A Reference Model for Data Mapping tools

Martin Doerr, Achille Felicetti

[*martin@ics.forth.gr*](mailto:martin@ics.forth.gr)*, achille.felicetti@pin.unifi.it*

November 2012

# Introduction

This is a first draft of a reference model for mapping tools. Intension is to address a comprehensive and sustainable functionality in a scenario of information providers and aggregators, including long-term maintenance of resources. We assume a distribution of responsibilities, in which the information provider curates his resources and provides in regular intervals updates, whereas the aggregator is responsible for the homogeneous access and co-reference resolution. In the course of the transformation of resources to the target system, kinds of quality control can be done which the provider has no means to do (see also services provided by OCLC). Therefore the information provider receives and benefits from data cleaning information from the aggregator. The challenge is to define a modular architecture of as many components as possible that can independently be developed and optimized.

# Process Model

We assume the following sequences of processes for the information submission:

1. Provider experts and a target schema expert (e.g., CIDOC CRM) define a schema matching, which is documented in a schema matching definition file.
2. In order to do so, all source schema elements must be well understood and mapped to target schema paths. Therefore also the target schema must be well understood. Both tasks need two independent tools to visualize source and target schema and source content. Adequate navigation and viewing techniques must allow for overviews and understanding of details.
3. The matching process must lead the user through all source elements to make a mapping decision. This may be supported by tools suggesting mappings (automated mapping). The automated mapping tools should recalculate proposals with each new mapping decision by the user. They should make use of “mapping memories” of analogous cases. They should take into account incremental mappings after source or target schema updates.
4. The matching may need to interpret source or target terminologies in order to resolve data dependent mappings.
5. Some data may not allow for matching decisions and need improvement by the provider. It must be able to define filters for these data and feed them back to the provider.
6. After the matching process, the URI generation policies for each target class instance must be defined. This is typically a task the provider has not interest in or knowledge for. It depends on the aggregators policies for co-reference resolution. The URI generation policies can be introduced in an abstract form in the schema matching file. Some URI generation policies may include look-up of online resources. Changes of URI policies of the provider may result in changing the definitions without affecting the schema matching.
7. URI definition may reveal “dirty data” of the provider and be reported back. “Dirty data filters” must be introduced in this step. The source must again be analyzed for that.
8. The data transformation is executed. “Dirty data” are reported back.
9. Data are integrated at the aggregator. Co-reference resolution and other logical consistency diagnostics at the aggregator side will provide more information to the provider for data cleaning and improvement.

# Components

1. Source analyzer (**XML**, E-R, RDFS):
   * Schema documentation (scope notes/definitions)
   * Visual, hierarchical schema/ used schema/instance presentation
   * hide/expand subtrees
   * use statistics for each field
   * value lists for each field
   * random value samples for each field
   * user calls source cleaner
2. Target schema analyzer (XML, **RDFS**)
   * Schema documentation (scope notes/definitions)
   * Visual, hierarchical schema presentation, inherited properties!
   * Better: 3D hierarchical schema presentation
   * view generation by browsing: open/hide
   * hide/expand IsA subtrees,
   * hide/expand path graphs,
3. Schema Mapper
   * loop through (used) source schema, breadthfirst
   * “mapping memories (see below)” propose mappings.
   * user calls source analyzer & source cleaner.
   * user defines domain-link-range map, allow for conditions on **source or target terms**
   * user explains rationale, give feedback to mapping memory (negative and positive).
   * user defines “dirty source” conditions
   * exhaustive mapping loop: each source field is either “mapped”, “wrapper” or “ignored”.
   * visualize mapping: by parts of source schema/ by target concepts
   * create human readable schema matching file, same syntax for any kind of source.
4. Incremental Schema Mapper **(schema update)**
   * imports mapping file
   * imports old/new source/target
   * loops through nodes possibly affected by schema change
   * etc, as above, in increments.
5. URI Mapper
   * loop through schema map by source instance tree (visual)
   * define co-reference rules within records, subtrees of records (e.g., same name of persons = same person, shared intermediate nodes/events)
   * user calls source analyzer & source cleaner.
   * define URI rules for all independent nodes
   * allow for source conditions
   * user defines “dirty source” conditions
   * define use of terminology maps for term fields.
   * insert co-reference variables in mapping file, insert URI rules
   * **incremental mode**: imports new URI definion rules, loops through affected maps
6. Transformer
   * preprocess terminology: map source terms to target terms for target term conditions by a terminology map.
   * runs through source,
   * resolve conditions against source/target terms,
   * generates target instances
   * throws out “dirty sources” for reprocessing (special mapping, manual correction, etc)
   * post-process URIs: duplicate detection against authorities, databases
   * post-process terminology: map source terms to target terms by a terminology map.
7. Source cleaner (black magic, heuristics, machine learning, programming):
   * user defines rules for cleaning values, e.g. punctuation, date formats etc
   * verify against authorized terminology lists.
   * run rules for test on value lists
   * clean source
   * report changes
8. Mapping Memory tool
   * compares source patterns to schema matching definitions
   * (possibly also URI generation memory)
   * suggest to user mapping, displays rationale
   * learns from use
   * learns from **source/target schema updates**
9. Terminology maps
   * source/target terminologies mapped by SKOS format
   * coverage: each source term must be covered by a target term (combination), or source term enters target…
10. Mapping Memory Viewer
    * display mapping memories
    * user queries and browses
    * user feed-back

Typical mapping ambiguities:

Art Object: Info Object or Material Thing? What is the intention of the source?

Production: Conception, Expression Creation or Carrier Production? What is the intention of the source? In ceramics, painting it may be identical. But Renaissance workshops may aready make a difference. Then: Books, metal casts, etchings.

Nationality: Birth, Citizenship, Style

Lifedates: How to present living people?

Nation: Group, or Period? “Nation→has realm→Period”?

Settlement: material features, space-time spread, people see: City of Jerusalem in 12/13th century, Niniveh, Orvieto:population transfer, Heraklion-Roka.

Name, Title versus Type

Accept multiple mappings!

materialize shortcuts or not? Should we distinguish primary from secondary knowledge?

Then: do not create shortcuts, if full path exists…

3 levels: a) intellectual schema equivalence

b) identifier policies

c) target system reasoning policy/ supported queries/discourse