DDI 4 Core

UML Modeling Recommendations

The UML modeling recommendations were formulated by the MRT working group in the Ottawa meeting, April 22-24, 2019.

# Overview

These UML modeling recommendations were developed by the MRT working group in the Ottawa meeting, April 22-24, 2019. They encompass many of the issues coming out of the Prototype Review, and also the documentation of open issues following the Berlin Sprint at the end of 2018. It is the intention that these recommendations, once adopted, will be implemented in the DDI 4 Core model.

This document is organized around the tasks/issues for specific features of UML to be used in the model. These recommendations will impact the resolution of issues in the Jira tracker in various ways – specific issue resolutions are not addressed in this document, but will be handled in Jira as appropriate, with possible reference to this document. Within each topic are a series of requirements, general thoughts, recommendations, and proposals, ordered according to the discussion. Affected issues are listed at the end of each section as appropriate.

This is a working document - when guidelines are formulated, they will be organized appropriately. This document is organized a record of the discussions and thinking, however.

# Topics

## Associations

### Requirements

* Associations should be used in a precise and consistent way. This applies especially for the several available UML options to qualify associations and/or document them.
* Associations are rendered in UML diagrams by connecting classes with a line. This should be used efficiently to indicate the relationship of classes and the kind of relationship.
* Restricted associations should only be used with a clear need and reasoning.

### General Thoughts

**UML note**: Associations and properties (defined by data types) have a similar meaning in UML.

### Modeling Recommendation

* Associations should be viewed as predicates between a subject and an object. The name of the association should reflect that. The direction of the association should also be in line with the predicate so that it points from the subject (class) to the object (class). Additionally, the direction can be used for versioning purposes, i.e. changes to the object will trigger a new version of the subject. For this reason, associations should always be uni-directional.

### General Thoughts

It’s important to note that uni-directionality in the conceptual model doesn’t prevent bi-directionality at the syntax representation level or software implementation level.

**Background** (UML 2.5.1 specification, p 200): “Navigability means that instances participating in links at runtime (instances of an Association) can be accessed efficiently from instances at the other ends of the Association. The precise mechanism by which such efficient access is achieved is implementation specific. If an end is not navigable, access from the other ends may or may not be possible, and if it is, it might not be efficient.”

We don’t see any semantics in aggregations not already covered by common associations with appropriate directed names, but it could provide a way of easily visualizing whole-part relationships.

### Task

* We need to provide guidance on when to use Collections as oppose to simple aggregations. Collections are basically a specialized type of aggregation with additional semantics (order, structures). One case in which Collections come in handy is when we deal with order, either lists/arrays or more complex structures. Otherwise, for unordered Collections, we might just want to use aggregation.

### General Thoughts

Composition might make sense for a very few cases in which there is a strong lifecycle dependency, e.g. a cell in an array cannot exists without the array. However, it could provide a way of easily visualizing strong lifecycle dependency.

### Modeling Recommendation

* To use UML aggregation type and navigability (direction), but not UML ownership.
* To have mandatory multiplicities (cardinalities) on both ends of associations.

### Task

* We need a review of the existing aggregations/compositions, and any association with minimal cardinality of 1.

### General Thoughts

**UML note**: Association names must be unique within a package (UML uses package names as part of qualified association names). The current model has some name clashes in which two associations share the same name within a package.

### Modeling Recommendations

* Association names (the predicate) should be semantically meaningful, i.e. we should avoid generic names, e.g. has, is, contains, whenever possible.
* To have unnamed aggregations and associations for those generic cases where the association name is redundant, e.g. “has” and “contains” are implicit in the meaning of aggregations/associations. Transformations will add appropriate names when required, e.g. for the RDF representation all unnamed aggregations/compositions will be represented as “has”, which can be reused across packages.

### General Thoughts

**Note** on qualified names: Names in the UML model, both class and association ones, are fully qualified by their package name (could be understood as namespace). In other words,
QualifiedName = PackageName::LocalName, where the local name is the actual name of the class or association.

**UML note**: Qualified names of associations should be unique across the model. Local names for associations should be unique within their namespaces (packages) and can repeat across packages.

### Modeling Recommendations

* Associations with the same local name have the same semantics across the entire model.
* Local class names have to be unique across the model.
* Property names have to be unique across the model.

### General Thoughts

**Note**: UML association end names (roles) can be used to clarify the use of a class in different contexts. For instance, in a self-referent association, like isParentOf, roles can be used to identify the source and target classes as parent and child, respectively.

### Issues (post-prototype and prototype-review)

* Master issue [DMT-221](https://ddi-alliance.atlassian.net/browse/DMT-221) Associations
* [DMT-191](https://ddi-alliance.atlassian.net/browse/DMT-191) Associations, navigation, roles, aggregation
* [DMT-202](https://ddi-alliance.atlassian.net/browse/DMT-202) Association names in UML model - construction, uniqueness, name vs. role
* [DMT-214](https://ddi-alliance.atlassian.net/browse/DMT-214) HasMemberRelation should be an association, not a property

Documents with content of each issue (2018-11-28): <https://drive.google.com/open?id=1eWGxZBm_gTX_liv-Pn-2xbJ54LWCBoc3>

## Data Types

### Requirements

* Data types are important to assure consistency in the model.
* The definition and selection of primitive data types play a crucial role because more complex data types are based on these.
* Data types should be as consistent as possible across the model and the syntax representations. This minimizes mapping requirements and supports easy round tripping of metadata between different syntax representation instances.

### General Thoughts

**UML note**: Data types in UML are related to class definitions but are their own structural item. Data types can have properties but not relationships.

### Modeling Recommendation

* Classes that have no associations to other classes should be data types.

### Tasks

* The current classes in the structured data types package that have no associations to other classes should be defined as data types.
* Currently, there are classes in the structured data types package that have associations to other classes, they should be moved elsewhere.
* When a property is currently defined by a class rather than a data type, and the class has associations to other classes (hence it’s not a UML data type) we should either (i) make the property into an association or (ii) remodel the “data type” class into a real UML data type by removing associations, etc.

### General Thoughts

**UML note**: Data types include primitives, structured data types, and enumerations. A primitive has a definition which is outside of UML, i.e. cannot be defined further by UML. Structured data types have a structure which is defined by properties. The properties can be defined by other data types. Generalization of data types is allowed. Primitives include a set of five UML primitives and user-defined primitives.

### Modeling Recommendation

* Data types should be the same across different technologies, i.e. in the model and the syntax representations. If this is not possible, they should be easily mappable without the risk of information loss of the content or values.

### General Thoughts

**Background** from XML Schema Part 2: Datatypes Second Edition: “The framework has been influenced by the [ISO 11404] standard on language-independent datatypes as well as the datatypes for [SQL] and for programming languages such as Java.”

OMG provides a UML/XMI library for the XML Schema data types ([XSDLibrary.xmi](https://www.omg.org/spec/ODM/20131101/XSDLibrary.xmi)) in the context of ODM (Ontology Definition Metamodel). This library is large and complicated. It is not compliant to Canonical XMI. The only way to use it with DDI 4 would be to extract the XML Schema data type definitions and transform them into Canonical XMI.

### Modeling Recommendation

* To use UML primitive data types in the UML model and XML Schema data types in the syntax representations. Four out of five primitive data types, map directly to XML Schema data types at transformation time. The fifth one, the literal unlimited natural, needs to be handled separately to different syntax representations. More details on the latter are in the section on multiplicities below.

### General Thoughts

**Mapping of UML primitive data types to XML Schema data types:**

* Boolean – boolean
* Integer – integer
* Real – float
* String – string

### Proposal

* XML Schema data types are in the xsd namespace. We should maintain the namespace and use it in the UML to avoid name clashes. The primitive local names could have additionally the prefix “xsd:”. Colons seem to be allowed in data type names. Example: xsd:date.

### Task

* Create UML primitive data types for XML Schema data types which are used in DDI 4. These primitives should be in a separate package with the name XSD.

### Modeling Recommendation

* For XML Schema data types beyond the UML primitive data types we should define new UML primitives (approach followed by Eclipse developers). That way we can integrate XML Schema data types into the UML model without large effort. If we ever need data types that are not in XML Schema we propose to define new UML primitives by using the rules specified in ISO 11404. These additional primitive data types might need a special mapping to data types in the syntax representations which would require additional work. It is expected that this approach is used only in special cases if at all.

### General Thoughts

**Background** from Wikipedia: “ISO/IEC 11404, General Purpose Datatypes (GPD), are a collection of datatypes defined independently of any particular programming language or implementation.”

**Note**: The DDI 4 collection pattern could be applied for the data type aggregator from ISO 11404.

The overall understanding here is that the set of XML Schema data types and the ISO/IEC 11404 General Purpose Datatypes could be used in a complementary way.

Any other structured data types could be built on the basis of primitive data types.

This approach would support the use of data types in many syntax representations as the XML Schema data types are an acknowledged set across languages not just for XML. A mapping would not be required for these. They could be immediately used. It is expected that most requirements for data types in DDI 4 Core are covered by the XML Schema data types.

Structural data types which are built on the basis of primitive data types can be automatically generated for the syntax representations according to representation specific rules.

### Proposal

* Organization of all data types in a package structure like:
	+ DataTypes
		- Enumerations
		- Primitives
			* XSD
		- StructuredDataTypes

### Issues (post-prototype and prototype-review)

* Master issue [DMT-220](https://ddi-alliance.atlassian.net/browse/DMT-220) Data Types
* [DMT-200](https://ddi-alliance.atlassian.net/browse/DMT-200) Classes without relations should be defined as datatypes
* [DMT-205](https://ddi-alliance.atlassian.net/browse/DMT-205) Regular Expression rules for model
* [DMT-209](https://ddi-alliance.atlassian.net/browse/DMT-209) Datatypes - where are they appropriate
* [TC-62](https://ddi-alliance.atlassian.net/browse/TC-62) XML Schema Datatypes
* [XMI-1](https://ddi-alliance.atlassian.net/browse/XMI-1) How can XSD datatypes be included in the UML model?
* [TC-45](https://ddi-alliance.atlassian.net/browse/TC-45) Datatypes: clarification and improvement of organization

Documents with content of each issue (2018-11-28): <https://drive.google.com/open?id=1jbNrTmx0XLK2l3vRpCS7PzY85c-Hf9RW>

## Multiplicities

### General Thoughts

**Note**: To capture the unbound cardinality in associations and properties UML uses “\*”, which is represented as the UML data type LiteralUnlimitedNatural.

### Proposal

* In XML Schema, it should be represented as maxOccurs="unbounded".
* In OWL, it can be represented as maxCardinality defined as xsd:nonNegativeInteger, see [OWL reference](https://www.w3.org/TR/owl-ref/#maxCardinality-def). The value space of xsd:nonNegativeInteger is the infinite set {0,1,2,...}. The unbound cardinality could be only set by a very high number. There is no possibility to express infinity.
* In ShEx, it can be represented as MaxInclusive defined as xsd:integer. We need to check how that can be represented in RDF, e.g. ShEx. See [example](http://shex.io/shex-primer/#simple-expressions-example-1) in the ShEx Primer. The value space of integer is the infinite set {...,-2,-1,0,1,2,...}. The unbound cardinality could be only set by a very high number. There is no possibility to express infinity.

# Structural and Integrity Constrains

## Proposal

* We should use OCL to define constraints only when they are relevant and can be expressed in the different syntax representations. We should just document them in English otherwise.
* Formalizations of constraints are understood to be important. For the purposes of the core, however, they will be documented in English.

## Regular expressions for data type definitions

Regular expressions are constraints on string values. A property defined as string can have a related constraint in UML. Regular expressions don’t exist in OCL. The constraint language would be regular expression syntax.

### Modeling Recommendation

* Usage of the XML Schema regular expression syntax. This syntax is a common subset of many others used in Perl, Python, Java, etc. which is powerful enough for purposes in DDI 4. The usage of a common subset would enable immediate use of the regular expression in many syntax representations without adoption to local regular expression flavors which could be error prone.

**Example** constraint: regexpr: a+

**Background** from https://www.regular-expressions.info/xml.html: “Particularly noteworthy is the complete absence of anchors like the caret and dollar, word boundaries, and lookaround. XML schema always implicitly anchors the entire regular expression. The regex must match the whole element for the element to be considered valid.”

This is not limitation for regular expressions in the context of the DDI 4 UML model.

## Design Patterns

### Proposal

* The current realization of design patterns and their application use hidden multiple inheritance. We need to remove multiple inheritance as much as possible. One way of doing that, would be to make the Process Pattern extend the Collections Pattern.

### Issues (post-prototype and prototype-review)

* Master issue [DMT-222](https://ddi-alliance.atlassian.net/browse/DMT-222) Design Patterns
* [DMT-190](https://ddi-alliance.atlassian.net/browse/DMT-190) Issues to review regarding realizations of ProcessPattern
* [DMT-192](https://ddi-alliance.atlassian.net/browse/DMT-192) Design patterns in UML
* [DMT-201](https://ddi-alliance.atlassian.net/browse/DMT-201) Review the coverage, role, and value of patterns in the model
* [DMT-203](https://ddi-alliance.atlassian.net/browse/DMT-203) Review the role of Patterns and their implementation using realizes
* [DMT-208](https://ddi-alliance.atlassian.net/browse/DMT-208) Multiple realizations: Can a class realize multiple pattern classes
* [DMT-137](https://ddi-alliance.atlassian.net/browse/DMT-137) Signification pattern
* [DMT-213](https://ddi-alliance.atlassian.net/browse/DMT-213) Relation types belong to the whole relation, not to subsets of its members

Documents with content of each issue (2018-11-28): <https://drive.google.com/open?id=1SP0OmCl9cXMqt9nkUczJjZJWMZJMbsS1>