Creating and Using a Pattern

2016-08-15 – W. Thomas

Role of a Pattern

A Pattern in DDI represents a common, reusable structure to provide consistency in the relationships between classes which, as a group, relay common activities or structures such as, a collection of classes, a production process, defining methodology or approach, etc.

The intent of the Pattern is to provide implementers with a consistent representation of classes representing similar things in a variety of applications. The use of a Pattern lets the modeler apply a known Pattern and customize it to a particular use by constraining relationships or adding applicable properties or relationships.

Design Guidelines for Pattern Content

- The package containing the Pattern "Xxxx" should be named XxxxPattern
- The package should contain ONLY classes within the pattern (classes may relate to classes outside the pattern which can be used directly (without creating a new class)
- All classes in the Pattern must be abstract
- Classes in the Pattern may inherit from other classes via use of extension base
- Classes in the Pattern should not "realize" another class
- Class content should be as limited as possible
    - Use Identifiable rather than AnnotatedIdentifiable
    - Limit use of common descriptive properties unless required by the role of the class (name, usage, etc.)
    - Provide usage specific names for other common property types like Date or DateRange in order to limit possibility of name clashes in realizations (e.g. xxxxDate)

Implementing a Pattern

A Pattern is "realized" by creating a new class containing ALL of the properties and relationships of the Pattern class PLUS the following relationship:

Name	Target Object	Description	Source cardinality	Target cardinality	Relationship type
realizes	[Pattern class name]	Uses pattern for a [Pattern class name]	0..n	1..1	Neither

Use the following guidelines and refer to “Xxxx” document for detailed information on the use of Patterns.

- A class may realize more than one class (generally from multiple patterns)
    - Example: a single class could have the role of an Design in a Methodology Pattern and a Member in a Collection Pattern
    - In replicating content ensure that any common properties have the same data type, description and role (conflicts should be brought to the attention of the Modeling Team)
    - Clarify the nature of the multiple roles for the class in the class level documentation
- Carefully replicate all properties making sure to include all inherited properties
    - Note that some inherited properties are overridden by constraints. In this case used the constrained version of the property
    - Example: A StrictOrderRelation replicates the following properties and restricts them

Inherited from BinaryRelation	DataType	Cardinality	Restriction
totality	TotalityType	1..1	[no restriction]
reflexivity	ReflexifityType	1..1	Fixed to Anti_Reflexive
symmetry	SymmetryType	1..1	Fixed to Anti_Symmetric [1]
transitivity	TransitivityType	1..1	Fixed to Transitive

[1] Restricted in to Asymmetric in AsymmetricBinaryRelation then further restricted

- Carefully replicate Relationships including all inherited relationships
    - Retain the use of the relationship name if possible
        - If changed for clarification, note the name of the relationship it represents from the Pattern class
- Restrict the target object as needed
    - Note that if the Pattern Class requires a specific type (Member, Node, etc.) the selected Target Class must be of that type either through realization or inheritance
- You may constrain but not relax the source and target cardinalities
- Add the relationship name “realizes” and the appropriate pattern class
- Change extension base from Identifiable to AnnotatedIdentifiable if needed
    - If the extension base is other than Identifiable, simply add the property of has Annotation (0..1) Datatype=”Annotation” with the description “Provides annotation information on the object to support citation and crediting of the creator(s) of the object.”
- Add any additional properties or relationships needed for the specific use of this class.