddiId

The agency identifier of the organization as registered at the DDI Alliance register.

contactInformation

Graph



5.19.6 Relation

Describes the relationship between any two organizations or individuals, or an individual and an organization. This is a pairwise relationship and relationships may be unidirectional. Identifies the Source organization or individual and the Target organization or individual, describes the relationship, provides a keyword to classify the relationship, provides and effective period for the relationship, allows for addition information to be provided, and can contain a privacy specification.

Extends

AnnotatedIdentifiable

Properties

Name	Type	Cardinality
description	StructuredString	0..1
effectivePeriod	Date	0..n
privacy	CodeValueType	0..1
typeOfRelationship	InternationalCodeValueType	0..1

description

A description of the relationship. May be expressed in multiple languages and supports the use of structured content.

effectivePeriod

Time period during which this relationship is valid.

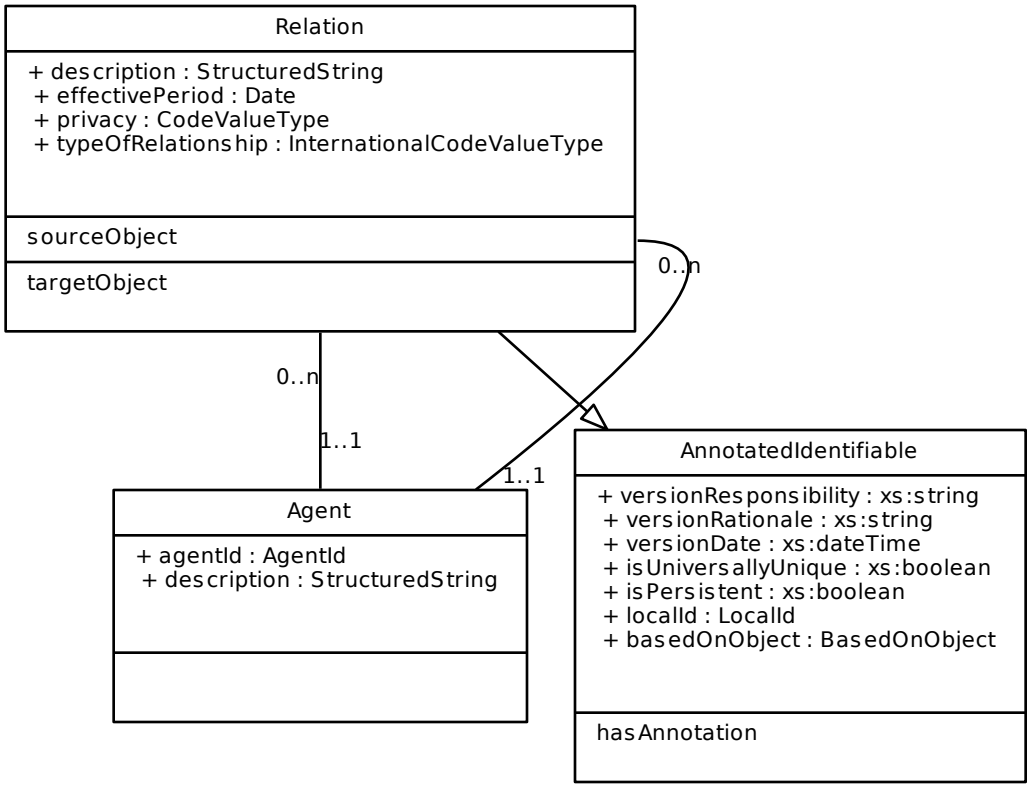
privacy

Specifies the level of privacy for the relationship specification as public, restricted, or private. Supports the use of an external controlled vocabulary.

typeOfRelationship

A brief textual identification of the type of relation. Supports the use of an external controlled vocabulary.

Graph



CHAPTER 6

Glossary

Abstract class In programming languages, an abstract type is a type in a nominative type system which cannot be instantiated directly. Its only purpose is for other classes to extend (see http://en.wikipedia.org/wiki/Abstract_type).

Binding Data binding is a way to un/serialize objects across programs, languages, and platforms. The structure and the data remain consistent and coherent throughout the journey, and no custom formats or parsing are required (see http://en.wikipedia.org/wiki/XML_data_binding).

Canonical Conforming to a general rule or acceptable procedure (see http://www.merriam-webster.com/dictionary/canonical).

Codebook A codebook is a type of document used for gathering and storing codes. Originally codebooks were often literally books, but today codebook is a byword for the complete record of a series of codes, regardless of physical format (see http://en.wikipedia.org/wiki/Codebook). DDI Codebook is the development line of the DDI specification that reflects the content of codebooks.

Class In the DDI model, as in software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects, (see https://en.wikipedia.org/wiki/Class_diagram).

Content Capture Content capture refers to the process of harvesting content from other sources. For DDI 4 development, content is being captured using the Drupal content management system to permit machine-processing.

Content Modeler One of the group of people determining the requirements for a Functional View and then identifying the set of objects needed to meet those requirements. In this process they may also need to describe new objects.

Data Modeler These people work with Content Modelers to insure that new objects are consistent with the objects accepted into the approved model. Data Modelers also arrange objects into the final namespace structure of the published model.

DDI The Data Documentation Initiative (DDI) is an effort to create an international standard for describing data from the social, behavioral, and economic sciences. The DDI metadata specification now supports the entire research data life cycle. DDI metadata accompanies and enables data conceptualization, collection, processing, distribution, discovery, analysis, repurposing, and archiving.

Drupal Drupal is a free and open-source content management framework written in PHP and distributed under the GNU General Public License. It is also used for knowledge management and business collaboration (see http://en.wikipedia.org/wiki/Drupal).

Enterprise Architect Enterprise Architect is a visual modeling and design tool based on the OMG UML. The platform supports: the design and construction of software systems; modeling business processes; and modeling industry based domains. It is used by businesses and organizations to not only model the architecture of their systems, but to process the implementation of these models across the full application development life-cycle (see http://en.wikipedia.org/wiki/Enterprise_Architect_(Visual_Modeling_Platform).

Extended Primitive An extended data type is a user-defined definition of a primitive data type. The following primitive data types can be extended: boolean, integer, real, string, date and container (see http://www.axaptapedia.com/Extended_Data_Types).

Extension Extension is the inheritance of one object's properties and relationships from another object. It also has a semantic relationship – an extending object provides a specialized use of the extended object. Extensions are used within the DDI-published packages to provide relationships between objects as they increase in complexity to meet increasingly complex functionality. Thus, a "simple" version of a questionnaire object might be extended into a more complex object, describing a more complex questionnaire.

Framework The basic structure of something; a set of ideas or facts that provide support for something; a supporting structure; a structural frame (see http://www.merriam-webster.com/dictionary/framework).

Functional View In DDI 4, a functional view identifies a set of objects that are needed to perform a specific task. It primarily consists of a set of references to specific versions of objects. Views are the method used to restrict the portions of the model that are used, and as such they function very much like DDI profiles in DDI 3.*.

Identification In the DDI specifications, each object is uniquely identified.

Instantiate To create an object of a specific class (see http://en.wiktionary.org/wiki/instantiate).

Library The Object Library for DDI 4 encompasses the entire DDI 4.0 model, but without any specific schemas or vocabularies for Functional Views. Objects contain primitives and extended primitives and are the building blocks used to construct the Functional Views. Objects are organized into packages in the Library.

Lifecycle The research data lifecycle is a set of processes that begins at study inception and progresses through data collection, data publication, data archiving, and beyond. DDI created a lifecycle model in 2004 to describe this flow (see http://www.ddialliance.org/system/files/Concept-Model-WD.pdf).

Management package DDI 4 packages containing library constructs – primitives, extended primitives, objects, and functional views – which are organized thematically.

Metadata Data about data.

Modeling The representation, often mathematical, of a process, concept, or operation of a system, often implemented by a computer program. For DDI 4 development, we are using the Universal Modeling Language or UML to model the specification.

Namespace A grouping of objects that allows for objects with the same name to be differentiated. The full name of an object is a combination of its namespace and its name within the namespace.

Ontology A formal representation of knowledge (see http://en.wikipedia.org/wiki/Ontology_%28information_science%29).

OWL Web Ontology Language is a semantic markup language for publishing and sharing ontologies on the World Wide Web (see http://www.w3.org/TR/owl-ref/).

Platform A pre-existing environment in which to represent data and metadata (see http://en.wikipedia.org/wiki/Computing_platform).

Primitive A basic type is a data type provided by a programming language as a basic building block. Most programming languages allow more complicated composite types to be recursively constructed starting from basic types.

RDF The Resource Description Framework (RDF) is a family of World Wide Web Consortium (W3C) specifications[1] originally designed as a metadata data model. It has come to be used as a general method for conceptual description or modeling of information that is implemented in web resources, using a variety of syntax notations and data serialization formats. It is also used in knowledge management applications (see http://www.w3.org/RDF/).

Serialization Transformation of some structure into a specific representation (see http://en.wikipedia.org/wiki/Serialization).

Sprint An activity where a group of people comes together to work exclusively on some project.

Study An activity producing data.

UML The Unified Modeling Language (see http://www.uml.org/).

UML Class Model A model describing objects and their relationships in the Unified Modeling Language (see http://www.uml.org/).

UML Package A grouping of classes in the Library (see namespace).

URI Uniform Resource Identifier, a unique string of characters used to identify an object (see http://en.wikipedia.org/wiki/Uniform_resource_identifier).

Versioning The assignment of some ordered attribute to objects, In the DDI model whenever a change is made to an approved object it must be given a new version. Objects with the same name and different versions are considered to be separate objects.

Workflow A formalized set of processes carried out in a defined sequence (for more detail see http://en.wikipedia.org/wiki/Workflow).

XMI An Object Management Group XML standard for exchanging metadata information (see http://www.omg.org/spec/XMI/).

xml Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format which is both human-readable and machine-readable.

XSD An XML Schema definition (see http://www.w3.org/XML/Schema.html).

CHAPTER 7

Guide to editing

7.1 Links

to make an external link just type it http://www.ddialliance.org

For a link a text

```
`Link text <http://example.com/>`_
```

To reference a section anywhere in the documentation e.g. *Question*

```
:ref:`Question`
```

7.2 Strong

To mark something as **bold text**

```
**bold text**
```

7.3 Italic

To mark something as *italic text*

```
*italic text*
```

7.4 Code

To highlight some sourcecode

```
.. code-block:: xml

   <some-xml></some-xml>
```

results in

```
<some-xml></some-xml>
```

7.5 Headings

Headings are created by adding by adding underlining characters

```
This is a  section heading
==========================
```

- # with overline, for parts
- * with overline, for chapters
- =, for sections
- -, for subsections
- ^, for subsubsections
- “, for paragraphs

7.6 Images

To link a image

```
.. image:: ddi-logo-mini.png
```

7.7 Lists

To do a list

```
* This is a bulleted list.
* This is item two

  * this is a sub item

1. This is a numbered list.
2. It has two items too.
```

Results in

- This is a bulleted list.
- This is item two
  - this is a sub item

1. This is a numbered list.
2. It has two items too.

7.8 Table

To do a table

```
=====  =====  =======
Col 1  Col 2  Note
=====  =====  =======
1      False  cats
2      False  dogs
3      True   frame
4      True   from
=====  =====  =======
```

Col 1	Col 2	Note
1	False	cats
2	False	dogs
3	True	frame
4	True	from

7.9 Glossary

To do a simple definition list

```
.. glossary::

   cats
     miau is the sound of the cat.

   dogs
     voff is the sound of the dog.
```

Results in

cats miau is the sound of the cat.

dogs voff is the sound of the dog.

For the documentation we should have a single glossary se *Glossary*

The we can make glossary links by writing

```
:term:`xml`
```

and get *xml*

7.10 Notes

To add a note, warning etc.

```
.. note::

   Above is the basics of reStructuredText.
   For a complete guide visit
   `Sphinx-doc <http://sphinx-doc.org/contents.html>`_ .
```

To get

Note: Above is the basics of reStructuredText. For a complete guide visit Sphinx-doc .

Index