## UML Model

The UML model is core to DDI4. Work on the Canonical XMI (PIM), the OWL representation, and the DDI4R has raised a number of naming and organization issues. Each topic will have:

* Requirements or Statement of the Problem
* Pros and Cons (Questions and Limitations)
* Linked sources of discussion
* Terminology note (optional)

## 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

Associations and properties (defined by other classes or data types) have a similar meaning in UML.

Associations add in a UML rendering a line between two items (classes and/or data types) which helps understanding a relationship.

A class property is often defined by a data type. Then the property definition is appropriate.

An aggregation is a restriction of an association. An aggregating item can have a life of its own without the aggregation, the aggregated item.

### Question

* Does aggregation really add important information to the model and/or representation level? Some critique from the UML community says that aggregation is somewhat loosely defined and should be a decision on the implementation level and therefore not used on the model level.

A composition is an even tighter constraint than aggregation. A composing item cannot have a life on its own without the composition, the composed item. A composition is a strong statement.

### Questions

* Do exist items which don’t have a life of its own? This is often very dependent from the perspective. I.e. when modeling a car a wheel seems to be a composing item without own life. But from the perspective of a warehouse a wheel has its own life without the car.
* In a world of open standards very tight perspectives are not recommended. One never knows how an item might be used in another context or from another perspective.

Associations can have names. These names can add information - especially on diagrams.

### Question

* Should a general rule be established to use relationship names?

### Limitations

* Any name in an UML packages needs to be unique. Therefore relationship names in a package must be unique. When thinking on a flexible package management, i.e. a set of classes might be moved to another package, the relationship name should be unique in the model.

Associations can have role names on each side of the relationship.

### Question

* Do role names really add readability and semantics to the relationship?

### Post-prototype and Prototype-review issues related to Associations

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>

Cardinalities

* Consistent rules for assigned cardinality
* UML restrictions on cardinality for compositions

## 

## 

## [Class](https://www.uml-diagrams.org/class.html) Construction Rules

### Requirements

* A class supports a single usage/activity – it cannot be used in multiple ways (in terms of intended content and role)
* An activity/use should be represented by a single class
* Property names are associated with a single datatype [or realization sub-type] across the model
* Properties with same names but from different classes/datatypes should be defined and named in the same manner across the model library.

COMMENT: This approach may have led to too many classes because we’ve tied semantics into the naming of classes – this should be reviewed. Can classes be more generic and use the relationship name to provide the semantic?

Review what should be properties. In particular the set of Structured Data Types which contain relationships. Should these be targets of relationships of type Composition with identification needs handled in representation?

Review the idea of full identifiers in the UML model as identification is handled differently in various representations (i.e. RDF items all have identifiers, while in XML only classes that are targets of relationships have identifiers).

Review UML rules regarding uniqueness of names – in relation to the identifiers within the Canonical XMI.

### Post-prototype and Prototype-review issues related to Class Construction Rules

Master Issue: [DMT-224](https://ddi-alliance.atlassian.net/browse/DMT-224)

## Collections

### Statement of the problem

* UML tools do not recognize collections the way they are currently used. We could use the is ordered property of an association and then use an association to indicate membership in a collection.
* Class membership
* Collections might be applicable for additional purposes in the model

Pros and Cons

The MemberIndicator property was added to the Collections pattern as a way to indicate order in a simple sequential collection. We did not have a way to do this with the subset of UML features we use. The disadvantage of this approach is that membership in collections is not directly apparent in UML diagrams.

We could use associations in indicating memberrship if we allow the use of the is ordered porperty. Each binding would then need to implement ordering however it best can. The advantage would be that UML diagrams would directly indicate membership. The disadvantage would be that each binding would possibly use a different method of showing order.

### Post-prototype and Prototype-review issues related to Collections

Master issue

* [DMT-214](https://ddi-alliance.atlassian.net/browse/DMT-214) HasMemberRelation should be an association, not a property
* [DMT-213](https://ddi-alliance.atlassian.net/browse/DMT-213) Relation types belong to the whole relation, not to subsets of its members
* [DMT-196](https://ddi-alliance.atlassian.net/browse/DMT-196) UML provides Collection types for Multiplicity Elements. Could this be used for Collections?
* [DMT-206](https://ddi-alliance.atlassian.net/browse/DMT-206) Extending to an external class in a pattern; issue of UML semantics
* [DMT-212](https://ddi-alliance.atlassian.net/projects/DMT/issues/DMT-212?filter=allissues) Prototype Review: Generic Collection

Documents with content of each issue (2018-11-28)

<https://drive.google.com/open?id=1TMGoWM-WdKQo9IhpIIJUfP_D01As4aaa>

NADDI 2018 presentation “The DDI4 Collections Pattern” https://zenodo.org/record/1219278

## 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 representations. This minimizes mapping requirements and supports easy round tripping of metadata between different representation instances.

[Data types](https://www.uml-diagrams.org/data-type.html) in UML are related to class definitions but are their own structural item. Data types can have properties but not relationships.

Data types include primitives, structured data types, and enumerations.

Primitives include a set of [5 UML primitives](https://www.uml-diagrams.org/data-type.html#primitive-type) and user-defined primitives. 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 are defined by properties. The properties can be defined by other data types.

### Question

* How many levels of reuse of other data types is reasonable? Too many levels are not easily understandable and create a network of dependencies.

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

### Proposal

* Usage of [XML Schema data types](https://www.w3.org/TR/xmlschema-2/#built-in-datatypes) would meet the requirements mentioned above. XML Schema data types included a larger set of acknowledged data types. These data types are not XML-centric but are used in XML Schema/XML, in OWL/RDF and have direct counterparts in most of the programming languages.

From [XML Schema Part 2: Datatypes Second Edition](https://www.w3.org/TR/xmlschema-2/): “The framework has been influenced by the [[ISO 11404]](https://www.w3.org/TR/xmlschema-2/#ISO11404) standard on language-independent datatypes as well as the datatypes for [[SQL]](https://www.w3.org/TR/xmlschema-2/#SQL) and for programming languages such as Java.”

The XML Schema data types would be realized as UML primitive data types. Additional primitive data types can be added as needed which might be required for special cases. From [Wikipedia](https://en.wikipedia.org/wiki/ISO/IEC_11404): “ISO/IEC 11404, General Purpose Datatypes (GPD), are a collection of datatypes defined independently of any particular programming language or implementation.” This standard might be helpful for creating additional data types. But the definition might be outside of UML. UML primitive data types have a definition beyond UML, i.e. the definition could be just a reference to a section in a document.

The overall understanding here is that the set of XML Schema data types and the [ISO/IEC 11404 General Purpose Datatypes](http://standards.iso.org/ittf/PubliclyAvailableStandards/c039479_ISO_IEC_11404_2007(E).zip) 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 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.

Additional primitive data types might need a special mapping to data types in the representations which would require additional work.

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

### Questions

* How can the XML Schema data types be defined as a set of UML data types? There is not a standard way available.
* Which ways are common in the UML community to define/use XML Schema data types?
* If the XML Schema data types are used, another question needs to be decided: should the 5 UML primitives be used or only the XML Schema data types. UML UnlimitedNatural is not available in the XML Schema data types.

### Solution Options

* Definition of the XML Schema data types as a set of user-defined UML primitives in a specific package.
  + Pro: straight-forward solution, example exists already for two data types in the [Canonical XMI](https://bitbucket.org/wackerow/ddi4_xmi/raw/49d8d1afee2cca1c1bb9cfb44ca32974989f27d7/DDI4_PIM_canonical.xmi)
  + Con: work to be done, custom solution
* Usage of proposed solution by OMG, [ODM](https://www.omg.org/spec/ODM/1.1/) (Ontology Definition Metamodel) [XSDLibrary.xmi](https://www.omg.org/spec/ODM/20131101/XSDLibrary.xmi)
  + Pro: standards approach, existing work
  + Con: seems to very complicated, is it easily transformable to Canonical XMI? This would be a requirement. Is it really used in the UML community?
  + Possible solution: the XSD data types could be extracted of this library and put in the Canonical XMI form
* Reuse of existing approach like from Enterprise Architect
  + Pro: reuse of existing work
  + Con: dependency from a package created by a commercial vendor

### Regular expressions

Regular expressions are constraint on string values. A property defined as string can have a related constraint in UML. The constraint can be defined in a chosen language which is usually indicated by a prefix. Common languages are OCL and English. Regular expressions don’t seem to exist in OCL. Therefore a specific regular expression syntax can be chosen for this purpose.

### Proposal

* Usage of [XML Schema regular expression syntax](https://www.w3.org/TR/xmlschema-2/#regexs). This syntax is a common subset of many others used in Perl, Python, Java, etc. The usage of a common subset would enable immediate use of the regular expression in many representations without adoption to local regular expression flavors.   
  Example constraint - regexpr:a+

Citation 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 really a constraint for regular expressions in the context of the DDI 4 UML model.

### Post-prototype and Prototype-review issues related to Data Types

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>

## Design Patterns

Purpose of design patterns (from Wikipedia):

“In software engineering, a software design pattern is a general, reusable solution to a commonly occurring problem within a given context in software design.”

### Requirement

* Design patterns have the task to make the model more consistent in areas which are similar and a design pattern can be applied to.
* Classes realizing a pattern class must have all the properties and relationships of the realized class and may restrict cardinality, target class to sub-class [or specific realization], or a more constrained relationship type
* Relationship names related to patterns are consistent and are unique [noted used outside of pattern structure/realization]. *This is only required if the current mechanism of realizing design pattern classes are maintained.*

Design patterns help the modelers to build a consistent model.

From the perspective of a user of the model, the consistency is important but not the design patterns themselves. Therefore the design patterns themselves don’t need to be included in any model (PIM or PSM) which will be used for the representations or for any other purpose.

It is assumed that in the context of DDI 4 mostly creational design patterns are used. There is a set of acknowledged creational patterns, see: <https://en.wikipedia.org/wiki/Creational_pattern> and <https://en.wikipedia.org/wiki/Software_design_pattern#Creational_patterns> .

The [Abstract Factory](https://en.wikipedia.org/wiki/Abstract_factory_pattern) ([see also](https://www.uml-diagrams.org/design-pattern-abstract-factory-uml-class-diagram-example.html)) might address the requirement in the model. “Provide an interface for creating families of related or dependent objects without specifying their concrete classes.”

### Questions

* Do the current design patterns in the model have a relationship to the set of acknowledged creational patterns? If yes, this should be described. Reuse of existing approaches would make sense. If they are not used, what are the reasons for this?

The current approach of defining and using the design patterns is a custom solution. UML tools wouldn’t be aware of the intended inheritance from the design pattern class to the realized class. If the realized class inherits from another abstract class this could be understood as a combination between regular UML class inheritance and hidden inheritance by the “realizes” relationship between the design pattern class and the class to which the pattern is applied to.

### Solution options

* In general there is the question on how to define and use a design pattern in a UML model. There might be acknowledged ways of doing this. Further research seems to be necessary.
* One solution option would be to have a PIM for the creators of the model with allowed multiple inheritance. This assures that realized classes have the right set of properties and relationships of the design pattern class and they can inherit additionally from regular (possibly abstract) classes. Even inheritance from multiple design pattern classes would be possible. This PIM could be transformed to another PIM without the design pattern classes and with realized “flattened” classes regarding the design pattern inheritance. This PIM would be for the users of the model.  
  This approach would also decrease the number of classes and relationships in the model.

### Post-prototype and Prototype-review issues related to Design Patterns

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>

## Library and Views

### Current situation

# Views are invented to make the use of DDI simpler and to represent a sub-set of the full library. Views should provide a specific perspective on DDI from the requirements of a specific user group and/or specific use cases.

# This goal is under heavy risk with the current realization of the XSD and OWL representations.Currently DocumentInformation from is injected for all Functional Views.

### Requirements

* Review views from a user (business) perspective, but in a way that avoids too much overlap between them when it comes to classes.
* Clearly document what each view can (and cannot) realise. Add a description of relationships between views.
* Define rules for the OWL (and XSD) representations

### Questions

* The question is where to draw the line in terms of pulling in related super classes (for OWL) and related classes to a view? How are the rules?
* How can the rules be formalized in a way that they are in sync for the different bindings (if possible), can be used by the transforming programs, and/or they can be documented in a formal way?
* Decide if DocumentInformation as specified in DMT-193 should be in the Functional Views.

### Pragmatic questions

* How can the gap be communicated between the claim that views make DDI usage simpler and the huge amount of classes per view?
* Can things be made simpler?
* Which rules should be applied for the OWL?

Options and solutions

The Functional View is intended to present a clear sub-set of classes needed for a common use case or set of use cases. In the model this ends up being a simple listing of classes with documentation on the boundaries of set (listed as restrictions in the availability of target classes for any relationship from an included class). The content of Functional Views may be stored in a class content library that provides the content for a generated Functional View in a specified representation. As such there needs to be a clear means of identifying all the classes that go into an instance of a Functional View, either by identifying one or more “top” classes which result in a hierarchy or graph of included content, or by defining some selection filter. The primary questions for review are whether the selection of classes for any given Functional View achieve the goal of that Functional View, then whether the model of the Functional View is appropriately expressed in the various representations.

● If Functional Views are seen as profiles of a subset of classes with discussion of where the boundaries are located the UML expression seems reasonable. Further investigation should be made of UML features that may express these more accurately or consistently.

o For example using UML means of presenting sub areas of the model through a “view”

Solution options

* Currently, views are defined as UML packages in the Canonical XMI. The packages import the relevant classes by reference from other packages. This seems to be a straightforward way. But multiple UML tools don’t deal with this part of UML (elementImport).
* All relevant parts for a view could be extracted from the library as copy and put into a separate PIM per view. This approach seems to be more promising because they would be recognized as model in the common UML tools. This approach needs more exploration and testing.

● Current Functional Views were created by hand, identifying relationships and following them through to additional classes, identifying restrictions on relationships, verifying that no orphan (unconnected) classes were included, and identification of potential entry points (top of hierarchy).

o In creating Functional Views we found many difficulties in determining end points (defining boundaries). We could not anticipate user needs even for the examples and had to add classes throughout the review process.

o It isn’t clear within an instance where to “enter” the content. Document Information was an attempt to clarify information about the instance itself and to clarify entry points. However, not all instances are “documents” so this does not seem to be the ideal solution.

● In looking at whether the representations in the various representations provide similar support we find that in RDF Functional Views were handled by putting them into an inheritance tree which allowed you to limit the view but had no effect on validation.

● The use of XML schema to express Functional Views created a number of problems:

o Created “hard” boundaries where relationships were blocked

▪ External references were allowed but in effect the schema would not be able to process the content of the reference

▪ Creates difficulty in the ability to share across a contributed environment. Not all uses of a class support all relationships raising the question of how this additional material is handled. For example if an Instance Variable created for a Descriptive Codebook instance were used by a Data Management Process instance the Data Management Process could not contain all of the content of the Instance Variable resulting in essentially 2 versions of the Instance Variable (one with all content and one with limited content).

▪ This situation also affects the ability to “round-trip” between representations and/or repositories.

▪ RECOMMENDATION: TC has found a number of issues with the XML rendering of Functional Views and reviews should provide recommendations for how Functional Views should be surfaced by representations. (see TC-75 for a fuller description)

### Post-prototype and Prototype-review issues related to Library and Views

Master issue [DMT-225](https://ddi-alliance.atlassian.net/browse/DMT-225) for Library and Views

* [DMT-193](https://ddi-alliance.atlassian.net/browse/DMT-193) Review use of DocumentInformation relationship classes in all Functional Views
* [TC-75](https://ddi-alliance.atlassian.net/browse/TC-75) too many classes in the chain / not clear where to stop the chain

Documents with content of each issue (2018-11-28)

<https://drive.google.com/open?id=1l_kMhjGUihvIg3qA1zhDSo1JenCvAny5>

## Model Goals

### Requirement

* Balancing conflicting goals of the model

There are different perspectives on the model:

* Creating the model, consistency of the model, persistence of the model
* UML conformance / usage of UML
* Using the model
  + With UML tools, subsetting the model, connecting to other models
  + Transformations to representations
* Supporting the main representations in a manner that the transformations to the representations are straightforward

One solution option could be to have two different PIMs, one for the creator’s perspective and one for the user’s perspective. The latter might be a simplification of the first and it might contain more support for the representations. Creator’s PIM -> User’s PIM -> PSMs -> Representations

Application examples:

* Design pattern classes could be only in the creator’s PIM. The transformation would remove them for the user’s PIM and enforce that all classes where the design patterns are applied to have the pattern set of properties and relationships
* Data types: the creator’s PIM could have primitive definitions according to ISO/IEC 11404. The transformation to the user’s PIM would map data types to XML Schema data types if possible

### Post-prototype and Prototype-review issues related to Model Goals

Master issue [DMT-226](https://ddi-alliance.atlassian.net/browse/DMT-226) related to Model Goals

* [DMT-198](https://ddi-alliance.atlassian.net/browse/DMT-198) Model: Balance between different goals? / Usage of UML?
* [DMT-207](https://ddi-alliance.atlassian.net/browse/DMT-207) Review parameters of our use of UML modeling
* [DMT-216](https://ddi-alliance.atlassian.net/browse/DMT-216) [flavio will file one regarding binding rules, what they cover, how close binding should follow UML model

Documents with content of each issue (2018-11-28)

<https://drive.google.com/open?id=1zybGU4uS1uPi-Sygz9bwsn5yxb-WMgLO>

## Name length conventions

Specification of the problem

Names of packages, classes, properties and associations

* + Name length restriction of 32 characters
  + Recommendation for names: 3-4 words
  + Consider a generic prefix for all names in the DDI namespace like “Ddi”,(solve model to model linkage) or introduce prefixes through the transformations/bindings (can be used/not used depending on the representation)

### Post-prototype and Prototype-review issues related to Name Length Conventions

* [DMT-218](https://ddi-alliance.atlassian.net/browse/DMT-218) Name length restrictions
* [DMT-219](https://ddi-alliance.atlassian.net/browse/DMT-219)Consider a generic prefix for all names in the DDI namespace like “Ddi”

## Package Structure

### Requirement

* Packages should express regions of the interrelated content
* Packages and classes should be named and organized in a way that they can be easily move to another location of the model

### Current situation

* The stated purpose of packages was to support internal management and provide some groupings of closely related classes for users. Currently they are closely related to development work groups.
* Large set of classes are “shared” across the model either as structural pieces or patterns
* Due to limitations of Lion, packages were created to development areas and pieces added for the use of a particular Functional View
* There are no rules to constitute what goes into a package which can result in subsets of unrelated classes within a package or confusion when classes relate across package lines
* The model is a complex network and interrelations are not well presented
* Patterns are clearly separated and all pattern classes are within the pattern package, even those that would be considered Structured Data Types

COMMENT: Need to revisit the Package structure in conjunction with the review of the UML modeling use. Clarify the role and use of Packages in the Class Library for modelers and for end users.

### Post Prototype and Prototype Review Issues related to Package Structure

Master issue [DMT-223](https://ddi-alliance.atlassian.net/browse/DMT-223) related to Package

* [DMT-195](https://ddi-alliance.atlassian.net/browse/DMT-195) Review the role of Packages in the COGS environment and rationalize rules for package content and naming
* [XMI-5](https://ddi-alliance.atlassian.net/browse/XMI-5) How to reference classes in other packages from the view package

Documents with content of each issue (2018-11-28)

<https://drive.google.com/open?id=1O6wIeQ2FR3nt690hannltHvHMT-BrGww>

## UML Features

UML has been identified as the modeling environment for DDI and the source document for the production process that generates multiple representations of its content. As such a clear documentation of decisions concerning the use of UML and the construction rules are important both to those modeling and to the end users. Decisions regarding specific aspects of usage in the areas of Data Types, Associations, Functional Views, and Patterns need to be reviewed for compatibility and the ability to present a coherent whole. In addition, specific issues of usage outside of these major areas need to be clarified to ensure a comprehensive usage document.

The goal should be one document with clear rules on which features of UML can be used and how they are used.

### Post-prototype and Prototype-review issues related to UML features

Master issue [DMT-226](https://ddi-alliance.atlassian.net/browse/DMT-226) related to UML features

* [DMT-197](https://ddi-alliance.atlassian.net/browse/DMT-197) UML Property modifier "id" should be used for DDI URN parts
* [DMT-199](https://ddi-alliance.atlassian.net/browse/DMT-199) PostPrototype: UML modeling
* [DMT-204](https://ddi-alliance.atlassian.net/browse/DMT-204) Review of use UML for appropriateness of application
* [DMT-211](https://ddi-alliance.atlassian.net/browse/DMT-211) Prototype Review: evaluate use of UML ordered property
* [DMT-196](https://ddi-alliance.atlassian.net/browse/DMT-196) UML provides Collection types for Multiplicity Elements. Could this be used for Collections?
* [DMT-206](https://ddi-alliance.atlassian.net/browse/DMT-206) Extending to an external class in a pattern; issue of UML semantics

Documents with content of each issue (2018-11-28)

<https://drive.google.com/open?id=1bCvkEBXW3IfRYN9_LZv0sUdriGRAmSIM>

## Versioning of the Model

Statement of the problem

How do we keep track of changes of classes and views?

Changes of classes and view will happen also after the prototype release. How should be kept track of these changes?

Following items could have versions: classes, packages, views, releases.

In general there is a distinction between a technical version and a business version. The technical version number changes each time the versioned object changes. The business version is assigned to objects which should be identifiable by this version. It is assumed that the focus is here on the business level.

There are at least two perspectives regarding versioning. The creators of the model would like to keep track of version changes. The users of the model and/or representations, especially software developers would like to know if a class definition has changed.

Thinking on UML, the version could be a property of each class. This way the version would be recognizable in UML tools. This property needs not to be implemented in a software.

There are two older proposals available for versioning, [one from Achim](https://ddi-alliance.atlassian.net/browse/RDFOWL-18) and [one from Flavio](https://ddi-alliance.atlassian.net/browse/DMT-10). This could be a basis to the resolution of this issue.

### Post-prototype and Prototype-review issues related to UML Versioning of the Model

* [DMT-194](https://ddi-alliance.atlassian.net/browse/DMT-194) How do we keep track of changes of classes and views?

Documents with content of each issue (2018-11-28)

<https://drive.google.com/open?id=1n-K7RIdxtvxEDEGE-Lg5Cxx0gCXOnPy_>

## 

## *To be reviewed and assigned to the correct area.*

The following is a summary of the issues relating to the UML model that have been classed as post-prototype.

Questions raised concerning the use of UML:

● What should the UML be used to express?

● Where are the lines between a conceptual model, and information model, and the implementation model (representations) and where does the DDI UML apply?

● Is there a better way to express patterns in UML?

● How can/should regular expressions be captured in the UML?

● Name uniqueness - within model, within package

● Are there UML conventions that would simplify our work (collection types for multiplicity elements, property modifier "id", UML profiles, UML isOrdered property)

● What is the role of Packages as used in Lion - internal only, organization for end user, mixed?

● What is the role of Functional Views within the UML model and how does this affect how they are expressed in representations?

Expression of different relationships from the UML to representations

● Relationship targets should not always require identifiers (example the target of a composition)

● Clarify the roles of property (target of a primitive or a set of primitives?)

● Which Structured Data Types should be treated as the target of a composition relationship?

● Are we capturing the needed information on relationships (type, source cardinality, target cardinality, directionality)

Lion/representation constraints that have creeped into modeling rules

● A class can realize only one class type

● Currently done by pattern classes extending from another pattern class

● Lion rules of "overwriting" relationships with the same name means only a single class can be realized

● Overwriting inherited content with another relationship/property of the same name