**Mapping Language Specifications**

Mapping cultural-historical data to semantic networks is relatively simple since i) specialist/primary information databases frequently employ a flat schema, reducing complex relationships into simple fields ii) cardinality constraints need not be enforced and iii) specialized source fields frequently map to composite paths under the CRM (or any other target schema), making semantics explicit using a small set of primitives more easy to learn. Another positive effect of mapping to composite paths is the use of intermediate nodes that frequently offer themselves as “hooks” for integration with other complementary sources, such as a production event between object and technique.

Mapping consists of three steps:

1. Schema matching: declarations of equivalence of source schema constructs with target schema constructs.
2. Instance generation policy: declarations how identifiers of nodes and numerical data types of data sets transformed into the target schema have to be generated from information elements in the source data sets. Step 1) and 2) form the mapping definition.
3. Transformation: Executing instructions of the mapping definition in order to transform a set of source data sets automatically into target data sets.

Domain experts that are aware of the meaning of the target schema can learn with reasonable effort and without IT skills how to perform schema matching, since they are aware of the meaning of the source schema. IT experts may not understand the meaning of either schema or underestimate it leading to errors and labor-intensive, time-consuming correction processes.

To assist domain experts on performing the mapping activity and the IT experts on performing the data transformation process, a Mapping Definition Language and a set of compatible tools are required.

The basic principles that the language and the tools should comply with are:

* The transformation should be possible by executing specifications given in the Mapping Definition Language by an ***automatic interpreter*** without human intervention. The schema matching should be expressed in a ***declarative way***, in order to be readable by both domain experts and machines.
* The language should be ***symmetric*** with respect to the way equivalent source and target schema paths are declared, and moreover ***potentially invertible*** allowing bidirectional interaction between providers and aggregator and thus supporting not only a rich aggregators’ repository but also corrections and improvements in the providers’ databases.
* Schema mappings should be defined in such a way that they can be ***collaboratively created*** and discussed by experts. Emphasis should be given on establishing a ***standardized mapping description*** which lends itself to collaboration and a sufficient specification for the transformation of each instance of a source schema into an instance of a target schema while preserving as much as possible its initial ‘meaning’.
* The Schema Matching and the Instance Generation policies should comprise different distinct steps in the data provision workflow. Instance Generation is more technical and does not require deep understanding of domain knowledge. Therefore it is more likely that is better understood by an IT expert than by a domain expert and the language should ***decouple the Instance Generation*** from the schema matching and to completely separate the definition of the schema matching from the actual execution.
* The Schema Matching declarations should allow for declaring the connectivity of the target graph in **a symbolic way** comprehensible to the domain expert, i.e., which entities reoccurring in the declarations will be transformed into the same identical per source data unit (record, parent tag etc.). Connectivity of the target graph should not be achieved by “smart” instance generation policies.
* ***Domain experts*** should be capable of testing the semantics, reading and validating the schema matching with adequate tools. Therefore there should be a distinction between mapping information from the domain experts who know and provide the data and information created by the IT technicians who actually implement data translation and integration solutions, and serves as an interface between both.
* There should be the capability to keep the ***schema mappings*** between different systems ***harmonized*** by semiautomatic comparisons of schema matching instructions.

Specifically, regarding **CIDOC-CRM** as target schema, the language should support

* interpretation of source schema as semantic model (nodes and links)
* mapping each element of that to an equivalent target schema path, such that each instance of an element of the source semantic model can be converted into a valid construct of the target schema with the same meaning.