Viewpoint Metamodel Documentation

2/6/2023

Table of Contents

[Overview 3](#_Toc126585389)

[Intent of this metamodel 3](#_Toc126585390)

[Intent of concept modeling 3](#_Toc126585391)

[Applying the viewpoint metamodel to classes of use cases 4](#_Toc126585392)

[The subset of OWL includes: 4](#_Toc126585393)

[Concept Modeling Extensions 4](#_Toc126585394)

[Defining Composite Concepts 4](#_Toc126585395)

[Representing data schema 5](#_Toc126585396)

[Data represents concept 5](#_Toc126585397)

[Stakeholder Viewpoints 5](#_Toc126585398)

[Extension Mechanism 5](#_Toc126585399)

[Examples 5](#_Toc126585400)

[Implementation 5](#_Toc126585401)

[Viewpoint Metamodel 6](#_Toc126585402)

[Diagram: Viewpoint Metamodel 6](#_Toc126585403)

[Class Constrained Property 6](#_Toc126585404)

[Class Constraint 7](#_Toc126585405)

[Class Context 7](#_Toc126585406)

[Class Data Attribute 7](#_Toc126585407)

[Class Domain 7](#_Toc126585408)

[Class End 7](#_Toc126585409)

[Class Entity 8](#_Toc126585410)

[Class Enumeration 8](#_Toc126585411)

[Class Equivalent Constraint 8](#_Toc126585412)

[Class Facet 8](#_Toc126585413)

[Class Package 9](#_Toc126585414)

[Class Path Step 9](#_Toc126585415)

[Class Record 10](#_Toc126585416)

[Class Relationship 10](#_Toc126585417)

[Class Representation Constraint 10](#_Toc126585418)

[Class Restriction 10](#_Toc126585419)

[Class Value 10](#_Toc126585420)

[Class Value Constraint 11](#_Toc126585421)

[Class Viewpoint 11](#_Toc126585422)

[Class Visibility Constraint 11](#_Toc126585423)

# Overview

The viewpoint metamodel is designed to satisfy five categories of related use cases

1. Capturing concept models at various levels of fidelity from RDFS to full concept modeling.
2. Defining the meaning of composite concepts to ground data elements, in particular catalogs of attributes such as MDMI semantic elements and data dictionaries.
3. Providing for user centric viewpoints of models that simplify and prioritize parts of the model for better stakeholder comprehension.
4. Representation of the basic structure of data elements to provide for mapping
5. Mapping of data structures and mappings (such as MDMI) to their meaning

Every effort has been made to both separate concerns and technologies for the above use cases while simplifying the metamodel as much as possible. The metamodel is defined in Magicdraw-CCM and generates OW to ground the semantics, but it is intended to be self-defining.

Note that tis overview section is authored” where as the “Viewpoint Metamodel” section is generated from the Magicdraw model.

## Intent of this metamodel

The viewpoint metamodel is based on [W3C RDF Schema 1.1](https://www.w3.org/TR/rdf-schema/) but extends the capability with additional capabilities for the above use cases. These extensions include capabilities to represent a subset of OWL, but the representation and intent of the model is different from OWL as described below.

### Intent of concept modeling

The intent of the viewpoint metamodel and the meta-concepts defined for concept modeling differ from that of OWL and first-order logic. In particular, the intent of this metamodel is defining the meaning and leveraging those definitions for stakeholder understanding and the integration of information across different applications, domains, and technologies. Unlike OWL and first-order logic, it is not a goal of this model to infer new knowledge. It is a goal to integrate concepts and inferences that may have been derived from traditional ontologies.

There is typically a divide between “data” and “concepts”. Data provides representations of information about things in our world for a particular purpose – typically an application build for a particular stakeholder group. Data schema tell us what is required for those purposes and how it is stored and/or communicated – as such data schema play a crucial role. The issue with schema (even “enterprise logical models”, is that they, by necessity, provide a limited and localized structure and terminology that makes integration with other data schema difficult.

Concepts define meaning, semantics. How we understand, classify, and quantify the world. Concepts are not specific to a particular purpose and many concepts span stakeholder groups. The terminology used for concepts is more general, not localized to the purpose. Concepts alone do not provide for efficient or understandable data structures do to their generality and abstractions.

The viewpoint metamodel is intended to bridge the gaps between stakeholder’s viewpoints, data, and concepts.

## Applying the viewpoint metamodel to classes of use cases

The foundation of the metamodel is the capability to capture “meaning” and business concepts. This is accomplished with a subset of OWL and a set of extensions.

### The subset of OWL includes:

* All capabilities of RDFS including: subClassOf, subPropertyOf, Domain, Range
* Cardinality of properties
* Equivalence
* Keys
* All values from (classes and individuals)
* Import
* Property restrictions for the above supported constraints

### Concept Modeling Extensions

Besides the OWL subset, concept modeling capabilities are included based on the draft “Semantic Modeling for Information Federation” specification. Note that use of these extensions is optional.

* Context - an abstract thing with a set of constraints, capabilities, and characteristics that must be satisfies for anything contextualized by that context.
* Entities – things that have an independent lifetime and identity in the world.
* Facets – things that represent contextual (temporal, special, or behavioral) aspects of things.
* Values – information without an independent identity, usually the type of a property.
* Relationships as first-class things.
* Enumerations – classes with a fixed set of members.
* Packages – “Owners” of the base definition of things.
* Domains – Context relative to an area of concern.

The above concept models bay be defined using the “Cameo Concept Modeling” extension to Magicdraw.

### Defining Composite Concepts

The relationship between data and semantic concepts if often complex and not able to be satisfied by a simple reference or “pointer”. The approach taken in this specification is to define composite, extended, or restricted concepts using the concept model that are then able to be grounded with a simple reference we call “represents” (more on this below.

The requirements for extending and restriction classes can largely be accomplished with class constraints found in OWL. A data class typically represents one or more semantic classes, perhaps with these restrictions. On the other hand, properties can be more difficult because data properties frequently represent a “path” though the concept model with various restrictions along that path – this is accomplished using the “Patch Step” element, each step in the path can traverse a property and apply constraints to those steps, this is similar to but goes beyond OWL “property chains”. Once the composite property is defined it can be represented by data attributes.

### Representing data schema

Data schema can, of course, be quite complex and define implementations for specific technologies, we do not attempt to replicate this information but to reference it. We limit our representation of data to “Records”, “Data Attributes”, Keys, and “Foreign Keys”. We import this subset of schema into the model for the purpose of mapping.

### Data represents concept

The fundamental theory of this specification is that “data represents concepts” – that is for each data element there is one or more semantic concepts that the data is “about”, or “represents”. Once we have the composite concepts defined we can have a simple set of “represents” relationships from the data to the concepts.

### Stakeholder Viewpoints

Non-technical stakeholders must be able to understand and comment on the concepts in their domain. But, to enable the generality and abstraction required to fully capture semantics, the concept models may become to complex and may not use their preferred “localized” terminology. To provide a stakeholder friendly view we define “Viewpoints”. A viewpoint subsets, simplifies a model for a stakeholder group using their preferred terminology. As part of this simplification concepts are tagged with their “importance” for that stakeholder, this is called the “visibility”. Viewpoints also use the restrictions and paths we use for mapping data.

### Extension Mechanism

To separate concerns between the metamodel and various semantic or physical modeling paradigms, the “Archetype” extension mechanism is utilized. An architype is similar to a UML stereotype or OWL annotation. The assumption made is that an archetype is a supertype of a domain concept. E.g. “Activity” may be defined as an archetype and will “tag” all activities.

In the metamodel the “has Archetype” relationship defines the type that is the archetype for something. When defining a new archetype, the archetype “has Archetype” it’s self. The archetype may be inferred by finding an archetype in the class supertypes. In diagrams the archetype may be shown like a stereotype.

Extensions may also augment or subtype the metamodel to define additional meta-relationships. An archetype would be an instance of the augmented metatype.

# Examples

TBD

# Implementation

TBD

# Viewpoint Metamodel

The metamodel for viewpoints of ontological, conceptual, and physical models and the mappings between them.

## Diagram: Viewpoint Metamodel

1054974195.emf

1. Viewpoint Metamodel

## Class Constrained Property

An abstract kind of thing that defines constraints used by both properties and path steps.

Direct Supertypes

[Thing](#_d85fc9b2068aad8f1d625373a20313e6)

Attributes

1549810183.emf maxCardinality : [Integer](#_aeefbb09a8c456505ebb76cf8a103a03) [0..1]

The maximum number of values for a property for a given subject.

1549810183.emf minCardinality : [Integer](#_aeefbb09a8c456505ebb76cf8a103a03) [0..1]

The maximum number of values for a property for a given subject.

1549810183.emf isOrdered : [Boolean](#_6119a00b0834641b9fe3f5ae9f58237f) [0..1]

Assertion that the set of values is ordered.

## Class Constraint

A constraint specifies a condition that an instance of a thing must satisfy.

Direct Supertypes

[Resource](#_a56395373673207b30c301838b09ee77)

## Class Context

Context is an abstract thing with a set of constraints, capabilities, and characteristics that must be satisfies for anything contextualized by that context.

Direct Supertypes

[Thing](#_d85fc9b2068aad8f1d625373a20313e6)

## Class Data Attribute

A data attribute is a property of an information record, holding data about a characteristic of something. A data attribute may be defined by a semantic [property it represents.

Direct Supertypes

[Property](#_7d0cd769d25a6911af73f639cfbbf3d2)

Associations

1397781411.emf container : [Record](#_6a79db978122bea84d731bdeeb5bddc4) [0..1] *Subsets*: domain:[Class](#_e65367f1e38469d9781c300417de1342) subjectType:[Class](#_e65367f1e38469d9781c300417de1342)

THe container that defines the existence and fundamental characteristics of a resource. The container is considered the "systemn of record" about something but may not contain all the information about that thing.

1397781411.emf foreignKeyOf : [Property](#_7d0cd769d25a6911af73f639cfbbf3d2) [0..1]

A property that provides data to identify the opposite end of a relationship as used in typical E/R models.

## Class Domain

A domain is a context defined for a subject area or area of interest such as clinical health care or mortgage loans.

Direct Supertypes

[Context](#_5b6a434dc49b9ce419e4e6da686a3150)

## Class End

A property that defines one end of a relationship.

Direct Supertypes

[Property](#_7d0cd769d25a6911af73f639cfbbf3d2)

Attributes

1549810183.emf isSubject : [Boolean](#_6119a00b0834641b9fe3f5ae9f58237f) [0..1]

siSubject indicates that an end represents the subject, or "from" end of a directed relationship.

Associations

1397781411.emf inverseOf : [Property](#_7d0cd769d25a6911af73f639cfbbf3d2) [0..1]

The other end of a relationship, defined for convenience and OWL compatibility. if a relationship has two ends they will be inverse of each other.

## Class Entity

A type of thing with independent existence in the world. Things include physical-spacial/temporal and social constructs.

Direct Supertypes

[Class](#_e65367f1e38469d9781c300417de1342)

## Class Enumeration

A type that defines a fixed set of individuals, normally but not necessarily values.

Direct Supertypes

[Class](#_e65367f1e38469d9781c300417de1342), [Package](#_3d1c63367ff837f449baca6944f9128c)

Associations

1397781411.emf : [Value Constraint](#_e5dc32b665f509420d6646451010ed1c) [1..\*] *Subsets*: isConstrainedBy:[Constraint](#_bcd3dae50545ad03ddcf95af778ae822)

## Class Equivalent Constraint

An equivalent constraint requires that all hasEquivalent things have the same extent - that they classify the same set of things.

Direct Supertypes

[Constraint](#_bcd3dae50545ad03ddcf95af778ae822)

Associations

1397781411.emf hasEquivalent : [Thing](#_d85fc9b2068aad8f1d625373a20313e6) [2..\*]

One of a set of things that is equivalent based on an equivalent constraint.

## Class Facet

A contextual aspect of something else. This includes roles & phases. The type of things that may have this aspect is specified by facetOf, which is inherited by all subclasses of a facet.

A facet defined with withRespectTo has a lifetime corresponding the lifetime of the end, which in tern corresponds to the lifetime of the end's relationship. E.g. Joe is a pilot of a particular flight, while that flight is occurring.

A facet defined without withRespectTo has an arbitrary lifetime defined by factors outside of the model. E.g. Joe is a pilot, meaning Joe can fly airplanes.

Direct Supertypes

[Class](#_e65367f1e38469d9781c300417de1342)

Associations

1397781411.emf facetOf : [Class](#_e65367f1e38469d9781c300417de1342) [0..1]

The class of things that may have a particular kind of facet.

1397781411.emf withRespectTo : [End](#_034167e597a20e1f9a029822a01bbc1d) [0..1]

withRespectTo defines the and of a relationship that implies the existence of a facet.

## Class Package

A package is a structural context that contains the definition of resources.

Direct Supertypes

[Context](#_5b6a434dc49b9ce419e4e6da686a3150)

## Class Path Step

A path step is one node of a path that is one traversal - constrained step in a path, where the entire path defines a property that hasDefinition of the head of the path. Constraints are applied after traversal of the traverses property, if any.

Direct Supertypes

[Constrained Property](#_1ade95a6c0613a614022000bcde216ca)

Associations

1397781411.emf nextStep : [End](#_034167e597a20e1f9a029822a01bbc1d) [0..1]

the next step in a path, if any.

1397781411.emf traverses : [Property](#_7d0cd769d25a6911af73f639cfbbf3d2) [0..1]

The property traversed by a path step.

## Class Record

A data structure used for storing information about other things. Records may be semantically defined by specifying what they represent.

Direct Supertypes

[Class](#_e65367f1e38469d9781c300417de1342)

## Class Relationship

A relationship classifies how two things are connected via the ends of the relationship.

Note that a choice was made for relationships to be binary, however relationships may have attributes and participate as ends of other relationships.

Direct Supertypes

[Class](#_e65367f1e38469d9781c300417de1342)

Associations

1397781411.emf hasEnd : [End](#_034167e597a20e1f9a029822a01bbc1d) [1..2]

One end of a relationship.

## Class Representation Constraint

Constraint that the defining thing represents, is information about, the represented thing. The defining thing can be considered "epistemic" (about knowledge) where as the represented thing can be considered "ontological", about the world.

Direct Supertypes

[Constraint](#_bcd3dae50545ad03ddcf95af778ae822)

## Class Restriction

A restriction is a phudo property that places additional restrictions on the super-properties (subPropertyOf and redefines). Restrictions to not define an additional predicate but restrict other predicates. The URI for a restriction should not be used in a RDF triple.

Direct Supertypes

[Property](#_7d0cd769d25a6911af73f639cfbbf3d2)

## Class Value

A class if resources where their identity and contents are the same. Values do not change over time or have independent identity.Values may be primitive (e.g. numbers) or complex (e.g. a person's full name).

Direct Supertypes

[Class](#_e65367f1e38469d9781c300417de1342)

## Class Value Constraint

A Value Constraint is a constraint that requires that a class is one of the values specified by hasValue or one of the values contained by hasValueSet.

Direct Supertypes

[Constraint](#_bcd3dae50545ad03ddcf95af778ae822)

Associations

1397781411.emf hasValue : [Resource](#_a56395373673207b30c301838b09ee77) [0..\*]

one of a set of possible values for a value constraint.

1397781411.emf hasValueSet : [Package](#_3d1c63367ff837f449baca6944f9128c) [0..\*]

a package containing a set of values, any one of which may satisfy a value set.

## Class Viewpoint

A viewpoint is a context defined for a set of stakeholders.

Direct Supertypes

[Context](#_5b6a434dc49b9ce419e4e6da686a3150)

## Class Visibility Constraint

A visibility constraint specifies how relevant and important a concept is within a context.

Direct Supertypes

[Constraint](#_bcd3dae50545ad03ddcf95af778ae822)

Attributes

1549810183.emf visibility : [Integer](#_aeefbb09a8c456505ebb76cf8a103a03) [1]

A number indicating the relevance and importance of a concept within a context where 1 is the most important.

1549810183.emf defaultContextVisibility : [Integer](#_aeefbb09a8c456505ebb76cf8a103a03) [0..1]

A number indicating the relevance and importance of a set of concepts within a context where 1 is the most important.