5
StdTrip Tool

5.1 StdTrip Architecture

Figure 14 shows the architecture of the StdTrip. It consist of six components, *Conversion* to extract the metadata and convert to an ontology model, *Alignment* to execute match operations, *Selection* to manipulate the previous match result, in order to display to user, *Inclusion* to execute keyword search on the Web, *Completion* to suggest recommendations, and *Output* to generate outputs. The *Selection* and *Completion* modules require of the user interaction.

1. **Conversion:** This component manage the extraction of metadata and transformation from the relational database to the OWL ontology. These operations are broken in three specialized modules namely *Metadata*, *Relational-to-ER* and *ER-to-OWL*, detailed below.

   – **Metadata:** In this module, the metadata is extracted from the relational database, such as table names, column names, column datatypes, primary key columns, foreign key columns and column descriptions. This set of information is also called RDB schema, which is essential to initiate the tool execution. Currently the metadata extraction is supported for the MySQL Relational Database.

   – **Relational-to-ER:** In this module, as the name can suggests, the relational database model (logical model) is converted to the Entity Relationship model (conceptual model).

   – **ER-to-OWL:** This module is responsible for convert the ER, obtained in the previous module, to an OWL representation called *Database Ontology*. The RAP framework [Oldakowski et al. 2004] is used for the OWL serialization.

2. **Alignment:** This component address the iterative alignment between the Database Ontology and others standard ontologies. The K-Match tool that implements this process is detailed in section 5.2.

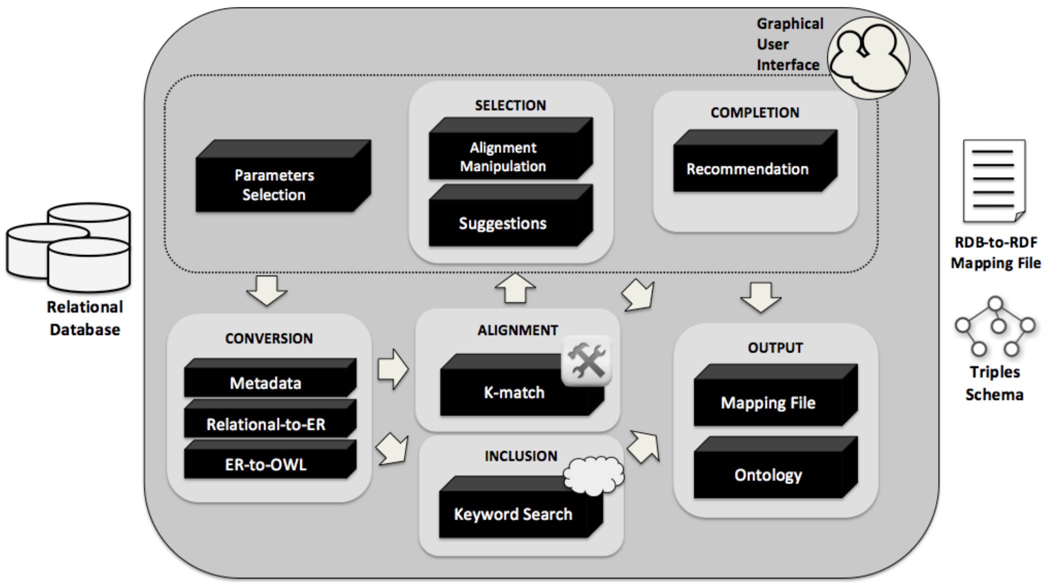

Figure 14: StdTrip Architecture

3. **Selection:** In this component, the match candidates, obtained as the result of the Alignment component, are manipulated to elaborate the suggestions. These suggestions are display to user through GUI, where a user may selects the term that he or she considers the best match for each of the database ontology terms. The manipulation of the alignment results and the elaboration of the suggestions are implemented in the following modules.

   – **Alignment Manipulation:** In this module, the alignment, obtained for each iteration of the alignment module, is parsed and saved for each ontology term. After the execution of this module, each one of the database ontology terms has a vector of possible matching terms.
   – **Suggestions:** In this module, each array of terms, obtained in the former module, is first arranged in decreasing order of similarity value and then displayed in a friendly interface for the user selection.

4. **Inclusion:** In this component, We applied a keyword-based search for the ontology terms not considered for the previous modules. This operation is done with the aid of the Watson [Sabou et al. 2007] API.

5. **Completion:** In the *Completion* component, a set of recommendations and best practices, on how vocabulary and new terms should be produced, are provided and strongly recommended to the user. The new

ontology terms generated in this process are analyzed for minor corrections and then send to the *Output* module.

6. **Output:** This component is responsible for the generation of the output, which is composed by two modules. In the Ontology module, the original data schema labels are substituted by ontolgy terms selected in the *Selected* module or produced by the *Completion* module. Code 1, as follows, illustrates the Triples Schema fragment for the Author-Publication running example. For the complete Triples Schema file, see the Appendix B

Code 1 "Triples Schema" for the Author-Publication example.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:owl="http://www.w3.org/2002/07/owl#"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
         xmlns:bibtex="http://purl.org/net/nknouf/ns/bibtex#"
         xmlns:foaf="http://xmlns.com/foaf/0.1/"
...
<rdfs:Class rdf:about="http://purl.org/net/nknouf/ns/bibtex#Article">
   <rdfs:label xml:lang="en">Article</rdfs:label>
</rdfs:Class>
...
<rdf:Description rdf:about="http://purl.org/net/nknouf/ns/bibtex#Article">
   <rdfs:subClassOf rdf:resource="http://purl.org/net/nknouf/ns/bibtex#Entry"/>
</rdf:Description>
...
<owl:DatatypeProperty rdf:about="http://purl.org/net/nknouf/ns/bibtex#hasJournal">
   <rdfs:domain rdf:resource="http://purl.org/net/nknouf/ns/bibtex#Article"/>
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
   <rdfs:label xml:lang="en">hasJournal</rdfs:label>
</owl:DatatypeProperty>
...
<owl:ObjectProperty rdf:about="http://purl.org/example#worksFor">
   <rdfs:domain rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
   <rdfs:range rdf:resource="http://dbpedia.org/ontology/EducationalInstitution"/>
   <rdfs:label xml:lang="en">worksFor</rdfs:label>
</owl:ObjectProperty>
...
```

In the Mapping File module, we use the ontology terms selected or produced in previous stages to generate the mapping specification file, which is used by a RDB-to-RDF engine to produces the triples set from the original database. The *Mapping File* module currently produce the mapping specification file for the Triplify Tool [Auer et al. 2009]. It is important to note that this module is easily customized for other RDB-to-RDF mapping files. For instance, Code 2 fragment below shows the Triplify mapping specification file for the Author-Publication example. For the complete mapping file, see Appendix A.

Code 2 Fragment of the Triplify mapping file for the Author-Publication example.

```
$triplify['queries']=array(
'article'=> "SELECT
          publication_id as 'id'
        , journal as 'bibtex:hasJournal'
     FROM article",
 'author'=> "SELECT
       author_id as 'id'
        , institution_id as 'ex:worksFor'
        , first_name as 'foaf:firstName'
        , last_name as 'foaf:familyName'
        , address as 'dbpedia:address'
        , email as 'foaf:mbox'
     FROM author",
  ...);

 $triplify['classMap']=array(
       "article" => "bibtex:Article",
        "author" => "foaf:Person",
 ...);

 $triplify['objectProperties']=array(
        "ex:worksFor" => "institution",
        "bibtex:hasAuthor" => "author");
 ...
```

5.2 K-Match Architecture

The *K-Match* ontology alignment tool is based on a collaborative approach to finding matches between ontology terms. This tool was inspired by the work of Do Hong Hai's doctoral thesis [Do 2006], in which he proposes a composite matching approach for schema matching.

Nowadays there are many Web sites that use data and the functionality borrowed or combined from other similar sites. This approach, also knows as mashup, is well exemplified by Expedia *(http://www.expedia.com)*, that uses *United Airlines*, *US Airways* websites and dozens of other airlines sites, in order to provide a new, more unified service. Using the same philosophy we developed the *K-Match* tool, a mashup of existing ontology alignment tools. Better that proposing "yet another ontology matching tool", *K-Match* capitalizes from years of collaborative research and results from the Semantic Web community, particularly from the OAEI competition [Euzenat et al. 2009] [Euzenat & Shvaiko 2007].

The collaborative aspect of the *K-Match* tool comes from the characteristics of the Web 2.0 itself, as the tool is a mashup application that uses and combines functionality from several alignment tools, already available through Web APIs[26]. The *K-Match* tools allow us to use different alignment applications (matchers), with the option of including new ones, and combining the results applying different matching strategies.

[26]API : Application Programming Interface

Figure 15 shows the architecture of the K-Match tool. It consists of five modules: *Ontology Parser* to manipulate the input ontologies, *Matcher Executor* to execute different matchers, *Combination* to aggregate the previous match results, *Selection* to execute selection strategies in order to obtain match candidates and finally, *Alignment File* to elaborate the alignment file, result of the alignment process.

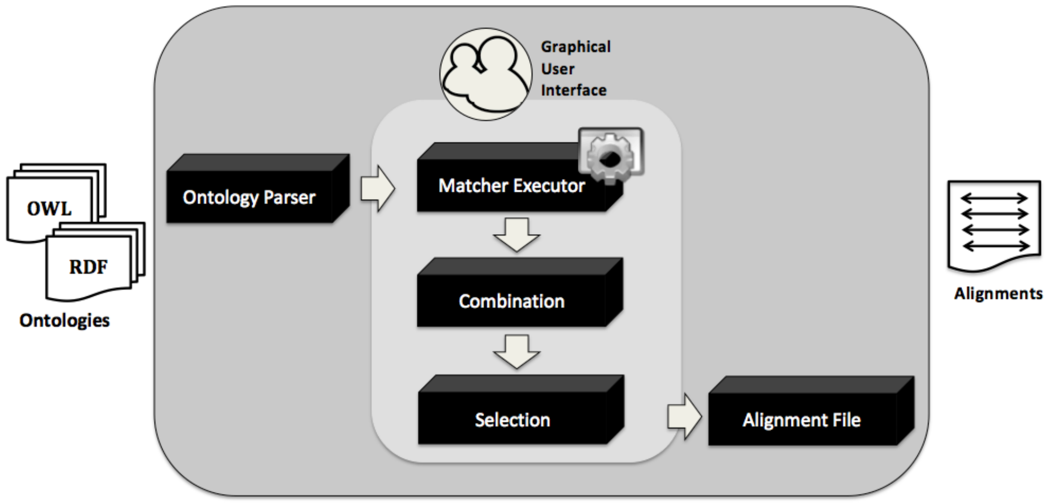

Figure 15: K-Match Architecture

1. **Ontology Parser:** In this module, the input ontologies are transformed to an internal representation using the RAP framework [Oldakowski et al. 2004].

2. **Matcher executor:** In this module, the K-Match tool uses three top ranked matchers of the OAEI 2009[27] contest (Figure 16), namely Lily [Wang & Xu 2009], Aroma [David 2009] and, Anchor-Flood [Seddiqui & Aono 2009]. The user select between these matchers, which are iteratively executed producing an alignment file in each execution. This alignment file is comprised of a set of mapping elements with similarity values that range from 0 to 1, depending on the similarity degree between elements pairs. The result produced is a set of alignment files, stored in a cube with KxNxM dimensions. The dimension K represent the matchers used in the process, the dimension N represents the elements from the standard ontology and, the dimension M represents the elements from the database ontology. It is important to note that most of the matchers use an syntactic approach, probably because of the lack of instances stored in standard vocabularies hosted in the Web,

[27]http://oaei.ontologymatching.org/2009/

that hinders the adoption of semantic, instance-based approaches. This is a limitation of existing tools, not of the *K-Match* tool.

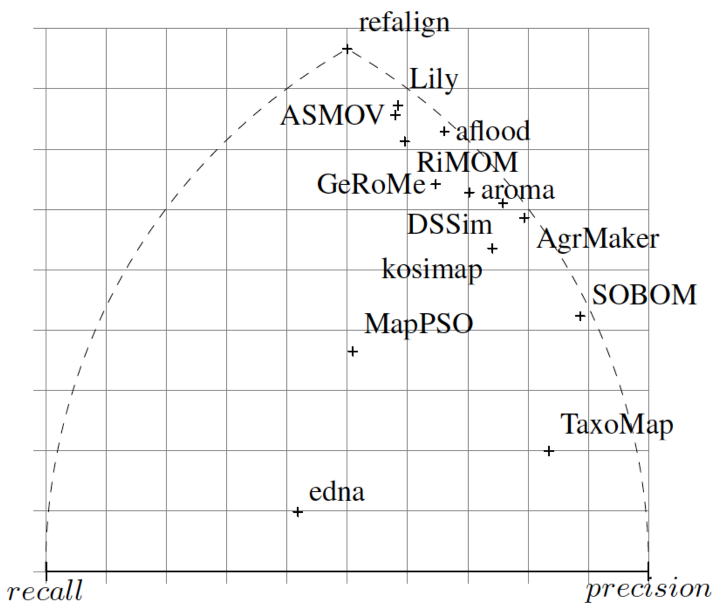

Figure 16: Results of the OAEI2009: Precision and recall [Euzenat et al. 2009].

3. **Combination:** In this module, the similarity cube generated by the previous module is compressed in a similarity matrix following an aggregation strategy. The following list presents the combination strategies provided by the *K-Match* tool to combine individual similarity values for pairs of terms into a unified value.

   – **Max:** This strategy returns the maximal similarity value of any matcher. It is optimistic, in particular in the case of contradicting similarity values.
   – **Weighted:** This strategy determines a weighted sum of the similarity values and needs relative weights for each matcher, which should correspond to the expected matchers importance.
   – **Average:** This strategy represents a special case of the *Weighted* strategy and returns the average similarity over all matchers, i.e. considers them equally important.
   – **Min:** This strategy uses the lowest similarity value of any matcher. As opposed to Max, it is pessimistic.
   – **Harmonic mean:** This strategy returns the harmonic mean over the matchers, with values greater than zero.

4. **Selection:** In this module, several selection strategies are implemented to obtain match candidates for each term of the database ontology. The following strategies are available in the K-Match tool:

- **MaxN:** The ***n*** source ontology elements with maximal similarity are selected as match candidates. n=1, i.e. Max1, represents the natural choice for 1:1 correspondences. Generally, n>1 is useful in interactive mode to allow the user to select among several match candidates.
- **MaxDelta:** The source ontology element with maximal similarity is determined as a match candidate, plus all source ontology elements with similarity differing at most by a tolerance value ***d***. Value ***d*** is a calibration factor set by the users. The idea is to return multiple match candidates when there are several source ontology elements with the same or almost the same similarity value.
- **Threshold:** All source ontology elements with a similarity value higher than the threshold ***t*** are returned. Value ***t*** is set by the user.
- **Max-Threshold:** This strategy represent a special case of the *Threshold* strategy. It returns the source ontology element with the maximal similarity above a given threshold ***t***. Value ***t*** is fixed by the user.

5. **Alignment File:** In this module, the obtained match candidates can be serialized in two Alignment API [Euzenat 2004] formats.

   - **RDFRendererVisitor:** displays the alignment in a RDF based format.
   - **OWLAxiomsRendererVisitor:** generates an OWL ontology merging both aligned ontologies.

For instance, we used the K-Match tool to align the Author-Publication ontology to the FOAF vocabulary. With that purpose we applied the *Max* combine strategy and the *Max-Threshold* selection strategy with the threshold value set to **0.6**.

Code 3 and Code 4, illustrate fragments of the result Alignment file in the RDFRendererVisitor and OWLAxiomsRendererVisitor formats respectively, for the scenario described above.

For the complete Alignment file, see Appendix C for the RDFRendererVisitor format and Appendix D for the OWLAxiomsRendererVisitor format.

Code 3 Alignment File - RDFRendererVisitor format

```
<rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema#">
 <Alignment>
  <xml>yes</xml>
  <level>0</level>
  <type>11</type>
  <onto1>http://purl.org/example</onto1>
   <onto2>http://xmlns.com/foaf/0.1</onto2>
   <uri1>http://purl.org/example</uri1>
   <uri2>http://xmlns.com/foaf/0.1</uri2>
   <map><Cell>
       <entity1 rdf:resource="http://purl.org/example#institution"/>
       <entity2 rdf:resource="http://xmlns.com/foaf/0.1/Organization"/>
       <measure rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0.695652173913</measure>
       <relation>=</relation>
     </Cell></map>
   <map><Cell>
       <entity1 rdf:resource="http://purl.org/example#first_name"/>
       <entity2 rdf:resource="http://xmlns.com/foaf/0.1/firstName"/>
       <measure rdf:datatype="http://www.w3.org/2001/XMLSchema#float">1</measure>
       <relation>=</relation>
     </Cell></map>
   ...
```

Code 4 Alignment File - OWLAxiomsRendererVisitor

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:owl="http://www.w3.org/2002/07/owl#"
            xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema#">
 <owl:Ontology rdf:about="">
  <rdfs:comment>Aligned ontollogies</rdfs:comment>
  <owl:imports rdf:resource="http://purl.org/example"/>
  <owl:imports rdf:resource="http://xmlns.com/foaf/0.1"/>
 </owl:Ontology>
 <owl:Class rdf:about="http://purl.org/example#institution">
   <owl:equivalentClass rdf:resource="http://xmlns.com/foaf/0.1/Organization"/>
 </owl:Class>
 <owl:Class rdf:about="http://purl.org/example#first_name">
   <owl:equivalentClass rdf:resource="http://xmlns.com/foaf/0.1/firstName"/>
 </owl:Class>
 ...
```