Designations and Code Item in DDI

*By Flavio Rizzolo - Draft 0.3*

What follows is based on my basic knowledge of semiotics and my personal understanding of the definitions in the paper “Identifiers, Labels, Names, and Designations”, by Frank France and Dan Gillman.

First things first, we need a model for Sign.



A Sign links a Signified with a Signifier that denotes it. A Signifier is a concept whose extension consists of tokens (perceivable objects). This is where things get muddier because many people call these tokens “values”, when in fact they are Signifiers of Values (via a Datum). See diagrams below.

Codes enter into the picture as Designations. A Designation is simply a Sign where the Signified is a Concept.



A Code then is a type of Designation that has Non-Linguistic Signifiers and where the Signified is a Category.

We don’t need the notion of Datum to model Code Item, but I will include it here for completeness and also to make sure that everything is consistent with other parts of the model.



A Datum is a Designation of a Value. A Value is a Concept with an equality operation defined. Values are sometimes confused with the tokens which denote them. For that reason I’m hesitant to use the word “value” in our model.

Modelling designations

So far the theory. The challenge is to come up with a simplified model that works well in practice and aligns with the major concepts outlined above. One of the issues is that association classes are not supported in our current modelling platform. Another is that the notion of a Signifier is too abstract for most modelers and users. One possible solution that takes care of both issues is to make the Signifier into a property of the Sign, which I call *representation* and most people would call “value”. The Data Type for *representation* is still to be determined.

The next figure depicts the proposed solution. Sign and Signified will become the two top level classes, related by the *denotes* association. As before, a Designation is a type of Sign and Concept a Signified. In addition, Code is a type of Designation and Category a Concept. We could use the *denotes* association inherited from Sign, or we could create specializations for each level (I prefer the latter because we can restrict the target to the appropriate class). Designation and Code are going to inherit the *representation* property from Sign.

A Node takes meaning from a single Category and has an optional set of Designations. In the case of the Code Item subtype, at least 1 Designation is required, a Code more precisely. It’s important to note that a Code Item can have only 1 Category, thus if there are multiple Codes associated with a Code Item, all of them will have to denote the same Category. In other words, Code Item must contain one or more Designations, at least one of them must be a Code, and all its Designations must be synonyms, i.e., be associated with the same Concept.



This constraint is currently missing in the DDI and GSIM models, and it’s also missing from the diagram above because we have cardinality constraints only on individual associations in our current modelling platform. However, it can be document in the Code Item description.

Alternatively, we could have the Signifier as a class in the model, in which case the *representation* property in Sign would become the *is represented by* association to Signifier. That association in Code would be specialized to Non-Linguistic Alphanumeric Signifier.



The advantage of this approach is that it’s easier to perform lexicographical analysis (e.g. homograph identification, stop words removal, stemming, term categorization) since Signifiers are first class object and can be more easily manipulated. On the other hand, it complicates the model and makes it more difficult to use for the average modeler/user. One solution could be to have the Signifier as a Data Type in the conceptual/logical model and clarify in the documentation that people have the option of making Signifier into a class in their physical implementation