RSDL Workbench Results for OAEI 2014*

Simon Schwichtenberg[1], Christian Gerth[2], and Gregor Engels[1]

[1] University of Paderborn, s-lab – Software Quality Lab, Germany
{simon.schwichtenberg,engels}@upb.de
[2] Osnabrück University of Applied Sciences, Germany
c.gerth@hs-osnabrueck.de

Abstract The RSDL workbench was developed as a part of a service composition platform for service markets and provides tools to specify structural and behavioral aspects of services based upon the Rich Service Description Language (RSDL). Such comprehensive service descriptions allow a multi-faceted matching of service requests and offers in terms of their data models, operations, and protocols. Domains and application contexts of such service requests and offers are not known to the matchers in advance. Our data model matcher exploits several background ontologies to find corresponding data model elements. Data model alignments are represented in the form of relational Query View Transformation (QVT) scripts that are used to normalize behavioral models, which is a prerequisite for operation matching. For the OAEI campaign, we excluded background ontologies, because the involved additional costs did not justify the gain yet. In this paper, we present our system and the results for the OAEI campaign.

1 Presentation of the system

RSDL Workbench (RSDLWB) is a collection of tools for the specification, discovery and composition of services. A service discovery brings service requesters and providers together by matching requirements and existing services. These requirements or the offered functionality can be described in terms of the structural as well as behavioral aspects of the service through RSDL [4]. An RSDL specification consists of a data model, operation signatures, Visual Contracts (VCs) [2], and protocols. For the specification of a service, a data model determines relevant data types and their relationships in terms of a Unified Modeling Language (UML) class model. A VC is typed over such and specifies the behavior of certain operations. In particular, a VC describes pre- and postconditions of operation calls by graph grammar rules whose graphs conform to the class model.

The domain(s) of the service requests and offers are not known to the matcher in advance. Even though they share the same domain and describe semantically equivalent concepts, their respective class models might be heterogeneous, because they are created independently most likely. VCs might be heterogeneous as well, due to the heterogeneity of their respective class models they conform to. Consequently, VCs of a requester and a provider and hence the behavior of service offers and requests cannot be compared directly and must be normalized.

* This work was partially supported by the German Research Foundation (DFG) within the Collaborative Research Centre "On-The-Fly Computing" (SFB 901)

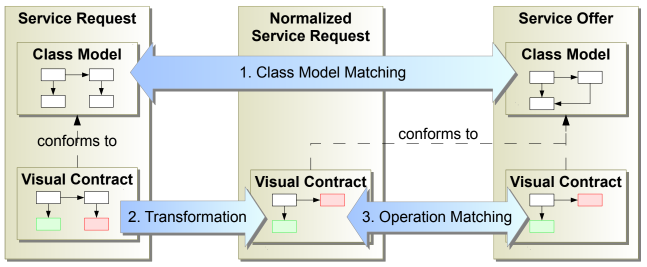


Figure 1: Matching Process [9]

Fig. 1 gives an overview of our approach: (1) The class models are matched and a list of class, attribute, and association mappings is returned. (2) Based on the list of mappings, a relational QVT [1] model transformation script is automatically generated which allows bidirectional model transformations. The VCs of the requester are normalized according to the providers' class model by executing the model transformation. The normalization of the VCs is a prerequisite for the operation matching. (3) Once all VCs conform to the same class model, they can be compared directly. In a next step, the operations are matched based on the normalized VCs, which is explained in detail in [5].

The list of operation mappings is the input for the protocol matcher, that checks if the operation invocation sequences requested by the requester match with the operation invocation sequences allowed by the provider. The data model matcher and the transformation script generation was previously presented in [9]. The system was realized as an Eclipse plug-in and implements the interface of EMF Compare[3] in order to reuse its graphical user interface. This paper focuses on its data model matching techniques and the results of the OAEI campaign.

1.1 State, purpose, general statement

As explained in Sect. 1, the purpose of the system is to match heterogeneous class models. The system automatically matches two UML class models that are part of respective RSDL specifications and generates a relational QVT model transformation script, which acts as a mediator enabling the translation of behavioral models. If necessary, the generated script can be manually revised.

In context of our system, the relevant OAEI tracks that we aim to compete in, are as follows: *benchmark*, *anatomy*, and *conference*. In the future, we also plan to participate in the *multifarm*, *library*, and *largebio* track. The tracks *interactive*, *instance matching*, and *ontology alignment for query answering* are less relevant for RSDLWB and support for these tracks is not scheduled.

In our knowledge, none of the existing matching system fulfills all the requirements of RSDLWB class model matcher, i.e. (1) process UML class models as input, (2) create 1:1, 1:n, n:1, n:m class mappings, (3) generate a transformation script from the mappings.

[3] `http://www.eclipse.org/emf/compare/`

1.2 Specific techniques used

According to the classification of [3], RSDLWB uses the following matching techniques: 1. String-based (normalization), 2. Language-based (tokenization), 3. Constraint-based (type similarity), 4. Linguistic resources / domain specific ontologies (background ontologies)[4], 5. Taxonomy-based (upward cotopic similarity)[4].

RSDLWB matches classes (`DataProperties`), attributes (`DataProperties`), and associations (`ObjectProperties`) pairwise and independently. The similarity of a pair is basically determined on the basis of their labels. In case of attributes, their type similarities [10] are considered as tie breakers. Before labels of two concepts are matched, they are split into *tokens*. Each single token is *normalized* by lowercasing and suppression of non-alphabetical characters. Next, the tokens are matched for their part. The overall label similarity arises from the average similarity of the token matching. If two tokens have identical normalized strings, they are assumed to match and get the highest similarity value.

The rest of this section addresses techniques that were not used in the OAEI campaign for reasons that are explained in Sect. 3. When two tokens are not identical, their Upward Cotopic (UC) similarity [6] is computed. The UC similarity is the quotient of the number of the tokens' shared hypernyms and the number of all their hypernyms according to a Background Ontology (BO). Such a BO is selected when it contains two concepts with the same normalized labels as the tokens to be matched. In particular, an individual BO is selected for each label pair. BOs are stored in a relational database. The transitive closure of the hypernyms is precalculated for each BO concept and also stored in the database. We imported different ontologies to our database like WordNet [8], DBpedia [7], etc.

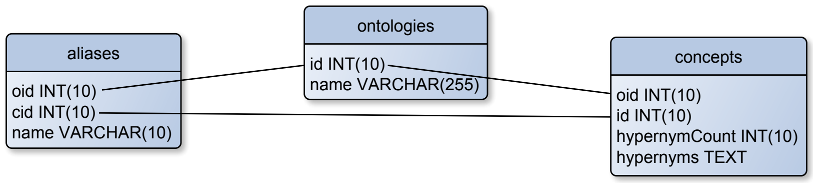


Figure 2: Database Tables and Foreign Key Relations

Fig. 2 shows the database schema: The *ontologies* table contains a row for each imported BO. The *alias* table contains all synonyms for the *concepts*, which are stored in a separate table. The column *hypernyms* stores all hypernyms as a list of concept ids. Additionally, *hypernymCount* contains the number of hypernyms. The edges illustrate foreign keys of the tables. A database index was added to *aliases.name*, which allows faster lookups of hypernyms for inquired aliases.

Listing 1 shows an exemplary SQL query that illustrates how two tokens from the input ontologies are anchored in a BO and how their hypernyms are retrieved. For each pair of tokens, an individual BO is selected. It might happen that a token is anchored in a BO by a homonym. The selection strategy prioritizes BOs with deeper taxonomic

[4] Technique was not used in the OAEI campaign (c.f. Sect. 3)

```
SELECT c1.hypernyms AS hypernyms1, c2.hypernyms AS hypernyms2,
    a1.oid AS id, LEAST(c1.hypernymCount, c2.hypernymCount) AS
    prio FROM aliases AS a1, aliases AS a2, concepts AS c1,
    concepts AS c2 WHERE a1.name = 'person' AND a2.name = '
    author' AND a1.oid=a2.oid AND c1.id=a1.cid AND c2.id=a2.cid
    AND c1.oid=a1.oid AND c2.oid=a2.oid ORDER BY prio DESC LIMIT
     1;
```

Listing 1: Querying Background Ontologies

hypernyms1	hypernyms2	id	prio
160, **843**, **138**, **269**, **515**, **325**, **932**	346, 930, 431, **160**, **843**, **138**, **269**, **515**, **325**, **932**	2	7

Table 1: Query Result

hierarchies, because shallow hierarchies produce UC similarity values that are close to each other. Tab. 1 shows the query result set that contains the hypernyms of *person* and *author*, the ontology id and the priority. Accordingly, the UC similarity is:

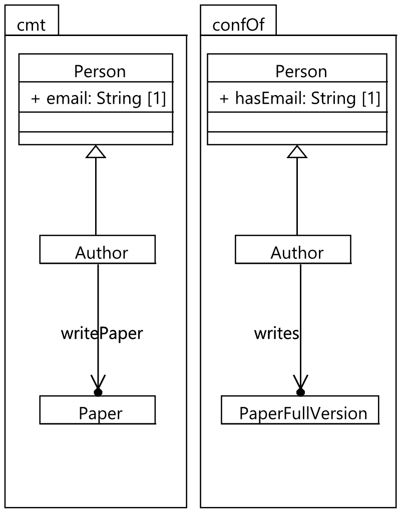
$$\sigma_{UC} = \frac{|hypernyms1 \cap hypernyms2|}{|hypernyms1 \cup hypernyms2|} = \frac{7}{10} = 0.7$$

To create n:m class mappings, a simple greedy algorithm is used. At first, the class pairs are sorted in a descending order according to their similarity. The algorithm iterates over the pairs and if none of the current pair's classes is part of a mapping, a new mapping is created. If one class is already part of a mapping and the second is not, the second is added to the mapping the first is already part of. If both classes are part of a mapping, the pair is ignored.

1.3 Generation of the Model Transformation

In this section, we want to explain briefly how the QVT transformation script is generated from the alignment. The generation is exemplified on the basis of the reference alignment for the `cmt` and the `confOf` ontologies that are part of the conference track. The UML diagram in Fig. 3a shows parts of these ontologies and is arranged in a way so that some mappings of the reference alignment can easily be seen.

Fig. 3b shows the generated QVTr script: Each class mapping corresponds to a **top relation**, which is a possible entry point for the transformation, e.g. <*Person, Person*> (line 2). During the transformation, free variables (**domains**) like `person1` are bound to instances of the source class model at first. Accordingly, `var_email` is bound to `person1`'s data attribute `email` (l. 5). The **enforce** keyword directs the transformation to create proper instances in the target data model (if necessary). Once `person2` is bound to a (newly created) instance, its attribute `hasEmail` is bound to `var_email` (l. 8). Variables for object attributes (l. 13, 16) are delegated to other relations to bind free variables (l. 19). The delegation is carried out in **when** clauses, which are preconditions for the **relations**. The creation of the script is not trivial, because n:m mappings have to be considered or mapped attributes do not necessarily belong to classes that have been mapped for their part, etc. For a more detailed description on the script generation and its current limitations, the reader is referred to [9].



(a) Excerpt of `cmt` and `confOf` Ontologies

```
transformation Cmt_ConfOf(cmt : cmt, confof : confof){
    top relation Person_Person{
        var_email:String;
        enforce domain cmt person1 : Person {
            email = var_email
        };
        enforce domain confof person2 : Person {
            hasEmail = var_email
        };
    }
    top relation Author_Author{
        enforce domain cmt author1 : Author {
            writePaper = var_paper1 : Paper{}
        };
        enforce domain confof author2 : Author {
            writes = var_paper2 : PaperFullVersion{}
        };
        when{
            Paper_PaperFullVersion(var_paper1, var_paper2);
        }
    }
    top relation Paper_PaperFullVersion{
        enforce domain cmt paper1 : Paper{};
        enforce domain confof paper2 : PaperFullVersion{};
    }
}
```

(b) Generated QVTr Script

1.4 Link to the system and provided alignments

The SEALS compliant[5] RSDLWB 1.1 is available at `http://goo.gl/3Uj9gS`. The provided alignments are available at `http://goo.gl/JLsELe`.

2 Results

The RSDLWB results are summarized in Tab. 2. The second column denotes how the values for precision, F-measure, and recall were calculated. The harmonic mean of all test cases is stated for *benchmark*, *conference*, and *multifarm*. The tracks *anatomy* and *library* comprise only one test case. Concerning the *conference* track, the values are calculated according two reference alignments ra1 and ra2. The *multifarm* track has two kind of tasks: The first kind matches the same ontology in different languages (same) and the second different ontologies in different languages (diff). Relating to *largebio*, RSDLWB could only complete the test case FMA-NCI within 10 hours.

2.1 benchmark

The test cases of the *benchmark* track are systematically generated from three seed ontologies – biblio, cose, and dog – by modifying or discarding ontology features. The evaluation is conducted in a blind fashion, i.e. neither the participants nor the organizers

[5] `http://oaei.ontologymatching.org/2014/seals-eval.html`

Track		Runtime [h:m:s]	Precision	F-measure	Recall
benchmark biblio	H-Mean	00:01:26	.99	.66	.5
benchmark dog	H-Mean	04:00:17	.99	.75	.6
anatomy	Mouse-NCI	00:22:17	.978	.749	.607
conference	H-Mean ra1	00:00:36	.81	.59	.47
conference	H-Mean ra2	00:00:36	.76	.54	.42
multifarm	H-Mean (diff)	00:18:00	.16	.04	.02
multifarm	H-Mean (same)	00:18:00	.34	.02	.01
library	TheSoz-STW	09:07:08	.781	.073	.038
largebio	FMA-NCI	00:36:57	.956	.38	.237

Table 2: RSDL Workbench Results for OAEI 2014

know the generated test cases in advance. RSDLWB achieved very good results regarding F-measure for the biblio and dog test cases. However, RSDLWB did not produce an alignment for cose.

2.2 anatomy

The Adult Mouse Anatomy and a part of the National Cancer Institute Thesaurus (NCI) describing the human anatomy are matched in the *anatomy* track. In regard to precision, F-measure, and recall, RSDLWB performs slightly worse than baseline StringEquiv. RSDLWB achieved high precision for the price of low recall compared to other systems.

2.3 conference

In the *conference* track, seven independent ontologies in the domain of organizing conferences are matched pairwise, resulting in 21 test cases. The produced alignments from the participants are evaluated against the reference alignments ra1 and ra2. The reference alignment ra2 is generated as the transitive closure computed on ra1. While ra1 was available to participants, ra2 was not. Regarding F-measure, RSDLWB performs better than baseline StringEquiv, but slightly worse than baseline edna, which means an average performance. Since RSDLWB relies only on string-based techniques the results are similar to the baseline algorithms.

2.4 multifarm

The goal of this track is to evaluate the ability of the matcher to deal with ontologies in different languages. The cross-lingual matching scenario is relevant for RSDLWB, but we did not investigate on this scenario yet. The low precision, F-measure, and recall values result from the fact, that labels in different languages share less common tokens. Even with enabled BOs, the matcher does not support other languages than English at the moment.

2.5 library

The task of the *library* track is to match the STW and the TheSoz thesaurus, which include a huge amount of concepts and additional descriptions. These ontologies define multiple labels per concept in different languages. However, RSDLWB does not support multiple labels per concept yet. Rather, it selects an arbitrary label, so that these labels are possibly in different languages, which leads to the same problems as for multifarm and explains the weak results.

2.6 largebio

The data set of this track comprises the large biomedical ontologies Foundational Model of Anatomy (FMA), SNOMED CT, and NCI. These ontologies are semantically rich and contain a huge amount of concepts. The input size of the ontologies vary across the six test cases. RSDLWB could only complete the smaller FMA-NCI test case within the given time frame of 10 hours. For this particular test case, RSDLWB achieved significantly lower F-measure than the average of all participants.

3 General comments

Several adjustments had been made to enable a participation of the RSDLWB in the OAEI campaign: (1) An abstraction layer for the input models was introduced in order to enable the matching of Web Ontology Language (OWL) ontologies. Since RSDLWB was designed to match UML models, it does not support other OWL features except labels of `Classes`, `DataProperties`, and `ObjectProperties`. (2) The matcher was configured to create only 1:1 mappings instead of n:m mappings, because n:m mappings had a negative impact on the most tracks. (3) Originally, as presented in [9], the matcher partially used some combinatorial algorithms which were replaced by simple greedy algorithms to improve the runtime. (4) The UC similarity was disabled, because the additional lookups of hypernyms in the BOs did not justified the matching results. With enabled UC similarity, more false positives than true positives were created, resulting in a decreased average F-measure.

3.1 Comments on the results

After the first participation of the RSDLWB in the OAEI campaign, we conclude that the system is not optimal for the OAEI test tracks yet and that there were no improvements in any of the OAEI disciplines. As the results show, the matcher heavily relies on labels and rarely on other ontology features. Furhermore, the system in its current shape is not suitable to match large ontologies.

3.2 Discussions on the way to improve the proposed system

RSDLWB depends very much on labels. To overcome this issue, similarity metrics must be introduced that take e.g. structural features of the ontologies into account. Since the importance of the similarity metrics varies between the test tracks and cases, the matcher should be adaptive and adjust the weights for these metrics. RSDLWB failed to complete test tracks with large ontologies in a reasonable time – even without using BOs. To improve the runtime of the matcher, we plan to parallelize the retrieval of hypernyms and the calculation of similarities. When BOs are used, the system often produces false negatives because it uses homonyms for the anchoring in BOs. Therefore, we want to adjust the matcher so that it is aware of the matching task's domain. Furthermore, we want to address cross-lingual matching by importing multilingual data sets of DBpedia or by integrating a translation service. We are confident that we can improve the system once the BO can be exploited effectively.

4 Conclusion

The first evaluation of RSDL workbench in the OAEI 2014 campaign showed good results for the benchmark track, but average to weaker results for the other tracks. The runtime and the quality of the matching results is improvable compared to other systems. We excluded the usage of background ontologies, because they increase the runtime of the system, but did not improve the matching results on average. As soon as we can effectively exploit BOs, we need to improve the systems' efficiency, because the retrieval of hypernyms has an extra effect on the runtime.

References

1. Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification Version 1.1. `http://www.omg.org/spec/QVT/1.1/PDF/` (January 2011)
2. Engels, G., Güldali, B., Soltenborn, C., Wehrheim, H.: Assuring Consistency of Business Process Models and Web Services Using Visual Contracts. In: Schürr, A., Nagl, M., Zündorf, A. (eds.) AGTIVE, LNCS, vol. 5088, pp. 17–31. Springer (2007)
3. Euzenat, J., Shvaiko, P.: Ontology Matching, vol. 18. Springer (2007)
4. Huma, Z., Gerth, C., Engels, G., Juwig, O.: A UML-based Rich Service Description Language for Automatic Service Discovery of Heterogeneous Service Partners. In: CAiSE Forum. pp. 90–97 (2012)
5. Huma, Z., Gerth, C., Engels, G., Juwig, O.: Towards an Automatic Service Discovery for UML-based Rich Service Descriptions. In: France, R., Kazmeier, J., Breu, R., Atkinson, C. (eds.) MODELS. LNCS, vol. 7590, pp. 709–725. Springer (2012)
6. Maedche, A., Zacharias, V.: Clustering Ontology-based Metadata in the Semantic Web. In: Elomaa, T., Mannila, H., Toivonen, H. (eds.) PKDD 2002, LNCS (LNAI), vol. 2431, pp. 348–360. Springer (2002)
7. Mendes, P.N., Jakob, M., Bizer, C.: DBpedia for NLP: A Multilingual Cross-domain Knowledge Base. In: Proc. of the 8th International Conference on Language Resources and Evaluation (LREC) (2012)
8. Miller, G.A.: WordNet: A Lexical Database for English. Communications of the ACM 38(11), 39–41 (1995)
9. Schwichtenberg, S., Gerth, C., Huma, Z., Engels, G.: Normalizing Heterogeneous Service Description Models with Generated QVT Transformations. In: Cabot, J., Rubin, J. (eds.) Modelling Foundations and Applications, LNCS, vol. 8569, pp. 180–195. Springer (2014)
10. Tibermacine, O., Tibermacine, C., Cherif, F.: WSSim: a Tool for the Measurement of Web Service Interface Similarity. In: Proceedings of the french-speaking Conference on Software Architectures (CAL) (2013)